UCLA
UCLA Electronic Theses and Dissertations

Title

Multiscale Modeling, Control and Soft Sensing in Semiconductor Manufacturing Processes

Permalink

https://escholarship.org/uc/item/6bh172nd

ISBN

9798247932543

Author

Ou, Feiyang

Publication Date

2026-05-28

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA

Los Angeles

Multiscale Modeling, Control and Soft Sensing in Semiconductor Manufacturing Processes

A dissertation submitted in partial satisfaction

of the requirements for the degree Doctor of Philosophy

in Chemical Engineering

by

Feiyang Ou

2026

© Copyright by

Feiyang Ou

2026

ABSTRACT OF THE DISSERTATION

Multiscale Modeling, Control and Soft Sensing of Semiconductor Manufacturing Processes

by

Feiyang Ou

Doctor of Philosophy in Chemical Engineering

University of California, Los Angeles, 2026

Professor Panagiotis D. Christofides, Chair

The rapid advancement of modern technology is driven largely by improvements in semiconductor devices, which require manufacturing techniques capable of producing increasingly complex nanoscale three-dimensional structures with high precision, high throughput, and reasonable cost. Atomic layer deposition (ALD) and atomic layer etching (ALE) have emerged as critical techniques in advanced semiconductor manufacturing because their self-limiting surface reactions enable atomic-scale control of material deposition and removal, resulting in highly uniform thin films and precise surface modification. To better understand and optimize these processes, this dissertation develops multiscale modeling frameworks that integrate reactor-scale transport phenomena with surface reaction mechanisms to analyze atomic layer processes and identify optimal operating

conditions, reactor geometries, and control strategies that improve precursor utilization and overall manufacturing efficiency. First, a multiscale simulation framework is developed for the Al_2O_3 ALE process to investigate process mechanisms and optimize operating conditions and reactor configurations. The framework is then extended to study temperature management and multiple control strategies that enhance process stability, efficiency, and robustness under disturbances by run-to-run control, model predictive control, and machine learning-based endpoint detections. Next, a multiscale model is developed for a novel area-selective ALD process for SiO_2, where internally integrated etching steps enable selective growth on SiO_2 surfaces while suppressing deposition on Al_2O_3 regions. This bottom-up selective deposition strategy provides a potential solution to lithography alignment challenges and offers guidance for optimizing etching durations to maintain high selectivity while maximizing process efficiency. In addition, this dissertation investigates machine learning-based soft sensing methods for semiconductor manufacturing processes using large-scale industrial data from Seagate Technology. A systematic framework for data cleaning, data aggregation, and ML modeling is developed. The proposed soft sensing models are applied to multiple semiconductor manufacturing processes, including plasma dry etching and slider milling processes, demonstrating their ability to predict wafer pass/fail outcomes and reduce reliance on costly offline metrology.

The dissertation of Feiyang Ou is approved.

Yuzhang Li

Xiaochun Li

Carlos Gilberto Morales Guio

Panagiotis D. Christofides, Committee Chair

University of California, Los Angeles

2026

Contents

1 Introduction **1**

1.1 Background and Motivation 1

1.2 Atomic Layer Deposition and Etching 3

1.2.1 Fundamentals of Atomic Layer Processes 3

1.2.2 Thermal Atomic Layer Etching of Al_2O_3 4

1.2.3 Area-Selective Atomic Layer Deposition 5

1.3 Multiscale Modeling Framework 5

1.4 Process Control: Run-to-Run and Model Predictive Control 6

1.4.1 Run-to-Run Control 6

1.4.2 Model Predictive Control 7

1.5 Soft Sensing and Virtual Metrology 7

1.5.1 Background on Soft Sensors 7

1.5.2 Industrial Machine Learning for Semiconductor Manufacturing 8

1.6 Motivation and Scope 8

1.7 Dissertation Structure 9

2 Multiscale computational fluid dynamics modeling of thermal atomic layer etching: application to chamber configuration design **11**

2.1 Introduction 11

2.2 Multiscale CFD Modeling for Thermal ALE 14

2.2.1 Background and Overall Modeling Framework 14

2.2.2 Microscopic Modeling 15

2.2.3 Macroscopic Gas-Phase Modeling for Four Reactor Configurations 18

2.3 Simulation Results and Reactor Design Evaluation 31

2.3.1 Simulation Results of Multiscale CFD Modeling and Validation 31

2.3.2 Comparison of Reactor Designs 33

2.3.3 Efficiency of the Inclined Plate Reactor 35

2.4 Conclusion 38

Nomenclature 40

3 Multivariable Run-to-Run Control of Thermal Atomic Layer Etching of Aluminum Oxide Thin Films **45**

3.1 Introduction 45

3.2 Multiscale CFD Modeling of Thermal ALE 48

3.2.1 Microscopic Surface Modeling 49

3.2.2 Macroscopic Modeling 51

3.3 Multivariable R2R Control Formulation 54

3.3.1 R2R-1 Modeling and Tuning 57
3.3.2 R2R-2 Modeling and Tuning 65
3.4 Simulation Results and Discussion 66
3.5 Conclusion 73

4 Sparse Identification Modeling and Predictive Control of Wafer Temperature in an Atomic Layer Etching Reactor 74

4.1 Introduction 74
4.2 Transient Modeling of Radiative Heating in ALE 78
4.2.1 2-D Reactor Model and Mesh Generation 79
4.2.2 Macroscopic Energy Model 80
4.2.3 Discrete Ordinate Radiation Model 81
4.2.4 Implementation and Monitoring 84
4.3 Sparse Identification Modeling 86
4.3.1 Open-Loop Data Generation 87
4.3.2 Dynamic Model Development 92
4.4 Model Predictive Control 96
4.4.1 Implementation of Model Predictive Controller 98
4.4.2 Feedback-Based Time-Varying Steady State in MPC 99
4.5 Results and Discussion 102
4.5.1 Open-Loop Control Performance 102
4.5.2 Closed-Loop Performance with Fixed Steady-State Power 103

4.5.3 Closed-Loop Performance with Feedback-Based Time-Varying Steady-State in MPC . . . 108

4.6 Conclusion . . . 109

5 Atomistic-Mesoscopic Modeling of Area-Selective Thermal Atomic Layer Deposition 111

5.1 Introduction . . . 111

5.2 Atomistic-Mesoscopic Modeling . . . 116

5.2.1 Surface Kinetics . . . 116

5.2.2 DFT Calculation . . . 121

5.2.3 Steric Hindrance . . . 126

5.2.4 Kinetic Monte Carlo Simulation . . . 134

5.3 Simulation Results and Discussion . . . 137

5.3.1 Surface Modeling for Protective layer . . . 138

5.3.2 Selectivity of ASALD . . . 139

5.3.3 Impact of Operating Conditions . . . 141

5.4 Conclusion . . . 143

6 Multiscale Modeling of Area-Selective Atomic Layer Deposition of SiO_2 on Al_2O_3 and SiO_2 Surfaces with Intermediate Etching Steps 156

6.1 Introduction . . . 156

6.2 Microscopic Monte-Carlo Based Collision Model . . . 161

6.2.1 Step A: Inhibitor Adsorption . . . 170
6.2.2 Step B: Precursor Adsorption . . . 175
6.2.3 Step C: Ozone Oxidation . . . 178
6.2.4 Step D: TMA Exposure for AlF_3 Removal and Conversion Modification . . 180
6.2.5 Definition of Film Thickness and Selectivity . . . 186
6.3 Macroscopic Computational Fluid Dynamics Model . . . 187
6.3.1 Reactor Design and CFD Model Development . . . 189
6.3.2 Verification of CFD Model . . . 191
6.4 Multiscale Real-Time Simulation . . . 198
6.5 Results and Analysis . . . 203
6.5.1 Multi-Batch Microscopic Simulation . . . 204
6.5.2 Multiscale Simulation . . . 219
6.6 Conclusion . . . 228

7 Industrial Data-Driven Machine Learning Soft Sensing for Optimal Operation of Etching Tools 230

7.1 Introduction . . . 230
7.2 Data Processing and Modeling . . . 233
7.2.1 Industrial Data Generation . . . 235
7.2.2 Data Preprocessing . . . 236
7.2.3 Classification Model . . . 240

7.2.4 Regression Model 248
7.3 Results and Analysis 253
7.3.1 Classification Model Performance 253
7.3.2 Regression Model Performance 268
7.4 Conclusion 277

8 Industrial Multi-Machine Data Aggregation, AI-Ready Data Preparation, and Machine Learning for Virtual Metrology in Semiconductor Wafer and Slider Production 279

8.1 Introduction 279
8.2 Data Preprocessing and Modeling 284
8.2.1 Overview 284
8.2.2 Data Generation and Preprocessing for Etching and Slider Tools 287
8.2.3 Model Training 299
8.3 Results and Discussion 308
8.3.1 Etching Tools 308
8.3.2 Slider Tools 316
8.4 Conclusion 330

9 Playbook for Industrial Data Processing and Modeling 334

9.1 Convert Data into Parquet and Separate by Time 337
9.1.1 Situation to Use 337

9.1.2 Inputs & Outputs Description 337
9.1.3 Sample Code 337
9.2 N/A Data Padding & Necessary Feature Extraction 340
9.2.1 Situation to Use 340
9.2.2 Inputs & Outputs Description 341
9.2.3 Sample Code 341
9.3 Categorical Feature Encoder 344
9.3.1 Situation to Use 345
9.3.2 Inputs & Outputs Description 345
9.3.3 Sample Code 345
9.4 Data Aggregation & Correlation Analysis 347
9.4.1 Inputs & Outputs Description 348
9.4.2 Sample Code 348
9.5 Normalization (Scaling) & Dimension Reduction (Single Tool) 353
9.5.1 Situation to Use 353
9.5.2 Inputs & Outputs Description 354
9.5.3 Sample Code 354
9.6 Scaling & Dimension Reduction in Data Aggregation 355
9.6.1 Inputs & Outputs Description 356
9.6.2 Sample Code 356
9.7 FNN Model Training (Binary Classification) 358
9.7.1 Situation to Use 358

9.7.2 Inputs & Outputs Description 359
9.8 XGBoost Model Training (Binary Classification) 363
9.8.1 Situations to Use 363
9.8.2 Inputs & Outputs Description 363
9.8.3 Tuning Instruction 365
9.9 Data Mixup 366
9.9.1 Situation to Use 366
9.9.2 Inputs & Outputs Description 366
9.10 FNN Model Training (Regression) 370
9.11 XGBoost Model Training (Regression) 373
9.12 Test Model Performance on Future Data (Binary Classification) 375
9.13 Advanced: LightGBM Training (Binary Classification) 377
9.13.1 General Description 377
9.13.2 Legend 378
9.13.3 Why LightGBM for a First Baseline? 379
9.13.4 Recommended Public Data Layout 380
9.13.5 Split Data by Time 381
9.13.6 Identify Numeric and Categorical Columns 382
9.13.7 Make Category Handling Consistent 383
9.13.8 Train LightGBM with Built-In Categorical Support 384
9.13.9 Predict and Evaluate 387
9.13.10 LightGBM versus XGBoost 389

9.13.11 Hyperparameter Tuning Instructions 390
9.13.12 Common Data Leakage Mistakes 394
9.13.13 Complete Minimal Example 394
9.13.14 Final Notes 398
9.14 Advanced: Physics Structure Informed Graph Attention Network Training 398
9.14.1 General Description 398
9.14.2 Legend 399
9.14.3 Recommended Public Data Layout 400
9.14.4 Split Data by Time 401
9.14.5 Prepare Public Grouped Features and Auxiliary Features 403
9.14.6 Fit Preprocessing on the Training Split Only 405
9.14.7 Build Graph Tensors 407
9.14.8 Define the GAT Model 410
9.14.9 Train the Model 415
9.14.10 Evaluate on Validation and Future Test Data 420
9.14.11 Data Leakage Checklist 423
9.14.12 Quick Summary for New Users 424
9.14.13 Notes for Internal Customization 424

10 Conclusion **425**

Summary 425
10.1 Synthesis and Cross-Cutting Takeaways 426

10.2 Chapter-Level Summary 427

10.3 Main Contributions 428

10.4 Future Work 428

List of Figures

2.1 The thermal ALE cyclical process for Al_2O_3. The process begins with Step A, in which HF fluorinates the surface of the substrate and modifies the surface producing AlF_3. Following Step A is a purge step to remove H_2O vapor and residual HF. Next, Step B consists of the etching cycle to convert the modified AlF_3 layer into the volatile species DMAF using the reagent, TMA. The cycle concludes with another purging step to remove trace TMA and DMAF produced during the etching cycle. The addition of heat allows for complete vaporization of volatile species. 16

2.2 (a) Twelve substrate regions for microscopic simulations. (b) Twelve substrate positions for the investigation of the flow distribution. 18

2.3 Schematic diagrams of the ALE reactors: (a) typical, G0; (b) multi-inlet, G1, (c) showerhead, G2; and (d) inclined plate, G3. The input(s) (dark gray) are located on the left-hand side of the reactor and the output (dark gray) is located on the right-hand side of the reactor. The wafer is presented in blue. 21

2.4 Meshes for each of the reactors produced from Ansys Fluent's Meshing Mode: (a) the typical reactor, (b) the multi-inlet reactor with three inlets, (c) the showerhead reactor, and (d) the inclined plate reactor. 25

2.5 Contours of Reynolds Number for Step A of various reactor configurations at 0.025 s in the standard condition in Table 2.3 32

2.6 Contours of Reynolds Number for Step B of various reactor configurations at 0.025 s in the standard condition in Table 2.3 33

2.7 Centerline HF pressure data for each reactor at various times for an HF feed flow rate of 150 sccm. The substrate position is numbered starting from the top of the divided wafer in Figure 2.2b to the bottom. 36

2.8 Contours of pressure of HF on the surface of the wafer for a Step A process time of 0.1 s and for an HF feed flow rate of 150 sccm. 36

2.9 Contours of pressure of TMA on the surface of the wafer for a Step B process time of 0.2 s and for a TMA feed flow rate of 70 sccm. 37

2.10 Complete cycle of G3 displaying the pressure of HF and TMA and coverage of AlF_3 for an HF and TMA feed flow rate of 150 sccm and 70 sccm, respectively. The blue solid line and the orange dashed line indicate the pressure of HF and TMA over time, respectively. The yellow solid line shows the coverage of AlF_3. The AlF_3 is formed in Step A and etched in Step B. 37

2.11 Process time and consumption of HF and TMA per year comparison between the CFD simulations of the typical reactor (G0) and of the inclined plate reactor (G3). The blue and orange bar indicate the consumption of the precursors for G0 and G3, respectively, for a single reactor. The black and yellow solid lines indicate the half-cycle time of G0 and G3 for both steps, respectively. 38

3.1 A schematic diagram of the inclined plate reactor (a) and the inclined plate reactor mesh generated from Ansys Fluent (b) are illustrated from [1]. 52

3.2 Multivariable run-to-run control of the inclined plate reactor. 57

3.3 The Multivariable Run-to-Run control system. 59

3.4 The input-output relationship between the fractional coverage of AlF_3 and the precursor valve opening time (a) for Step A and the etching fraction of AlF_3 and the precursor valve opening time (b) for Step B, which is derived from a standard linear regression model for the EWMA-based R2R controller, R2R-1. For Steps A and B, the R^2 values from Table 3.2 indicate marginal linear behavior for Step A and a lack of linear behavior for Step B. 60

3.5 The input-output relationship between the fractional coverage of AlF_3 and the precursor valve opening time (a) for Step A and the etching fraction of AlF_3 and the precursor valve opening time (b) for Step B, which is derived from a standard linear regression model divided into linear piecewise functions for the EWMA-based R2R controller, R2R-1. The R^2 values from Table 3.2 indicate a marginal to moderate linear relationship of multiscale CFD data. 61

3.6 The input-output relationship with the logarithmized multiscale CFD data and logarithmized time from the modified median-effect equation for Step A (a) with γ = 0.99 and ϵ = 0.35 and for Step B (c) with γ = 1.00 and ϵ = 0 for the EWMA-based R2R controller, R2R-1. The R^2 values in Table 3.2 indicate a strong linear relationship for Steps A and B. The multiscale CFD results with the standard modified median-effect equation are presented in (b) and (d) for Steps A and B, respectively. 64

3.7 The input-output relationship between the sum of partial pressure deviations and the flow rate for Step A (a) and Step B (b), which is derived from a standard linear regression model for R2R-2. The R^2 values from Table 3.2 indicate a moderate linear relationship. 67

3.8 Comparison of the responses for various regression methods of R2R-1 under the presence of a kinetic disturbance for Steps A (a) and B (b). The weight factors (λ) of 0.3 and 0.1 are used for Steps A and B, respectively. 70

3.9 Comparison of the responses of various R2R control systems under the presence of a kinetic and pressure disturbance for Steps A (a) and B (b). The weight factor (λ) of 0.3 is chosen for all case studies. R2R-1 and the multivariable R2R system are simulated with the modified median-effect regression model. 71

3.10 Progression of the adjustments made to the recipes (process time, precursor flow rate) in the presence of a kinetic and pressure disturbance through various EWMA-based R2R control systems for Steps A (a-b) and B (c-d). 72

4.1 Schematic figure that compares the (a) conventional heating plate structure and (b) novel radiative heating structure, where red arrows represent the direction of energy transfer. 76

4.2 Schematic diagram of the 2-D thermal ALE, cross-flow reactor: The red lines are the Center lamp group; the yellow lines are the Edge lamp group; the solid black lines are the Side lamp group; the dashed black line is the quartz window; the gray rectangle part is the wafer. A comprehensive list of geometry parameters are displayed in Table 4.2. 79

4.3 Angular discretization of an adjacent, 2-D rectangular cell on a heating lamp, where control angles are generated to reflect the light emissions produced from the heating lamps. 84

4.4 Temperature profiles of sample open-loop datasets used for model training to illustrate the surface temperature uniformity for various lamp power configurations. Dashed lines are for the uniform 2000 W/m^2 case and solid lines are for the nonuniform case where $P_1 = P_2 = 450$ W/m^2, $P_3 = 4000$ W/m^2. The side surface temperature is over 10 K lower than center temperature at $t = 2000$ s in the case of uniform lamp power; in contrast, nonuniform lamp power results in better surface temperature uniformity. 89

4.5 Comparison of the temperature profiles of the original and reconstructed data. The reconstructed temperature approaches the final steady-state value within 20 s, which enables the dynamic model trained using the reconstructed data to capture the correct steady state while retaining the trend of the original data. 92

4.6 (a) Derivative fitting and (b) state prediction plots for checking model performance. The fitted line in (a) closely aligns with the original data, suggesting that the SINDy model successfully captures the system dynamics. The fitted line in (b) shows strong correspondence between the predicted temperature and original data, where the maximum absolute error is 8 K when predicting 20 s of temperature. 96

4.7 Temperature profiles with open-loop control strategy, where the lamp power is fixed at P^{ss}_{final}. Due to the low power input to the system at all times, the system needs over 2000 s to reach the target temperature, which implies that a controller is needed. 103

4.8 Temperature profile under closed-loop MPC with steady-state power in input penalty term fixed at P^{ss}_{final}. This performance surpasses open-loop control since the temperature is closer to target. The grey middle dashed line is the target temperature; the two red solid lines on the bottom and top are boundaries of control range around the target. 105

4.9 Lamp power profiles under closed-loop MPC with steady-state power in input penalty term fixed at P^{ss}_{final}. The low power for the center (P_1) and edge (P_2) lamps compared to the power of the side lamp (P_3) corresponds to the phenomenon that the center temperature is significantly lower than the edge temperature in Figure 4.8, which is attributed to a low $P^{ss}_{final,1}$ and $P^{ss}_{final,2}$. 106

4.10 Temperature profile under closed-loop MPC with steady-state power in input penalty term fixed at P^{ss}_0. The plotting criterion is identical to that used in Figure 4.8. The system response is quicker, but severe overshoot occurs due to a higher total lamp power. 107

4.11 Lamp power profiles under closed-loop MPC with steady-state power in input penalty term fixed at P_0^{ss}. The center lamp power (P_1) is significantly higher than that observed in Figure 4.9 because $P_{0,1}^{ss}$ and $P_{0,2}^{ss}$ have a larger magnitude than $P_{final,1}^{ss}$ and $P_{final,2}^{ss}$, which results in a higher center temperature than the edge temperature in Figure 4.10. 107

4.12 Temperature profile under closed-loop MPC with the feedback-based time-varying steady-state (VSS) power approach. The temperatures reach the target within 10 s and are kept within the acceptable range as desired. The blue dotted lines are the lower and upper temperature bounds (571.0 K, 574.5 K) for the VSS to stop or resume decreasing P^{ss} in the cost function of the MPC formulation. 109

4.13 Lamp power profiles under closed-loop MPC with the feedback-based time-varying steady-state (VSS) power approach. The red points indicate that VSS stops decreasing P^{ss} in MPC cost function, while green points indicate resume decreasing. . . . 110

5.1 Schematic illustration of the ABC-type SMI-based ASALD. The molecules A, B, and C correspond to the inhibitor, precursor, and oxidant, respectively. 115

5.2 Minimum energy paths from the DFT calculations for the Hacac adsorption on (a) Al_2O_3 and (b) SiO_2, for the BDEAS adsorption on (c) SiO_2, and for the O_3 adsorption on (d) SiO_2. Color code for atoms: aluminum, dark gray; silicon, light brown; oxygen, red; hydrogen, white; nitrogen, dark blue; carbon, gray. The paths for the Hacac adsorption from **A1** to **A3** was sourced from [2]. Specifically, **A2**, **V2**, **V5** and **V7**, and **V9** indicate the physisorption reactions for Hacac on the NGA, BDEAS on the GA, ozone on the GA, and Hacac on the GA, respectively. 127

5.3 The intervals between two functional groups of the chelate and the monodentate are 4.8 and 3.8 Å, respectively. For the monodentate configuration, the CO group is considered as the group that is potentially able to hinder neighboring adsorption reactions. It is assumed that the CH_3 group at the far right of the monodentate product does not contribute to steric effects on adjacent reaction sites. Color code for atoms: aluminum, dark gray; oxygen, red; hydrogen, white; carbon, gray. 131

5.4 Top view of (a) Case 1 for Geometry **V2** (b) Case 2 for Geometry **V2** (c) Geometry **V3**. The blue circled reaction sites are hindered and deactivated by the current reaction site in yellow. Color code for atoms: silicon, light brown; oxygen, red; hydrogen, white; nitrogen, dark blue; carbon, gray. 133

5.5 Top view of a OH terminated SiO_2 surface, expressed by Geometry **V1** in Table 5.3. Sites 2 through 5 are numbered with Site 1 as the center. Color code for atoms: silicon, light brown; oxygen, red; hydrogen, white. 133

5.6 Conversion of the mesoscopic surface kinetics into a matrix of identifiable numbers. The blue and red circles in the lattice, corresponding to 1 and 2 in the matrix, indicate different reaction statuses. 135

5.7 The adsorption pattern of Hacac chelate (represented by unfilled geometry) and monodentate (represented by filled geometry) configurations on Al_2O_3 sites (blue dots) from the 2D stochastic simulation model. The empty capsule with two circles in black and the two conjoined blue circles symbolize monodentate and chelate molecules, respectively. 139

5.8 Hacac coverage versus time at $T = 423\ K$ and $P = 400\ Pa$. The red and blue solid lines denote the Hacac adsorption on Al_2O_3 and SiO_2, respectively. 140

5.9 Scatter charts for process time collected from kMC simulations under the operating window ($423 \leq T \leq 573\ K$ and $10 \leq P \leq 500\ Pa$) in (a) Step A, (b) Step B, and (c) Step C, respectively. 143

5.10 2D plots of process time as a function of temperature with different pressures in (a) Step A, (b) Step B, and (c) Step C, respectively. 144

5.11 Histograms depicting the distribution of dosage times for 100 iterations while at constant operating conditions for (a) Step A ($T = 523\ K, P = 100\ Pa$), (b) Step B ($T = 523\ K, P = 350\ Pa$), and (c) Step C ($T = 523\ K, P = 200\ Pa$). 145

6.1 Bare Al_2O_3 staggered arrangement grid surface with uniform 4.76Å site distance. . 164

6.2 Bare SiO_2 nonuniform staggered arrangement. Described in [3], surface grid separated into 2-column groups, within each group the vertical site distance 4.99Å, horizontal distance 3.13Å, between-group distance 5.40Å. 165

6.3 Orange molecules are monodentates consisting of two circles, blue molecules are chelates consisting of a rectangle and semicircles. 171

6.4 Diagram of Chelate molecule. Each 4Å circle denotes a CH_3 group on the same horizon. 173

6.5 Diagram of Monodentate molecule with two circles of 4Å diameter denoting a CH_3 group and 3.4Å diameter denoting projection of CH_3 on another side. 174

6.6 Diagram of Trilate DEA-Silane Structure. Center 5Å diameter circle denotes the Si-H_2 silane group, and each 4Å diameter circle denotes the ethyl group. 178

6.7 The diagram for 2D exposure mechanism. The selected bridge would block half of the hemisphere, and the adjacent bridges would block by angles between tangent lines. 183

6.8 The diagram for 3D exposure mechanism. The height difference between the selected site and surrounding deposited sites can results in different obstructions. . . 183

6.9 Overview of the CFD reactor geometry and computational mesh. (Top) Optically accessible ALD reactor with inlet and outlet configurations. (Middle) Unstructured tetrahedral volume mesh showing refined regions near the optical window and wafer. (Bottom) Detailed view of the prism boundary layers and local refinement applied at the wafer surface. 190

6.10 (Top) Experimental data from NIST showing pressure at multiple locations. (Bottom) Simulated absolute pressure contour with highlighted chamber pressure region (P2), showing excellent agreement (317 Pa experimental vs. 318 Pa simulated). 193

6.11 Comparison of simulated $MoCl_5$ concentration and experimentally measured absorbance at multiple chamber locations (Maslar and Kalanyan, 2025). Top: Reactor geometry showing monitoring points A (25.5 mm), B (15.7 mm), C (5.8 mm), and the wafer-plane cylinder (20 mm diameter, 106 mm length). Bottom: Normalized time-series data illustrating $MoCl_5$ buildup and decay, showing close agreement in transient behavior and peak timing between simulation and experiment. 194

6.12 Wafer surface velocity distribution showing dominant velocity range of 0.10–0.12 m s^{-1} (blue region), consistent with the NIST experimental measurement of 0.1 m s^{-1} . . 195

6.13 Transient $MoCl_5$ concentration response showing buildup during precursor pulse (0.5–0.7 s) and subsequent decay during purge. The average residence time of $\approx$0.8 s closely matches the experimentally reported value (Maslar and Kalanyan, 2025). 195

6.14 $MoCl_5$ image comparison between experiment and simulation during a pulsed injection sequence in the NIST optically accessible ALD reactor. **Top:** Experimental absorbance images acquired through the optical viewing window using a telecentric lens and CMOS camera (Maslar and Kalanyan, 2025). **Bottom:** CFD-predicted $MoCl_5$ concentration fields extracted from circular-masked mid-plane slices, aligned to match the camera field of view. 197

6.15 The schematic diagram of the multiscale simulation. The CFD program is started and controlled in PyFluent environment, sending temperature and pressure to microscopic python scripts, and receive the outputs of microscopic model as sources of species. 201

6.16 Inhibitor adsorption on NGA on bare grid. Orange molecules are monodentates, blue molecules are chelates. 205

6.17 Precursor substitution on inhibitor-adsorbed NGA surface. The green circle is the final Silane bridge structure, the green trilate is the intermediate Silane-DEA structure. 207

6.18 Normalized benchmark of BDEAS direct adsorption on NGA surface, with 898 total sites adsorbed and 838 of them are end points of bridge structure. 208

6.19 BDEAS precursor adsorption on GA with the microscopic mechanism described in [3]. The surface grid are separated into 2-column groups and the bridge can only formed between the two columns in each group. 209

6.20 The ozone would complete the deposition of the deposited silane structure by converting the -H to -OH structure, at the same time strip off all other adsorbed inhibitor molecules and trilate molecules to prepare the surface for next batch. 211

6.21 Second round of inhibitor adsorption on NGA surface. The pre-existing deposited SiO_2 takes some space that reduces the total number of inhibitors on surface and increases the proportion of monodentate inhibitor as the monodentate takes less spaces on the surface. 212

6.22 Number of monodentates and Chelates added in first 20 batches. Monodentate number doesn't change too much because of reduction in spaces. Chelates and total adsorbed molecule number decrease quickly with batches. 213
6.23 Second round of Step B, more contamination is formed on NGA surface. 214
6.24 Second round of Step C, more unwanted nucleation is completed on NGA surface. . 215
6.25 Pure microscopic simulation multi-batch result without etching. After a nucleation delay of about 10-15 batches, the NGA has identical deposition rate as the GA. The selectivity never achieves pass criterion (0.99) and decreases very quickly. The comparison between random seed of 42 and 0 shows the robustness and low variance of this simulation method under 50×50 scale. 216
6.26 Under fixed 0.6s etching time, the selectivity cannot be maintained above pass criterion (0.99) at anytime, which cannot fulfil any thickness requirement. 217
6.27 Under fixed 0.8s etching time, the selectivity can be maintained between 5-10 batches, which is suitable for very thin film requirement on GA, but for thicker film that requires over 10 layers the selectivity cannot fulfil the requirements. . . . 218
6.28 Under fixed 1.0s etching time, the selectivity is maintained for the whole 40 batches at perfect level, which indicates the etching on NGA is almost saturated. 218
6.29 Holistic view of the GA/NGA thickness and selectivity at different batch numbers and etch times. The Selectivity improves very quickly with increased etch time and saturates at about 1.0s. The GA thickness decreases with higher etch time and shows nucleation delay-like behavior at long etch times because of surface roughness developed by etching. 219

6.30 Without the involvement of exposure mechanism, the etching is more aggressive on both surfaces, which can be inferred by comparing with Figure 6.26 whose final average layers are higher on both GA and NGA than in this figure. Without exposure involvement, the selectivity is also apparently higher, which can misguide the implementation. 220

6.31 The comparison on first batch between microscopic and multiscale simulations. Because of the pressure threshold, the reaction starts with a lag and because of lower partial pressure of precursors, the initial slope of all steps is smaller for multiscale simulations. 223

6.32 Selectivity evolution over 40 batches under different etching times predicted by multiscale simulation. At an etching time of 1.0 s, the microscopic-only model predicts near-perfect selectivity across all batches; however, incorporation of precursor transport and delivery effects in the full ALD reactor model leads to a pronounced degradation in selectivity to an unacceptable level. Increasing the etching time to 1.6 s marginally maintains selectivity above the pass criterion, but the response remains unsaturated and exhibits a gradual downward trend, suggesting limited process margin for thicker depositions. Further increasing the etching time to 1.8 s drives the system close to saturation in the multiscale simulation, indicating an effective 0.8 s delay relative to predictions from the purely microscopic model. . 226

7.1 The overall manufacturing system for an industrial etching equipment. Each tool is an etching reactor that has multiple modules and can run various processes. Wafers start as pure silicon substrates, and after a series of production processes, they become a finished product. 234

7.2 Input Data Preprocessing **Step 1**: Eliminate or pad missing data. **Step 2**: Scale the numerical features, and encode the discrete features. **Step 3**: Combine numerical and discrete features into one complete input dataset. 237

7.3 Output Data Preprocessing **Step 1**: Eliminate missing data. **Step 2**: Encode binary output to FAIL(0) and PASS(1). **Step 2-2**: Scale numerical output with MinMax Scaler. 238

7.4 The general structure of the Feedforward Neural Network (FNN) is shown here. The hidden layer neurons take a weighted sum from the input layer or the previous hidden layer, which is then activated by nonlinear functions. The output layer takes the weighed sum from the last hidden layer to output the result. 242

7.5 ROC plot for all five datasets. The performance of the models trained on datasets T5-PM2 and T7-PM2 is similar to random guesses, while the models based on the other three datasets have superior performances. 256

7.6 Best possible AUC score for all five tool-module combinations. Model performances improve substantially as data increases for all five cases. 258

7.7 ROC plots of the representative models of T5-PM2 and T7-PM1, which were trained by aggregating all available datasets in the module. The performances are noticeably better than that of random guessing and single-set models. The FPR values for an 80% TPR value have also significantly improved. 259

7.8 Best possible AUC score as a function of how many datasets are aggregated among the four datasets. Model performance improves significantly for all tool-modules as data aggregation increases. 262

7.9 Comparison between the best AUC score for a **two-set** superset and the AUC score for the superset proposed by the candidate dataset selection method. Proposed AUC scores conform with the best-case scores for most tool-modules. 263

7.10 Comparison between the best AUC score for a **three-set** superset and the AUC score for the superset proposed by the candidate dataset selection method. Proposed AUC scores conform with the best-case scores for most tool-modules. 264

7.11 AUC scores for the models trained on the supersets proposed by the candidate dataset selection method. The AUC scores for all the tool-module combinations improve as more datasets are aggregated into the modeling set. 265

7.12 ROC plots for model performance on T2-PM2 and T7-PM1, which are trained by aggregating three datasets. The performances are noticeably better than the models trained on single datasets. The FPR values at 80% TPR are improved from good (around 40%) to perfect (around 20%). 266

7.13 The median percentage error after training with the **MAPE** loss function, compared to the benchmark model. Each tool-module labeled on the x-axis has three groups of bars, which represent the training, validation and test sets, respectively. Within each groups of bars, the blue bar is the median percentage error of the trained regression model, and the orange bar is the median percentage error of the benchmark model. Figure 7.14 to Figure 7.18 have the same formatting. 270

7.14 Comparison of the median percentage error between various regression models trained with the **MAPE** loss function for the training, validation and test sets and the benchmark model on runs with target oxide thicknesses of **less than** 200 Å. . . 271

7.15 Comparison of the median percentage error between various regression models trained with the **MAPE** loss function for the training, validation and test sets and the benchmark model on runs with target oxide thickess of **over** 200 Å. 272

7.16 The median percentage error after training with the **MSE** loss function, compared to the benchmark model. For all tool-module combinations and all datasets, the trained regression model performance is worse than that of the benchmark model. . 273

7.17 Comparison of the median percentage error between various regression models trained with the **MSE** loss function for the training, validation and test sets and the benchmark model on runs with **low** target oxide thicknesses. The performance of the regression model here is even worse than in Figure 7.16. 274

7.18 Comparison of the median percentage error between various regression models trained with the **MSE** loss function for the training, validation and test sets and the benchmark model on runs with **high** target oxide thicknesses. The performance of the regression model is better than that of the benchmark model on all examined runs for all three training, validation and test datasets. 275

8.1 The industrial etching equipment's manufacturing system consists of multiple etching reactors, each equipped with several modules capable of running different processes. The production begins with pure silicon wafers, which undergo a series of processing steps before transforming into the final product. 286

8.2 This figure illustrates the input data preprocessing workflow for etching tools and slider production tools. Common steps include invalid data removal, normalization, and dimension reduction. Etching tools incorporate encoders for categorical data management, whereas slider tools utilize Mixup to mitigate data scarcity. 288

8.3 This figure illustrates the output data preprocessing workflow for etching tools and slider production tools. Common steps include invalid data removal, etching tools requires 0/1 encoding, and slider tools requires separate scaling. 289

8.4 Scatter plot of T15-PM1 output, illustrating how data points are primarily grouped into several clusters. 296

8.5 This figure shows the AUC scores for the FNN and XGBoost models applied to etching tool data using single tool and dual tool training. 311

8.6 Single tool transfer learning results in heat map form. The performance difference in the same row is minimal (different train/test length ratio), but the performance difference in the same column (different train data length) is significant. 314

8.7 Dual tool transfer learning results in heat map form. The performance shows slight improvements compared to single-tool training and follows similar trends observed in single-tool training. 316

8.8 This figure shows the MAE and R^2 scores for the FNN model applied to slider tool data. MAE generally decreases when training data is aggregated with another tool. R^2 generally increase with this aggregation as well. 319

8.9 This figure shows the MAE and R^2 scores for the XGBoost model applied to slider tool data. MAE generally decreases when training data is aggregated with another tool, but not as much as seen in the FNN model. R^2 generally increase with this aggregation as well. 320

8.10 MAE and R^2 scores for the FNN model trained with data from a single slider tool using linear Mixup. In general, there is a decrease in MAE and an increase in R^2 scores across most slider tools. 324

8.11 MAE and R^2 scores for the XGBoost model trained with data from a single slider tool using linear Mixup. Similar trends of decreased MAE and increased R^2 scores are observed for most tools, except for T05-PM2 where we saw a marked decrease in R^2 scores. 325

8.12 MAE and R^2 scores for the FNN model trained with data from two slider tools using linear Mixup. There is a decrease in MAE in 5 tools and an increase in 2 tools for both schemes. R^2 scores decreased in 5 tools for both schemes, with a slight increase in two tools. 326

8.13 MAE and R^2 scores for the XGBoost model trained with data from two slider tools using linear Mixup. MAE generally decreased across most tools, with increases being very slight. R^2 scores saw a mixed pattern of increasing and decreasing scores. 327

9.1 General Flow Chart 336

List of Tables

2.1 The mesh quality acceptability criteria range and mesh parameters calculated from Ansys Fluent for various reactor geometries. For orthogonality, the minimum value is presented on the left and the average value is presented on the right. 22

2.2 Thermophysical material properties of DMAF specified in Ansys Fluent. 27

2.3 Operating conditions for the Thermal ALE process. 30

2.4 Half-cycle times determined by the kMC simulation of the multiscale CFD model. 33

3.1 Standard operating conditions for the multiscale CFD simulation. 57

3.2 A comparison of the standard linear, piecewise, and modified median-effect regression model parameters that are calculated from the standard least squares method for Steps A and B. 65

4.1 The mesh quality acceptability criteria range and mesh parameters calculated from Ansys Fluent for the ALE reactor. For the orthogonality, the minimum value is presented. For the aspect ratio, the maximum value is presented. 80

4.2 Parameters and constants of CFD model. 82

4.3 Parameters and constants of the radiation model. 83

4.4 List of lamp powers used to generate open-loop data (W/m^2). 88

4.5 List of Δt applied during different time regions. 90

5.1 Summary and description of variables. 148

5.2 Self-determined parameters for DFT calculations. 149

5.3 Geometry description for reaction paths 149

5.4 Van der Waals radii of functional groups in Hacac molecule 150

7.1 Classification FNN Hyperparameters and Tuning Range. 243

7.2 Overall Size and Distribution of Datasets. 244

7.3 Regression FNN Hyperparameters and Tuning Range. 250

7.4 Single Dataset Training AUC Score. 255

7.5 Difference score between each pair of datasets. 260

7.6 Historical and Future Data Difference Score. 263

7.7 Percentage of runs with High Targets in each dataset. 276

8.1 Mean Percentage Difference of Average Value of All Features 292

8.2 Data Distribution Across Different Ranges 296

8.3 Hyperparameters for FNN 303

8.4 Hyperparameters for XGBoost 303

8.5 Hyperparameters for Transfer Learning Model Training 306

8.6 Hyperparameters for FNN 307

8.7 MAE Change 328

8.8 R2 Change 328

9.1 Custom Table with Full Document Width 335

ACKNOWLEDGMENTS

This dissertation would not have been possible without the guidance, encouragement, and support of many individuals.

First and foremost, I would like to express my deepest gratitude to my faculty advisor, Professor Panagiotis D. Christofides, for his invaluable mentorship and support throughout my doctoral studies. As my advisor, department chair, and a distinguished scholar, he has continuously inspired me with his vision, creativity, and dedication to research. His guidance, encouragement, and insightful discussions have been essential to the completion of this work, and I am deeply grateful for the opportunity to learn from him.

I would also like to thank the members of my Ph.D. dissertation committee, Professor Carlos Morales-Guio, Professor Yuzhang Li, and Professor Xiaochun Li, for their time, thoughtful feedback, and valuable suggestions that helped improve this dissertation.

I am grateful to my mentors and colleagues who introduced me to the world of engineering science research and supported me throughout this journey. In particular, I would like to thank Dr. Zhe Wu, Dr. Yimin Ren, Dr. Sungil Yun, and Dr. Matthew Tom for their mentorship and friendship during the early stages of my research career. I would also like to thank my collaborators and coworkers, Dr. Henrik Wang, Julius Suherman, Chun-Pei Lin, and Abdul Alghamdi, for their collaboration, insightful discussions, and support. I sincerely appreciate all members of the Christofides Research Group for creating a supportive and intellectually stimulating research environment.

I would also like to acknowledge the support from our industrial collaborators at CESMII and

Seagate Technology, including Dr. James Davis, Dr. Chao Zhang, Dr. Yu Huang, Dr. Sthitie Bom, and Agnes Zarate. Their collaboration and the industrial data they provided were invaluable to this research.

Financial support for this work was generously provided by the National Science Foundation (NSF) and the UCLA Dissertation Year Fellowship, and I gratefully acknowledge their support.

Finally, I would like to express my deepest gratitude to my parents, Wenli Ou (欧文利) and Peili Qiu (仇培莉), for their unconditional love and support. As the first graduate student in my family, this journey has been filled with challenges and unexpected experiences, and their encouragement and belief in me have always been a source of strength. I would also like to thank all of my friends who have supported and encouraged me throughout this long journey.

轻舟已过万重山，长风破浪会有时

This dissertation includes excerpts from the following manuscripts:

- Yun, S., M. Tom, F. Ou, G. Orkoulas, P. D. Christofides, "Multiscale Computational Fluid Dynamics Modeling of Thermal Atomic Layer Etching: Application to Chamber Configuration Design," Computers & Chemical Engineering, 161, 107757, 2022.

- Yun, S., M. Tom, F. Ou, G. Orkoulas, P. D. Christofides, "Multivariable Run-to-Run Control of Thermal Atomic Layer Etching of Aluminum Oxide Thin Films," Chemical Engineering Research and Design, 182, 1-12, 2022.

- Ou, F., F. Abdullah, H. Wang, M. Matthew, G. Orkoulas, P. D. Christofides, "Sparse Identification Modeling and Predictive Control of Wafer Temperature in an Atomic Layer Etching Reactor," Chemical Engineering Research and Design, 202, 1-11, 2024.

- Yun, S., F. Ou, H. Wang, M. Tom, G. Orkoulas, P. D. Christofides, "Atomistic-Mesoscopic Modeling of Area-Selective Thermal Atomic Layer Deposition," Chemical Engineering Research and Design, 188, 271-286, 2022.

- F. Ou, A. Alghamdi, C. P. Lin, G. Orkoulas, P. D. Christofides, "Multiscale Modeling of Area-Selective Atomic Layer Deposition of SiO_2 on Al_2O_3 and SiO_2 Surfaces with Intermediate Etching Steps," Chemical Engineering Science, 123422, 2026.

- Ou, F., H. Wang, C. Zhang, M. Tom, S. Bom, J. F. Davis, P. D. Christofides, "Industrial Data-Driven Machine Learning Soft Sensing for Optimal Operation of Etching Tools," Digital Chemical Engineering, 13, 100195, 2024.

- Ou, F., J. Suherman, C. Zhang, H. Wang, S. Bom, J. F. Davis, P. D. Christofides, ”Industrial Multi-Machine Data Aggregation, AI-Ready Data Preparation, and Machine Learning for Virtual Metrology in Semiconductor Wafer and Slider Production,” Digital Chemical Engineering, 15, 100242, 2025.

Curriculum Vitae

Education

University of California, Los Angeles *Sept. 2020 – June 2022*

B.S., Chemical Engineering Los Angeles, CA

University of California, Santa Barbara *Sept. 2018 – June 2020*

Chemical Engineering coursework; no degree conferred Santa Barbara, CA

Experience

SanDisk Corporation *July 2026 – Present*

Staff Engineer – Product Development Engineering Milpitas, CA

Publications

1. S. Yun, M. Tom, **F. Ou**, G. Orkoula, and P. D. Christofides. *Multiscale computational fluid dynamics modeling of thermal atomic layer etching: Application to chamber configuration design*. *Computers & Chemical Engineering*, 161, 107757 (2022).
2. S. Yun, M. Tom, **F. Ou**, G. Orkoula, and P. D. Christofides. *Multivariable run-to-run control of thermal atomic layer etching of aluminum oxide thin films*. *Chemical Engineering Research and Design*, 182, 1–12 (2022).
3. S. Yun, **F. Ou**, H. Wang, M. Tom, G. Orkoula, and P. D. Christofides. *Atomistic-mesoscopic modeling of area-selective thermal atomic layer deposition*. *Chemical Engineering Research and Design*, 188, 271–286 (2022).
4. M. Tom, S. Yun, H. Wang, **F. Ou**, G. Orkoula, and P. D. Christofides. *Multiscale modeling of spatial area-selective thermal atomic layer deposition*. *Computer Aided Chemical Engineering*, 52, 71–76 (2023).
5. M. Tom, S. Yun, H. Wang, **F. Ou**, G. Orkoula, and P. D. Christofides. *Machine learning-based run-to-run control of a spatial thermal atomic layer etching reactor*. *Computers & Chemical Engineering*, 168, 108044 (2022).
6. M. Tom, H. Wang, **F. Ou**, G. Orkoula, and P. D. Christofides. *Computational fluid dynamics modeling of a discrete feed atomic layer deposition reactor: Application to reactor design and operation*. *Computers & Chemical Engineering*, 178, 108400 (2023).
7. S. Yun, H. Wang, M. Tom, **F. Ou**, G. Orkoula, and P. D. Christofides. *Multiscale CFD modeling of area-selective atomic layer deposition: Application to reactor design and operating condition calculation*. *Coatings*, 13(3), 558 (2023).
8. M. Tom, H. Wang, **F. Ou**, G. Orkoula, and P. D. Christofides. *Machine learning modeling and run-to-run control of an area-selective atomic layer deposition spatial reactor*. *Coatings*, 14(1), 38 (2023).
9. M. S. Alhajeri, Y. M. Ren, **F. Ou**, F. Abdullah, and P. D. Christofides. *Model predictive control of nonlinear processes using transfer learning-based recurrent neural networks*. *Chemical Engineering Research and Design*, 205, 1–12 (2024).

10. H. Wang, **F. Ou**, J. Suherman, M. Tom, G. Orkoula, and P. D. Christofides. *Data-driven machine learning predictor model for optimal operation of a thermal atomic layer etching reactor*. *Industrial & Engineering Chemistry Research*, 63(45), 19693–19706 (2024).
11. **F. Ou**, F. Abdullah, H. Wang, M. Tom, G. Orkoula, and P. D. Christofides. *Sparse identification modeling and predictive control of wafer temperature in an atomic layer etching reactor*. *Chemical Engineering Research and Design*, 202, 1–11 (2024).
12. **F. Ou**, H. Wang, C. Zhang, M. Tom, S. Bom, J. F. Davis, and P. D. Christofides. *Industrial data-driven machine learning soft sensing for optimal operation of etching tools*. *Digital Chemical Engineering*, 13, 100195 (2024).
13. H. Wang, **F. Ou**, J. Suherman, G. Orkoula, and P. D. Christofides. *Integration of on-line machine learning-based endpoint control and run-to-run control for an atomic layer etching process*. *Digital Chemical Engineering*, 14, 100206 (2025).
14. **F. Ou**, J. Suherman, C. Zhang, H. Wang, S. Bom, J. Davis, and P. D. Christofides. *Industrial multi-machine data aggregation, AI-ready data preparation, and machine learning for virtual metrology in semiconductor wafer and slider production*. *Digital Chemical Engineering*, 15, 100242 (2025).
15. **F. Ou**, A. Alghamdi, C. P. Lin, G. Orkoula, and P. D. Christofides. *Multiscale Modeling of Area-Selective Atomic Layer Deposition of* SiO_2 *on* Al_2O_3 *and* SiO_2 *Surfaces with Intermediate Etching Steps*. *Chemical Engineering Science*, 203, 1–10 (2026).
16. A. Alghamdi, **F. Ou**, B. Kalanyan, J. E. Maslar, and P. D. Christofides. *A CFD-Based Digital Twin Framework for Transient* $MoCl_5$ *Transport in an Experimental Atomic Layer Deposition Process*. *Digital Chemical Engineering*, 19, 100304 (2026).

Chapter 1

Introduction

1.1 Background and Motivation

Semiconductor manufacturing has undergone continuous transformation driven by the increasing demand for computational power, data storage, communication bandwidth, and energy efficiency. The sustained scaling of device dimensions, historically associated with Moore' s law, has required not only advancements in device architecture but also substantial innovation in fabrication technologies [4]. As planar transistor scaling approaches physical and economic limits, modern devices increasingly rely on three-dimensional architectures such as FinFETs and gate-all-around (GAA) nanosheet transistors [5, 6]. These architectures significantly improve electrostatic control but simultaneously impose stringent requirements on thin-film deposition, etching precision, and surface quality.

At nanometer and sub-nanometer length scales, process variability that was previously negligible can have a pronounced impact on device performance and yield. Variations in film thickness,

roughness, conformality, and interfacial quality directly influence electrical characteristics, leakage currents, and long-term reliability. Consequently, semiconductor manufacturing increasingly demands atomic-scale precision in both material addition and removal processes.

In addition to physical precision, modern semiconductor fabrication is constrained by economic considerations, including throughput, cost per wafer, and equipment utilization. The increasing complexity of process flows also introduces challenges in process integration, tool-to-tool variability, and metrology limitations. Offline measurements, while accurate, are often expensive and slow, creating bottlenecks in production. These challenges motivate the development of predictive modeling, control strategies, and data-driven methods that can enhance process understanding, reduce variability, and improve operational efficiency.

This dissertation addresses these challenges through the integration of:

- Multiscale physics-based modeling of atomic layer processes
- Advanced control strategies including run-to-run (R2R) control and model predictive control (MPC)
- Data-driven soft sensing and virtual metrology for industrial semiconductor manufacturing

The overarching goal is to develop a unified framework that bridges fundamental process physics and industrial-scale data analytics to enable intelligent semiconductor manufacturing.

1.2 Atomic Layer Deposition and Etching

1.2.1 Fundamentals of Atomic Layer Processes

Atomic layer deposition (ALD) and atomic layer etching (ALE) are cyclic processes based on sequential, self-limiting surface reactions [7]. In ALD, alternating precursor exposures lead to controlled film growth, while in ALE, alternating surface modification and removal steps enable controlled material removal.

The defining characteristic of atomic layer processes is their self-limiting nature, which allows for sub-nanometer control over film thickness and etch depth. This makes them particularly suitable for advanced semiconductor applications involving high-aspect-ratio structures and complex geometries.

However, practical implementation deviates from ideal behavior due to:

- Finite precursor transport and diffusion limitations
- Reactor geometry effects
- Surface heterogeneity and defect formation
- Temperature-dependent kinetics

These factors necessitate multiscale modeling approaches to accurately describe process behavior.

1.2.2 Thermal Atomic Layer Etching of Al_2O_3

Thermal ALE of aluminum oxide using hydrogen fluoride (HF) and trimethylaluminum (TMA) has emerged as a model system for studying atomic layer etching [8]. The process consists of alternating fluorination and ligand-exchange reactions:

$$Al_2O_3 + 6HF \rightarrow 2AlF_3 + 3H_2O \tag{1.1}$$

$$AlF_3 + TMA \rightarrow \text{volatile products} \tag{1.2}$$

Although the reactions are self-limiting, the time required to reach saturation depends strongly on:

- Precursor partial pressure
- Reactor transport conditions
- Temperature
- Surface coverage evolution

Previous work in this dissertation demonstrated that reactor configuration significantly affects precursor distribution, cycle time, and efficiency through multiscale CFD modeling [9]. These findings highlight that atomic-level precision must be considered in the context of reactor-scale transport phenomena.

1.2.3 Area-Selective Atomic Layer Deposition

Area-selective ALD (AS-ALD) enables deposition on designated regions while suppressing growth elsewhere [10]. This approach offers a pathway to bottom-up patterning, reducing reliance on lithography.

However, maintaining selectivity over multiple cycles is challenging due to:

- Imperfect inhibitor coverage
- Defect-driven nucleation
- Accumulation of unwanted growth

This dissertation investigates AS-ALD of SiO_2 on SiO_2/Al_2O_3 substrates and introduces intermediate ALE steps to restore selectivity over multiple cycles [11]. The results demonstrate that selectivity is inherently a multiscale phenomenon influenced by both surface chemistry and reactor transport.

1.3 Multiscale Modeling Framework

Atomic layer processes span multiple scales:

- Atomic scale: reaction energetics
- Mesoscopic scale: surface coverage evolution
- Reactor scale: transport and flow dynamics

This dissertation develops multiscale frameworks that couple:

- Kinetic Monte Carlo (kMC) simulations
- Density functional theory (DFT)-informed parameters
- Computational fluid dynamics (CFD)

These models enable:

- Prediction of saturation times
- Reactor design optimization
- Understanding of nonuniformity

Such multiscale approaches are critical for bridging the gap between ideal chemistry and practical reactor behavior [1, 11].

1.4 Process Control: Run-to-Run and Model Predictive Control

1.4.1 Run-to-Run Control

Run-to-run (R2R) control is widely used in semiconductor manufacturing to adjust process parameters between wafers based on measured outputs. It is particularly suitable for atomic layer processes where direct real-time measurements are limited.

This dissertation develops multivariable R2R control strategies for thermal ALE, demonstrating improved robustness under disturbances such as kinetic variations and pressure fluctuations [12].

1.4.2 Model Predictive Control

Model predictive control (MPC) provides a systematic framework for handling multivariable systems with constraints [13]. In this chapter, MPC is applied to wafer temperature control using sparse system identification models [14].

The integration of data-driven modeling with MPC enables:

- Fast temperature regulation
- Reduced overshoot
- Improved uniformity

1.5 Soft Sensing and Virtual Metrology

1.5.1 Background on Soft Sensors

Soft sensors estimate difficult-to-measure variables using readily available process data. They are widely used in process industries and are increasingly important in semiconductor manufacturing [15].

Challenges include:

- Data heterogeneity

- Missing values
- Tool-to-tool variability
- Class imbalance

1.5.2 Industrial Machine Learning for Semiconductor Manufacturing

This dissertation develops machine learning frameworks for:

- Pass/fail classification
- Thickness prediction
- Multi-tool data aggregation

The results show that cross-tool aggregation significantly improves predictive performance [16, 17].

1.6 Motivation and Scope

The central motivation is to bridge:

- Physics-based modeling
- Process control
- Data-driven analytics

This integration enables more robust, efficient, and intelligent semiconductor manufacturing systems.

1.7 Dissertation Structure

The remainder of the dissertation is organized as follows.

Chapter 2 develops a multiscale CFD framework for thermal ALE of Al_2O_3 and applies it to reactor configuration design and operating-condition evaluation, with emphasis on how transport affects cycle times and precursor usage [1].

Chapters 3–4 present multivariable run-to-run control strategies for thermal ALE, including EWMA-based designs, regression/linearization approaches for nonlinear input–output behavior, and disturbance rejection case studies [12].

Chapters 5–6 address temperature management for thermal ALE reactors using data-driven modeling (sparse identification) and model predictive control, targeting fast heat-up with acceptable overshoot and improved wafer uniformity [14, 18].

Chapter 7 develops an atomistic-to-mesoscopic model of ASALD, capturing chemoselectivity and regioselectivity (including steric effects) and quantifying growth behavior under pressure and temperature variation [3].

Chapter 8 extends ASALD modeling to a coupled multiscale framework and introduces intermediate ALE steps for selectivity recovery, demonstrating how reactor transport nonidealities change the etch-time requirements for maintaining high selectivity over many cycles [11].

Chapter 9 develops industrial machine-learning soft sensors for etching tools, including dataset aggregation strategies and performance evaluation for classification and regression tasks under real industrial data constraints [16].

Chapter 10 studies industrial multi-machine data aggregation and AI-ready data preparation

for virtual metrology across multiple tools and production lines, and evaluates model choice and update strategies for practical deployment [17].

Chapter 11 concludes the dissertation and outlines directions for future research.

Chapter 2

Multiscale computational fluid dynamics modeling of thermal atomic layer etching: application to chamber configuration design

2.1 Introduction

The demand for high performance semiconductors caused by the Fourth Industrial Revolution has been increasing rapidly over the past decade. In line with this demand, extensive research for optimizing semiconductor manufacturing processes has been conducted recently. Fin field-effect transistors (FinFETs), as types of modern nanoelectronic semiconductors, were developed as a consequence to this increasing demand. FinFETs have facilitated the process of engraving three-dimensional (3D) circuit patterns on the substrate with a high aspect ratio, leading to higher computing speed with lower current leakage [19]. Atomic layer deposition (ALD) has greatly contributed to the development of FinFET technology. ALD is a thin film deposition process in which

a wafer is exposed to two precursor pulses in a sequential manner. Each precursor reacts with the surface species separately to avoid undesired reactions, known as self-limiting behavior, resulting in the production of high-quality thin films. As a result, ALD has led to the reduction in the size of FinFETs, which has scaled down from 22 *nm* to 5 *nm*. Despite the efforts to decrease the fin width of FinFETs to 5 *nm* or lower, a FinFET of 5 *nm* is rarely achieved, which causes undesirable mobility loss and short-channel effects [5]. Many leading semiconductor fabrication companies have extensively invested enormous resources to overcome this issue.

As a proposal to fabricate sub-5 *nm* nodes, a gate-all-around (GAA) approach has been pursued, which may one day become a potential successor to FinFETs, resulting in faster speed and greater power efficiency [20]. GAA transistors use vertically stacked nano-sheets or nano-wires instead of fins in FinFETs so nano-sheets are covered on all sides by the gate. The GAA technology has been predicted to reach an era of sub-5 *nm* thickness. Nevertheless, it has been difficult to commercialize GAA technology. In addition to ALD, atomic layer etching (ALE), as a counter part of ALD, has emerged as an essential process for GAA-based nano-chip production. ALE is an etching process in which the substrate is exposed to sequential precursor pulses to remove a mono-layer of the substrate in each etching cycle. ALE is a relatively new technique, and therefore, it has not been fully investigated in both empirical and computational ways. In order to completely understand the ALE process and make it possible to optimize the process configuration design, it is essential to fully develop a multiscale computational fluid dynamics (CFD) model for ALE processes.

A number of studies using a computational fluid dynamics (CFD) approach for atomic layer deposition processes have been carried out since 2010. [21] and [22] carried out CFD simulations for atomic layer deposition processes. An area-selective deposition process from a CFD point of view

has been studied [23]. [24] and [25] performed multiscale CFD simulations for plasma enhanced chemical vapor deposition (PECVD) and plasma enhanced atomic layer deposition (PEALD), respectively. Their research, however, is limited to simulations for understanding atomic layer processes but not for reactor design and optimization. [26] recently proposed an optimized showerhead design for top injection reactors. Despite the progress made on the research for these cross-flow reactors, with their strengths and drawbacks being generally described by [27], there has not been any quantitative comparison of reactor design performance via multiscale CFD-based modeling. Thus, this chapter is aimed to evaluate different types of cross-flow reactors for thermal ALE and to characterize their features and performances using multiscale CFD modeling. Several factors including film uniformity and reduction in process etching time will be investigated to determine the optimal reactor design.

Specifically, in this chapter, a multiscale computational fluid dynamics (CFD) model for thermal atomic layer etching of aluminum oxide (Al_2O_3) thin films is developed. Initially, a previously developed microscopic model based on the kinetic Monte Carlo (kMC) algorithm is adopted for the microscopic surface domain to describe the etching process [28] to capture the nature of the surface etching reactions at the atomic level. Next, a 3D CFD macroscopic model using Ansys Fluent 2021R2, as a commercial CFD software, is established for the gas-phase domain in which mass, momentum, and energy transport are considered. Lastly, the microscopic and the macroscopic models are combined to fully characterize the thermal atomic layer etching of aluminum oxide thin films and used to evaluate four reactor chamber designs.

2.2 Multiscale CFD Modeling for Thermal ALE

2.2.1 Background and Overall Modeling Framework

Experiments only permit data to be obtained from limited locations in the system that is equipped with sensors, and despite having these sensors, the amount of data collected experimentally may not gather the complete information of the system under various operating conditions. Meanwhile, 3D computational fluid dynamics (CFD) modeling based on the principles of fluid dynamics and transport phenomena allows one to obtain engineering data without any physical experiments for a considerably inexpensive cost. Moreover, the CFD simulation can be performed at various operating conditions, enabling one to ascertain an empirical model with a greater collection of data. It is, however, limited to provide the atomistic reaction information from a microscopic point of view even if macroscopic CFD modeling offers extensive data in terms of mass, momentum, and energy transport. To overcome this issue in this chapter, a 3D multiscale CFD model is built by combining a macroscopic CFD model with a previously developed microscopic model [28] of the etching process based on a kinetic Monte Carlo (kMC) algorithm, thus resulting in providing a comprehensive understanding for the thermal ALE process of aluminum oxide thin films. [25] and [29] have important and timely articles on multiscale CFD modeling for plasma enhanced atomic layer deposition (PEALD). The authors provided valuable insight of the PEALD process of hafnium oxide thin films and used these models to study real-time control. Despite their efforts, their multiscale CFD modeling lacked a degree of accuracy since they did not consider the consumption of reactants and the production of products, which would affect the pressure distribution of the system.

To address this issue, in this chapter, the heterogeneous surface reactions in the 3D CFD model are established in accordance with the reaction mechanisms of the microscopic model so that the etching of the surface species on the substrate can be simulated. Those reactions clearly have an impact on the momentum, energy, and mass transport in the gas-phase domain, in which the two precursors are consumed and the products of water and dimethylaluminum fluoride are yielded and transported from the substrate to the gas-phase domain. Pressure and temperature at different locations on the substrate are calculated at every time step and transferred to the microscopic model to calculate the etching progression in an atomistic level, of which detailed descriptions are provided in the following sections.

2.2.2 Microscopic Modeling

The thermal atomic layer etching (ALE) of aluminum oxide is driven by two reaction steps (Step A and Step B) using sequential and self-limiting thermal reactions that are each followed by purge steps. Hydrogen fluoride (HF) and trimethylaluminum [TMA, $Al(CH_3)_3$] are involved to remove the Al_2O_3 surface layer. In Step A (Modification cycle), HF exposure fluorinates the Al_2O_3 surface and forms AlF_3 on the substrate. During Step B (Etching cycle), TMA exposure facilitates ligand-exchange reactions and modifies the AlF_3 surface into a volatile layer composed of dimethylaluminum fluoride [DMAF, $AlF(CH_3)_2$]. Following Step A and Step B, a purge gas, N_2 is used to remove any byproducts produced and remaining precursors during a purge time. The schematic of the thermal ALE of aluminum oxide is illustrated in Figure 2.1 and the overall reaction can be

described by

$$Al_2O_3\ (s) + 6HF\ (g) + 4Al(CH_3)_3\ (g) \rightarrow 6AlF(CH_3)_2\ (g) + 3H_2O\ (g) \tag{2.1}$$

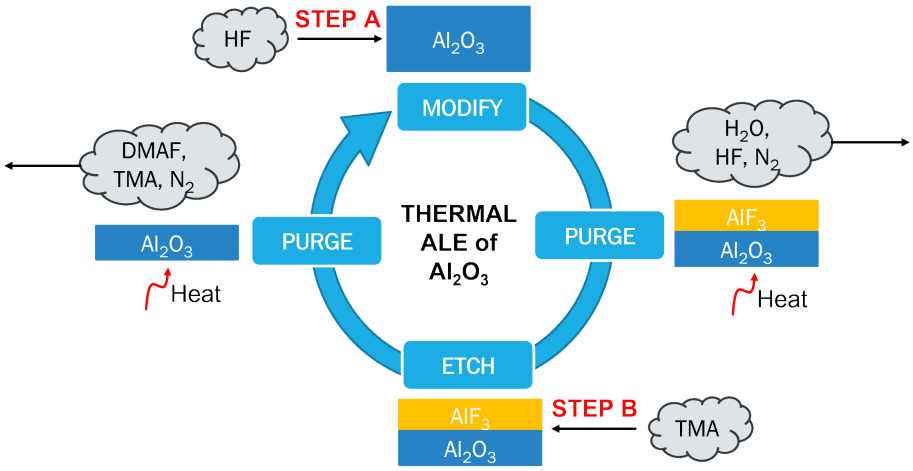


Figure 2.1: The thermal ALE cyclical process for Al_2O_3. The process begins with Step A, in which HF fluorinates the surface of the substrate and modifies the surface producing AlF_3. Following Step A is a purge step to remove H_2O vapor and residual HF. Next, Step B consists of the etching cycle to convert the modified AlF_3 layer into the volatile species DMAF using the reagent, TMA. The cycle concludes with another purging step to remove trace TMA and DMAF produced during the etching cycle. The addition of heat allows for complete vaporization of volatile species.

The microscopic model for the thermal ALE process of aluminum oxide thin films is formulated based on the variable step size method (VSSM) known as the kinetic Monte Carlo (kMC) algorithm, of which the detailed description was given in [29]. In this previous work, θ-Al_2O_3 ($\bar{2}$ 0 1) for the aluminum oxide structure was employed and approximated to a 300 $\times$ 300 lattice model. After modeling the surface, DFT (Density Functional Theory) calculations were performed to investigate all critical reaction steps that have significant impacts on the overall surface reaction time and to estimate their kinetic parameters.

In this chapter, the wafer is divided into twelve regions to spatially simulate the microscopic

model to obtain realistic and accurate etching data across the entire wafer surface, which is presented in Figure 2.2a. Pressure and temperature data at the twelve wafer regions are extracted from the CFD model at each time step and substituted into the kMC algorithm. All reaction rate constants are obtained from temperature and pressure by using Collision Theory and Transition-State Theory. Finally, the sum of the rate constants (k_{total}) is calculated by

$$k_{total} = \sum_{i=1}^{N} k_i \tag{2.2}$$

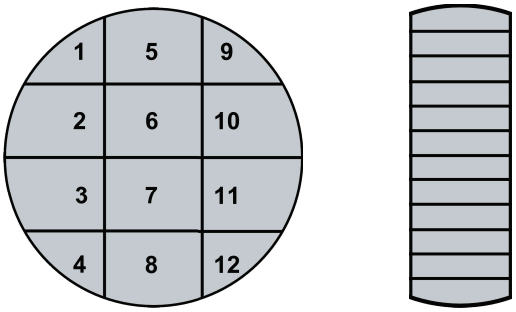
where k_i is the reaction rate constant of the reaction i, and N is the number of reaction pathways. For the reaction selection of a single reaction site on the wafer, a specific reaction can be randomly chosen as follows:

$$\sum_{i=1}^{j-1} k_i \leq \gamma_1 k_{total} \leq \sum_{i=1}^{j} k_i \tag{2.3}$$

where j represents the reaction j and $\gamma_1 \in (0, 1]$ is the first random number for the reaction selection. The reaction selection is implemented at every reaction site in which a random number is generated for each reaction site. If the value of $\gamma_1 k_{total}$ lies between $\sum_{i=1}^{j-1} k_i$ and $\sum_{i=1}^{j} k_i$, the reaction j is chosen for the reaction site. Otherwise, no reaction occurs at the reaction site. Once the reaction selection task for every reaction site is completed, the system clock evolves with a time interval determined as follows:

$$\Delta t = \frac{-\ln \gamma_2}{k_{total}} \tag{2.4}$$

where γ_2 is the second random number used for the time evolution ($\gamma_2 \in (0, 1]$).



(a) (b)

Figure 2.2: (a) Twelve substrate regions for microscopic simulations. (b) Twelve substrate positions for the investigation of the flow distribution.

2.2.3 Macroscopic Gas-Phase Modeling for Four Reactor Configurations

For macroscopic modeling, four different types of reactors are constructed and their performances are evaluated with respect to two metrics: film uniformity and etching speed (the specific designs considered are discussed in greater detail in the next subsection). The performance of a reactor is closely related to how fast a film on the wafer is deposited or etched. The magnitude of the deposition and etching rates is largely dependent on the type of fluid flow (laminar or turbulent), which is directly related to the precursor flow rate. For instance, [30] demonstrated that higher precursor flow rates resulted in higher deposition and etching rates, but the reduction in their process operating times was not significant. In addition to the deposition/etching rate, the film etching uniformity across the wafer is another key factor to evaluate reactor designs. Despite the fact that self-limiting behaviors have been reported in atomic layer processes [31], the spatial film etching uniformity could be degraded due to the non-uniform distribution of precursors [32] in the gas-phase above the wafer, and this could compromise the integrity of the etched product. For instance, turbulent flow can disrupt the uniformity of the fluid flow, thus undermining film quality of the

wafer. For this research, it is desirable to consider several reactor configurations that can introduce inherent reactor resistance to turbulent flow and maintain a laminar flow profile for operating flow rate regime to improve film etching quality. Therefore, these two aforementioned factors, etching rate and uniformity, are considered to compare the performances of the four reactor configurations considered in this chapter.

Reactor Chamber Designs

There are two general types of reactors for single-wafer systems: the top injection reactor and the cross-flow reactor [27]. The top injection reactor with distributors enables precursors to be uniformly injected above the wafer leading to highly uniform etching of the thin films. On the other hand, the cross-flow reactor has a smaller height of a few mm so that the gas displacement time is minimized. The cross-flow model also maximizes the lateral convective flow across the wafer. In this chapter, the cross-flow reactor is adopted to reduce the process and purge time since HF has a long residence time, which may remove self-limiting behavior resulting in spontaneous chemical vapor etching [33]. These cross-flow reactors are constructed with a feed source from one end of the reactor and the output source on the opposing end of the reactor to induce mass transport from one end of the wafer to the opposing end of the wafer. However, for conformal thin film etching, it is essential to obtain uniform flow profiles across the surface of the wafer [32]. Therefore, different distributors are employed on the cross-flow reactors to optimize the flow profiles of the precursor. Then, the modified reactors are compared to the simplest reactor geometry that has no distributor to determine if the distributor is effective in improving the performance with respect to a reactor design without distributors.

Specifically, in this chapter, four types of reactor chambers are created and their performances are evaluated by multiscale CFD simulations. First of all, the typical geometry (G0) is developed, which is a cylindrical-shaped chamber with a 500 *mm* outer diameter and 10 *mm* height as shown in Figure 2.3a. A wafer of 300 *mm* diameter is placed at the center of the bottom face of the chamber where an inlet of 20 *mm* diameter and an outlet of 40 *mm* diameter are located on the bottom face. Based on G0, a multi-inlet geometry (G1), a showerhead geometry (G2), and an inclined plate geometry (G3) are proposed. The multi-inlet geometry (G1) is constructed with three inlets in place of one inlet, in which each inlet has the same diameter as that of G0, but the total feed flow rate is divided evenly for all three inlets, with the total flow rate summing to the same inlet flow rate as that of G0. G1 is visualized in Figure 2.3b. As can be seen in Figure 2.3c, the showerhead geometry (G2) is constructed similarly to G0 but includes a showerhead divider of 2-*mm* thickness that distributes the inlet flow to achieve a uniform flow. The size of the pores is 4-*mm* in diameter for a total of 63 pores that are distributed into two rows. Lastly, the inclined plate geometry (G3) is developed with an arch-shaped inclined plate with 2-*mm* thickness between the inlet and the wafer surface with five degrees of deviation from the horizontal as shown in Figure 2.3d. The inclined plate is used to guide the center flow to regions near the edges in order to reach a uniform flow pattern for the film uniformity. There are several assumptions that are applied to the operation of the reactor:

1. There are carrier gas manifolds in the upstream facility so that the precursors and N_2 are well-mixed when introduced into the reactor.

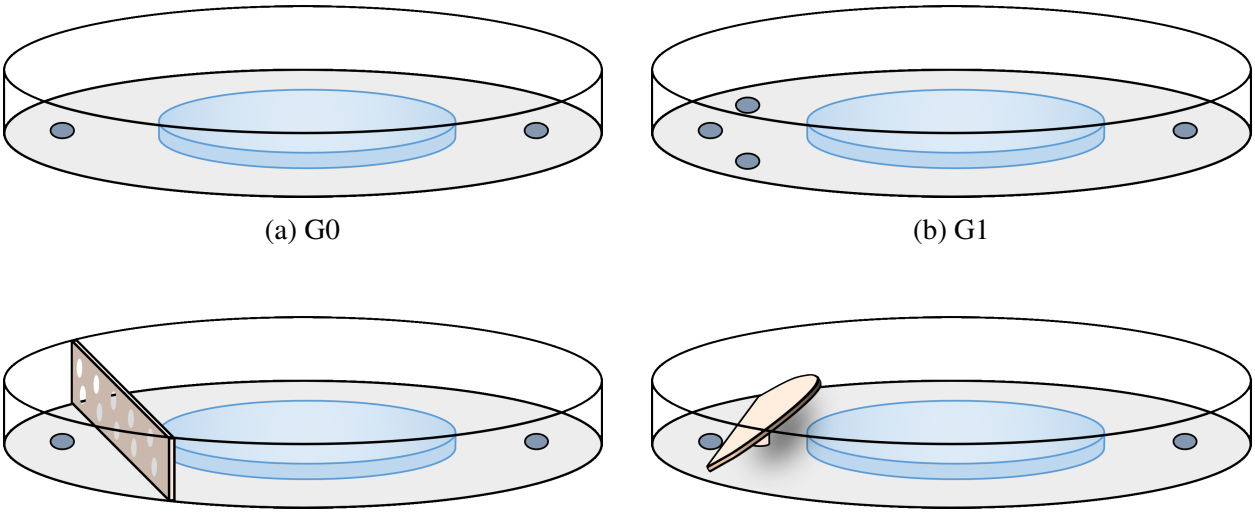
2. Other geometric objects such as sensors and mechanical structures are ignored.

3. The temperature of the substrate is maintained through a PID (proportional-integral-derivative) controller at a desired set-point.

4. The operating pressure is controlled and maintained via the vacuum pump in the downstream facility.

5. The flow through the reactor is characterized as laminar flow.

The above assumptions are implemented into the boundary conditions that would generate the mesh for the multiscale computational fluid dynamics simulation.



(a) G0 (b) G1

(c) G2 (d) G3

Figure 2.3: Schematic diagrams of the ALE reactors: (a) typical, G0; (b) multi-inlet, G1, (c) showerhead, G2; and (d) inclined plate, G3. The input(s) (dark gray) are located on the left-hand side of the reactor and the output (dark gray) is located on the right-hand side of the reactor. The wafer is presented in blue.

Meshing

The characteristics of the mesh for each reactor geometry will play a substantial role in the convergence, accuracy, and stability of the numerical solutions that will be calculated. Meshing Mode, an

application of Ansys Fluent, is used to construct the mesh for the reactors described in Section 2.2.3. An acceptable mesh can be determined by mesh quality criteria in accordance with the standards outlined by [34] as shown in Table 2.1.

Table 2.1: The mesh quality acceptability criteria range and mesh parameters calculated from Ansys Fluent for various reactor geometries. For orthogonality, the minimum value is presented on the left and the average value is presented on the right.

Quality Indicator	Orthogonality	Skewness	Aspect Ratio	Number of Cells
Criteria	0.001 ~ 1*	0* ~ 0.95	1* ~ 8	N/A
G0	0.130/0.727	0.271	2.060	266,291
G1	0.152/0.727	0.272	2.036	273,210
G2	0.002/0.741	0.252	1.525	1,550,322
G3	0.207/0.738	0.261	2.977	574,414

*Desired value for ideal mesh quality.

Specifically, Table 2.1 shows the key indicators used for analyzing the mesh quality. The quality of the mesh depends on the geometry of the cells and the boundary conditions used to define the overall geometry of the mesh. In this chapter, hybrid meshes, consisting of mixed element types, are generated to substantially reduce the computation time but still maintain acceptable mesh quality. Prism layers are utilized to resolve the boundary regions and tetrahedral cells are employed as a rudimentary element in the reactor chambers.

Among the factors that affect the mesh quality, skewness; orthogonality; aspect ratio; and resolution; are considered for the evaluation of the developed mesh structures for the various chamber geometries. The skewness of a cell is defined as the measure of the difference between a cell's geometry with that of an equivalent equilateral geometry of the same volume of the actual cell. The

equilateral skewness for the tetrahedral mesh is calculated from the following equation:

$$Skewness = \frac{optimal\ cell\ size - cell\ size}{optimal\ cell\ size} \tag{2.5}$$

The optimal cell size is defined as the size of an equilateral cell with the same circumradius. Thus, if the cell size is approximately equal to the optimal cell size, an ideal skewness of 0 is obtained. A low skewness is desirable to obtain an accurate and stable solution. The orthogonality is defined as the minimum value of all of the cells of the mesh from the following equation:

$$\frac{\vec{A}_i \cdot \vec{c}_i}{|\vec{A}_i||\vec{c}_i|} \tag{2.6}$$

where $\vec{A}_i$ is the area vector of a face and $\vec{c}_i$ is the vector from the centroid of the cell to the centroid of the adjacent cell. An ideal mesh has an orthogonality that is close to unity. For tetrahedral cells, the orthogonal quality is the minimum of the orthogonality of all cells in the mesh. Due to the variance in the orthogonality for each cell, the minimum orthogonality of all the cells should be greater than 0.001. The aspect ratio is another important indicator, which is a measure of the stretching of a cell. The aspect ratio is calculated as the ratio of the maximum to the minimum of the normal distance between the face centroids and the cell centroids, and the distance between the nodes and the centroid. A uniform aspect ratio is desirable for regions where the flow field varies greatly and an ideal aspect ratio is equivalent to unity for equilateral cells. Lastly, the resolution has a significant contribution to how critical regions are calculated and directly affects the total number of cells used to describe the mesh. The resolution of the mesh is a measure of the distribution of cells in particular regions of the mesh geometry and is measured in terms of coarseness or fineness.

Meshing mode contains a tool that produces adaptive sizing that automatically generates regions where the mesh is finer at the boundary regions and coarser in regions away from the boundary. It is important to obtain high resolution, especially in critical regions where boundary layers change dramatically in their behavior, specifically, wall-fluid boundaries, which will affect the accuracy of the computed numerical solution. Thus, the following equation, which is derived from the Blasius approximate solution for laminar flow over a flat plate, can be employed for determining the meshing in the flow domain near the walls:

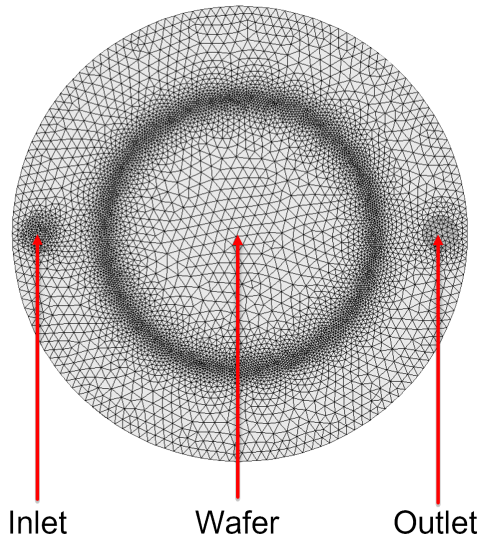
$$y_p \sqrt{\frac{u_\infty}{\nu x}} \leq 1 \tag{2.7}$$

where y_p is the distance to the wall from the adjacent cell centroid, u_∞ is the free stream velocity, ν is the kinematic viscosity of the fluid, and x is the distance along the wall from the starting point of the boundary layer.

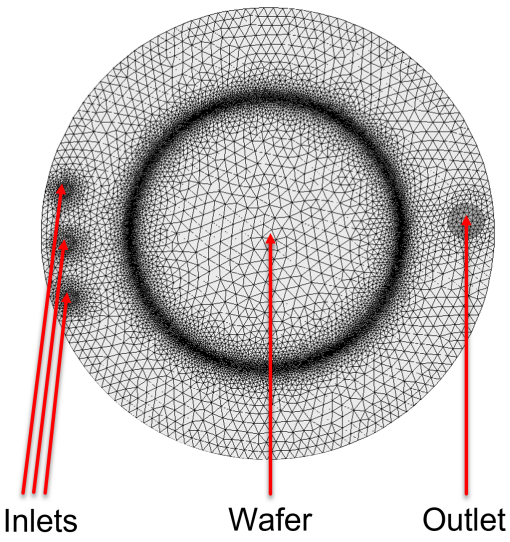
The consideration of the aforementioned factors affecting mesh quality leads to the development of the meshes for each reactor configuration, which are visualized in Figure 2.4. The results from the meshing process for all reactor configurations, which are listed in Table 2.1, indicate that all reactor geometries are within the acceptability criteria for the average values. This also implies that the meshes built via Fluent's Meshing Mode would have reliable computed results. Mesh independence studies were also carried out for all reactor designs to ensure that the simulation results are independent of mesh structure.

Thermophysical Property Calculation

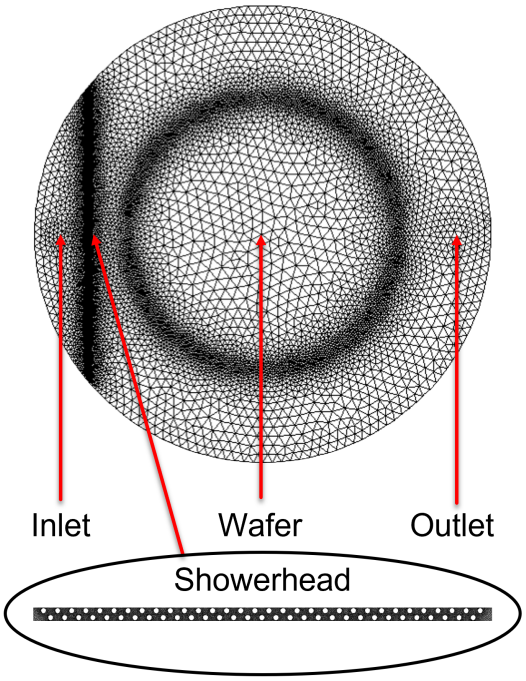
Thermophysical data are required for the materials used in the etching process and are employed in the computational fluid dynamics (CFD) simulations. However, some species produced during



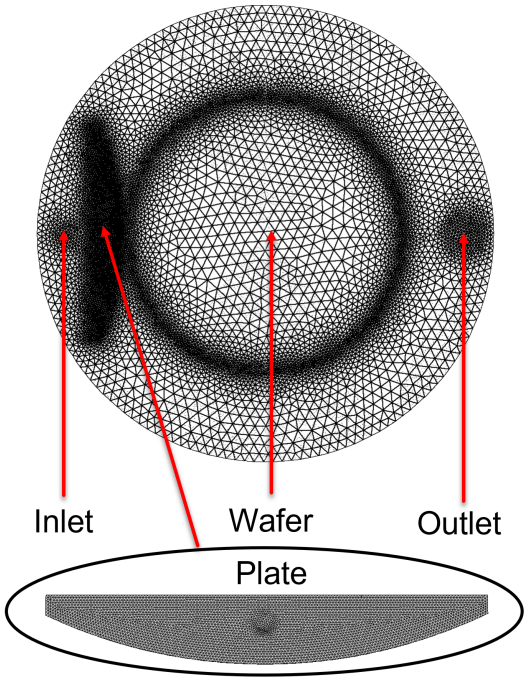
(a) G0



(b) G1



(c) G2



(d) G3

Figure 2.4: Meshes for each of the reactors produced from Ansys Fluent's Meshing Mode: (a) the typical reactor, (b) the multi-inlet reactor with three inlets, (c) the showerhead reactor, and (d) the inclined plate reactor.

the etching process have limited thermophysical data in the open literature. One of these species, dimethylaluminum fluoride (DMAF), has little to no available experimental data, thus computational chemistry calculations via the open-source thermochemistry simulation software, Quantum Espresso (QE), are utilized to calculate thermophysical parameters including the standard enthalpy, the standard entropy, and the specific heat [35, 36] for DMAF. Despite the accessibility of QE, there are limitations for calculating other thermophysical parameters including thermal conductivity and viscosity. For this chapter, these parameters are determined by adopting the parameters from a chemically similar molecule, dimethylaluminum chloride (DMACl), to DMAF [37], because chlorine and fluorine exhibit similar chemical behaviors as halogens. Lastly, the density of DMAF is calculated by assuming that the species behaves as an ideal gas due to the ambient environment of the reactor having low pressure and high temperature operating conditions, which are summarized in Table 2.3.

Quantum Espresso contains several packages that are required for calculating thermophysical data and these programs must be run sequentially. First, the PWscf (Plane-Wave self-consistent field) program is used to calculate the electronic properties and optimize the atomic positions of the molecule, which are modeled using Density Functional Theory (DFT) and PAW (Projector Augmented Wave) pseudopotential data. Next, the PHonon package is used to calculate the dynamical matrix of the phonons. Various programs are dedicated to the building of the dynamical matrix (PH program), to the solving of the interatomic force constants (IFC) from the dynamical matrix (Q2R program), and to calculating the eigenvalues of the dynamical matrix, which are the vibrational frequencies of the molecule (MATDYN program), by employing the finite displacement method and the density functional perturbation theory.

Lastly, the QHA (Quasi-Harmonic Approximation) package including the Partial Phonon DOS (Density of States) program is used to calculate atom projected density of states, the Mean Square Displacement program in the QHA package is utilized to calculate the deviation of the atom with respect to a reference position caused by the vibration of the atoms, and lastly, the FQHA (Fractional Quasi-Harmonic Approximation) program combines the results produced from the latter-mentioned programs to calculate the thermophysical properties including entropy, enthalpy, Helmholtz free energy, and specific heat at constant volume as functions of temperature. The results from the phonon calculation are displayed in Table 2.2. The formulation and derivation of the equations to solve the vibrational frequencies of the dynamical matrix and to calculate the thermophysical properties using statistical thermodynamics are discussed in greater detail by [38] and [39].

Table 2.2: Thermophysical material properties of DMAF specified in Ansys Fluent.

Thermophysical Parameter	Value	Units
Standard Enthalpy of Formation*	-499.290	kJ/mol
Standard Entropy of Formation*	196.421	J/(mol·K)
Specific Heat at Constant Pressure*	123.633	J/(mol·K)
Thermal Conductivity†	0.07268	W/(m·K)
Viscosity†	0.01100	kg/(m·s)

*Parameters calculated from Quantum Espresso.

†Property data of DMACl [37].

Three-Dimensional Computational Fluid Dynamics Simulation

Facilitated by the Hoffman2 Cluster at UCLA, the Fluent computations are implemented with 24 parallel central processing units (CPU) with 16 GB memory for each core processor so that the

parallel processing splits the gas-phase domain into multiple partitions to improve the computation efficiency. There are two solver technologies available in Fluent: pressure-based and density-based. The pressure-based solver has been traditionally used for incompressible and mildly compressible flow. The thermal ALE is operated at an extremely low pressure in a single-phase flow, and thus, the pressure change of the mixture is negligible. In addition, the feed composition of the precursor is low, therefore the density and pressure change of the species are negligible for this simulation. Therefore, the pressure-based solver is applicable for this chapter. The operating conditions of the reactor are listed in Table 2.3. In addition, under the pressure-based solver, the coupled algorithm is used to significantly decrease the convergence time, in which the momentum equation and the pressure-based continuity equation are solved simultaneously. Transient analysis for comprehensive computational fluid dynamics (CFD) modeling of thermal ALE is performed with a time step of 0.025 s with 200 iterations under transport phenomena. The conservation equations for mass and momentum are written as follows:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = S_m \tag{2.8}$$

$$\frac{\partial (\rho \overrightarrow{v})}{\partial t} + \nabla \cdot (\rho \overrightarrow{v}\,\overrightarrow{v}) = -\nabla p + \nabla \cdot \left(\overline{\overline{\tau}}\right) + \rho \overrightarrow{g} + \overrightarrow{F} \tag{2.9}$$

where ρ is the density of the mixture, $\overrightarrow{v}$ is the velocity of the mixture, S_m is the mass transfer source term, p is the static pressure, $\overline{\overline{\tau}}$ is a rank two stress tensor that is symmetric, $\rho \overrightarrow{g}$ is the gravitational body force, and $\overrightarrow{F}$ is the external body force. In addition, the conservation of energy is described by

$$\frac{\partial}{\partial t}(\rho E) + \nabla(\vec{v}(\rho E + p)) = -\nabla(\Sigma h_j J_j) + S_h \tag{2.10}$$

where E is the internal energy, h_j is the sensible enthalpy of species j, J_j is the diffusion flux of species j, and S_h is the heat transfer source term.

The inclusion of precursor consumption from Steps A and B to the multiscale 3D CFD simulation is employed to generate a more realistic flow profile. The consumption of the precursors and generation of products are calculated from the reaction rate constants determined by the modified Arrhenius equation that is defined as follows:

$$k_j = A_j T^{\beta_j} e^{-E_{A,j}/RT} \tag{2.11}$$

In the above equation, k_j is the reaction rate constant for reaction j, T is the ambient temperature of the reactor, β_j is the temperature exponent for reaction j, $E_{A,j}$ is the activation energy for reaction j, and R is the ideal gas constant. For simplicity, the temperature exponent, β_j, would be declared 0 for the simulation. Due to the large number of reactions and the lack of thermophysical data for numerous species obtained from prior microscopic research from [28], Eq. (2.1) will be simplified into two surface reaction steps that are defined below:

$$Al_2O_3\ (s) + 6HF\ (g) \rightarrow 2AlF_3\ (s) + 3H_2O\ (g) \tag{2.12}$$

$$2AlF_3\ (s) + 4Al(CH_3)_3\ (g) \rightarrow 6AlF(CH_3)_2\ (g) \tag{2.13}$$

First, the gaseous precursor, HF, physisorbs onto the surface of Al_2O_3 to produce AlF_3 and water vapor when reacting under high temperatures. Subsequently, the gaseous species, TMA, chemisorbs onto the AlF_3 surface forming the volatile species DMAF. After defining these reactions, operating

conditions are needed to fully define the system.

Table 2.3 shows the operating conditions of the multiscale 3D CFD simulations presented below. The operating pressure is set to be 133 Pa and the temperature is maintained at 573 K; thus, the material thermophysical properties are expected to be constant, hence, the data calculated in Table 2.2 is not required to account for the temperature dependence. As previously mentioned in Section 2.2.3, the temperature of the surface can be maintained through a control system that measures the temperature in real time while the operating pressure is controlled by discharging effluent through a vacuum pump. A constant flow of 150 sccm of N_2 gas is used to carry hydrogen fluoride (HF) and trimethylaluminum (TMA) into the reactor. The operating conditions are defined by a user-defined function (UDF) implemented in Ansys Fluent, in which the operating conditions are automatically adjusted according to the cyclical operation.

Table 2.3: Operating conditions for the Thermal ALE process.

Parameter	Value	Units
Operating Pressure	133	Pa
Operating Temperature	573	K
N_2 flow rate	150	sccm
HF half-cycle	2.0	s
1st Purge	5.0	s
TMA half-cycle	3.0	s
2nd Purge	5.0	s

2.3 Simulation Results and Reactor Design Evaluation

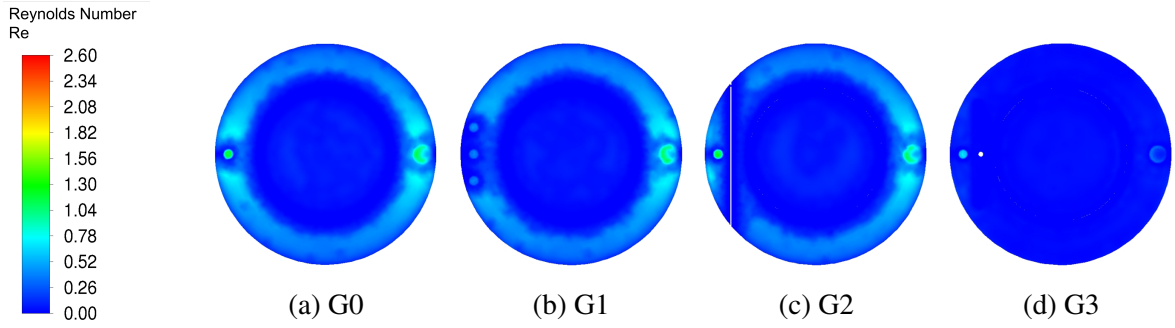
The multiscale computational fluid dynamics (CFD) simulations are first performed for each reactor model with an HF flow rate of 150 sccm and a TMA flow rate of 70 sccm to validate the 3D multiscale CFD model. Next, the results from the multiscale CFD modeling of the four reactors, G0 through G3, are discussed to observe which reactor design achieves a better distribution of precursor flow for the film uniformity and faster half-cycle times for Steps A and B. Finally, the reactor that produces the best performance is selected and simulated at different precursor flow rates to be compared with the typical type reactor (G0) in terms of efficiency and effectiveness.

2.3.1 Simulation Results of Multiscale CFD Modeling and Validation

In the multiscale CFD simulations, the flow is assumed to be laminar in the fluid dynamics point of view as discussed in Section 2.2.3. The assumption is validated by the contours of the Reynolds number at 0.025 s of half-cycle time elapsed at standard reactor operating conditons, which are illustrated in Figures 2.5 and 2.6. It is observed that the highest Reynolds number is localized at the precursor injection region for Steps A and B, with the largest Reynolds number among all reactor configurations being 1.30 for Step A and 2.59 for Step B with both values obtained from the showerhead reactor. Consequently, no turbulent regime is observed throughout all of the reactors due to the atmospheric operating condition which provides more control over the flow pattern. As a result, the flow through the reactor is characterized by laminar behavior and supports the assumption made in Section 2.2.3.

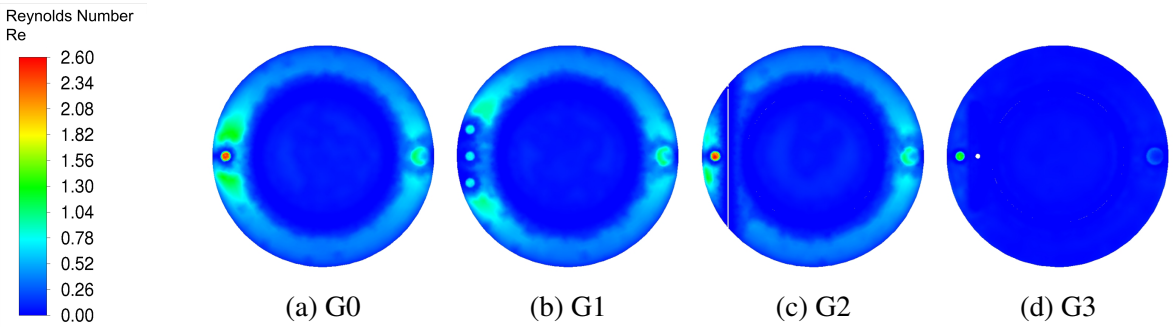
As shown in Figure 2.2a, the wafer is divided into twelve parts to calculate the etching pro-

gression for the microscopic simulations since the process data varies with location on the surface. Dividing the wafer into twelve sections enables one to collect more accurate and plausible numerical solutions than simulations with averaged pressure and temperature for the whole wafer. The results of multiscale computational fluid dynamics (CFD) simulations for the reactors are provided in Table 2.4. For all four reactors, the half-cycle times for Step A are calculated as 1.414 s through 1.446 s and the half-cycle times for Step B are also calculated as 2.498 s through 2.542 s using the kinetic Monte Carlo (kMC) method. The half-cycle times for both steps were computationally determined to be 1.38 s and 2.38 s, respectively, in the previously developed microscopic model [28] where the simulations were performed under ideal conditions without the influence of transport phenomena effects in the gas phase, which was supported by the experimental data from [40]. It is obvious that the half-cycle times from the multiscale CFD modeling are delayed due to the inclusion of mass transport as it takes some time for the wafer to be saturated by the precursors unlike the microscopic model at the steady-state. Furthermore, the consideration of the consumption of precursor species contributes to the slower process time. Hence, it is demonstrated that the overall multiscale CFD modeling including the thermophysical data is successfully developed.



(a) G0 (b) G1 (c) G2 (d) G3

Figure 2.5: Contours of Reynolds Number for Step A of various reactor configurations at 0.025 s in the standard condition in Table 2.3



(a) G0 (b) G1 (c) G2 (d) G3

Figure 2.6: Contours of Reynolds Number for Step B of various reactor configurations at 0.025 s in the standard condition in Table 2.3

Table 2.4: Half-cycle times determined by the kMC simulation of the multiscale CFD model.

Reactor	Step A	Step B
G0	1.437 s	2.514 s
G1	1.446 s	2.542 s
G2	1.436 s	2.528 s
G3	1.414 s	2.498 s

2.3.2 Comparison of Reactor Designs

The central region (sections 5 through 8 in Figure 2.2a) of the wafer, as shown in Figure 2.2b, is divided into 12 substrate positions (labeled in increasing order from the top to the bottom of the figure) to calculate the pressure of the precursors at each part so that the analysis of film uniformity is carried out. Figure 2.7 shows the flow patterns of the four reactors over substrate position, which exactly agrees with the pressure contours of the precursor in Figure 2.8. The pressure contours of HF precursor are also consistent with the pressure contours of TMA in Figure 2.9, thus the type of species in the thermal ALE cycle of Al_2O_3 plays a limited role in affecting the flow profile. As shown in Figure 2.7a, the flow of the typical reactor (G0) is formed in a circular shape due to

the isotropic flow. The largest pressure deviation (i.e., the difference between the maximum and minimum pressures) is given as 15.2 Pa at 0.2 s and the half-cycle times are 1.437 s for Step A and 2.514 s for Step B.

Figure 2.7b indicates that the flow of the reactor with three inlets (G1) is more evenly distributed than that of G0, and consequently, G1 has less pressure deviation than G0, of which the largest value is 11.4 Pa at 0.2 s. This is also shown in Figure 2.8b. However, the half-cycle times (1.446 s for Step A and 2.54 s for Step B) are greater than those of G0 due to the lower precursor velocity as can be seen in Table 2.4. Despite the input being distributed through three inlets, the flow appears to migrate more towards the outlet of the wafer, leading to an uneven flow profile.

Figure 2.7c reveals that the reactor with the showerhead (G2) improves the flow pattern when compared to that of G0 and G1, and the largest pressure deviation is calculated to be 8.5 Pa at 0.2 s. Also, with time progression, the uniformity of the flow improves faster, hence, the pressure deviation decreases, compared to G0 and G1. In the initial stages of flow development, the parabolic pressure profile mentioned in the discussion of G0 is also displayed with that of G2 in Figure 2.8c and Figure 2.9c. It is observed that the showerhead serves to decrease the amount of precursor in central regions of the wafer, and therefore G2 has better uniformity compared to G0 and G1. The half-cycle times (1.436 s for Step A and 2.528 s for Step B) are similar to the half-cycle times of G0 due to the flow resistance of the showerhead divider. Thus, the addition of the showerhead divider improves the uniformity, but marginally improves the half-cycle time for Step A and slightly worsens the half-cycle time for Step B.

As shown in Figure 2.7d, the reactor with the inclined plate (G3) has the most uniform flow pattern at every time step, in which the largest pressure deviation is 2.9 Pa at 0.2 s. Table 2.4 shows

that G3 has the least half-cycle times (1.414 s for Step A and 2.528 s for Step B) among the reactors despite the inclined plate acting as a flow resistance. Nevertheless, the results indicate that the effect of the uniformity outweighs the flow resistance, thus leading to the expediting of the etching process. Therefore, it is concluded that G3 may be able to give better film quality and thickness control than the other types of reactors. The complete etching cycle of Al_2O_3 for G3 is displayed in Figure 2.10. The inclined plate reactor (G3) shows the best performance in terms of the film uniformity and etching speed. Figure 2.10 shows the pressure of the two precursors and the etching progression over time, which is provided from the multiscale CFD simulation. The cycle consists of an HF dose of 2 s, an N_2 purge of 5 s, a TMA dose of 3 s, and an N_2 purge of 5 s.

2.3.3 Efficiency of the Inclined Plate Reactor

Multiscale CFD simulations are performed previously for different feed flow rates for the typical reactor (G0) and the reactor with the inclined plate (G3). The comparison of the half-cycle times of Steps A and B and the annual feed consumption of the precursor species, HF and TMA, are displayed in Figure 2.11. The annual feed consumption is calculated by assuming that 96 cycles of etching are conducted daily and that half-cycle times for each feed flow rate remain constant with each cycle in a single wafer system. The estimates for the precursor consumption are calculated for a single wafer. The results for Step A in Figure 2.11a indicate that the half-cycle time for Step A for G3 is consistently faster compared to that of G0, thus, the amount of precursor needed to ensure complete coverage is less than that of G0. By utilizing G3, at least 1.3×10^3 $std\ cm^3$ and at most 5.5×10^4 $std\ cm^3$ of HF can be saved for the range of the simulated flow rates with 600 sccm flow rate achieving the greatest amount of precursor that could be saved. The quantitative results from Step

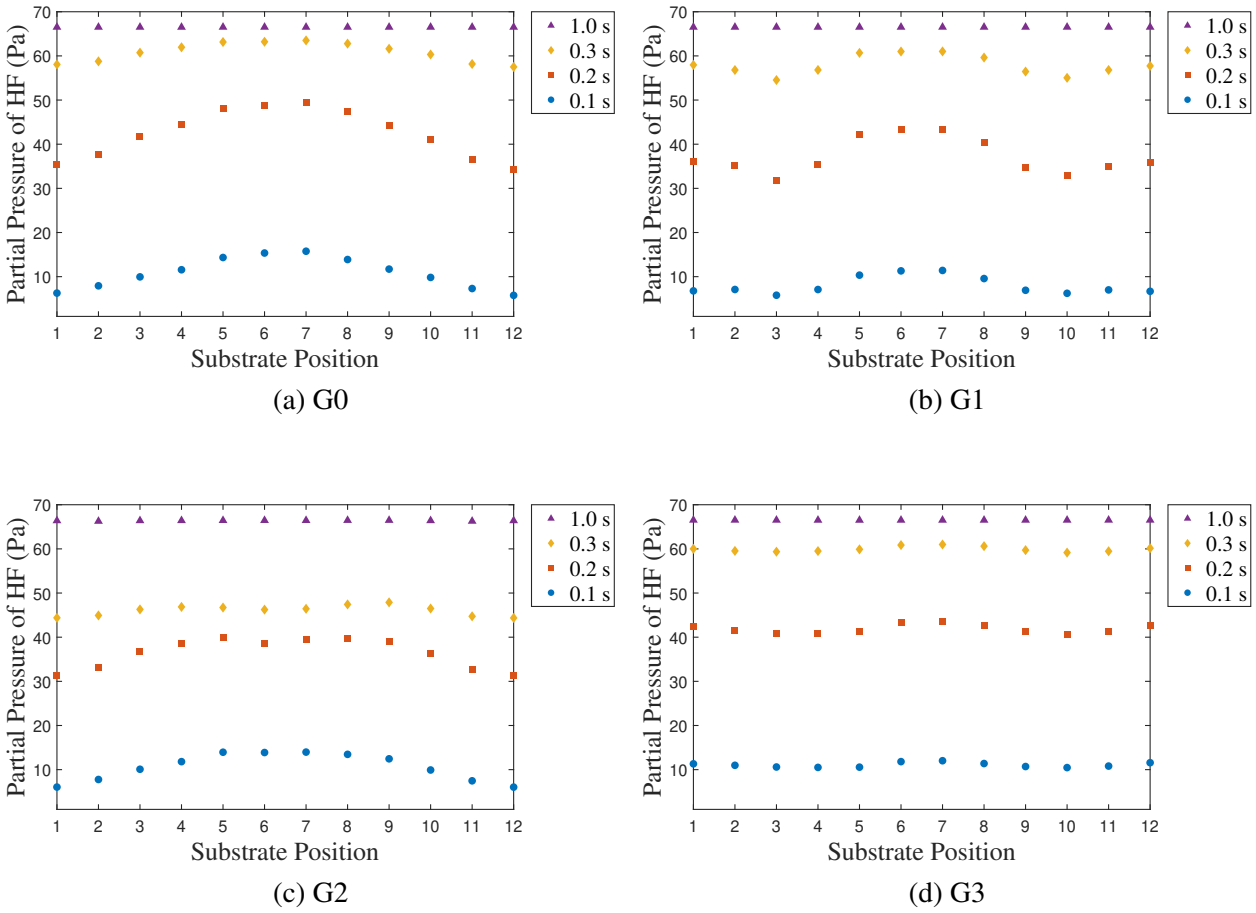


Figure 2.7: Centerline HF pressure data for each reactor at various times for an HF feed flow rate of 150 sccm. The substrate position is numbered starting from the top of the divided wafer in Figure 2.2b to the bottom.

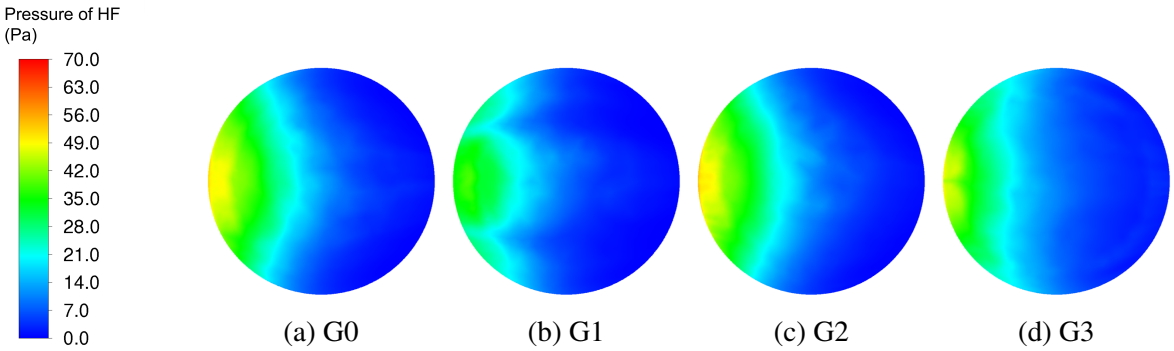


Figure 2.8: Contours of pressure of HF on the surface of the wafer for a Step A process time of 0.1 s and for an HF feed flow rate of 150 sccm.

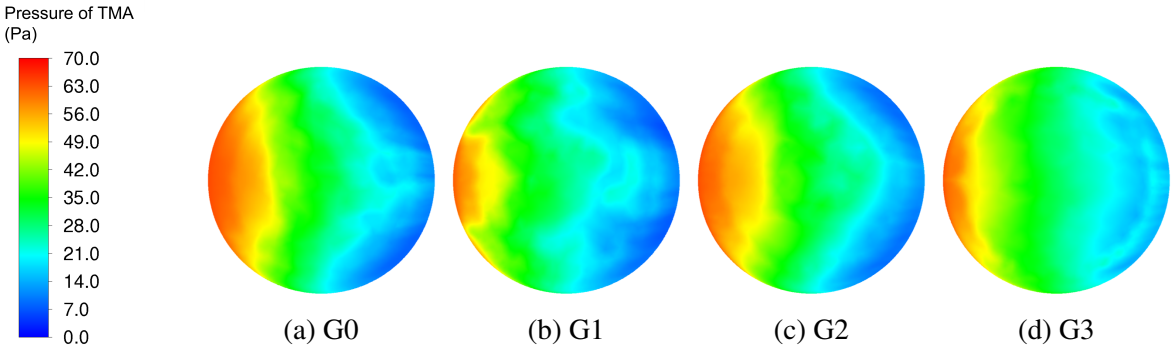


Figure 2.9: Contours of pressure of TMA on the surface of the wafer for a Step B process time of 0.2 s and for a TMA feed flow rate of 70 sccm.

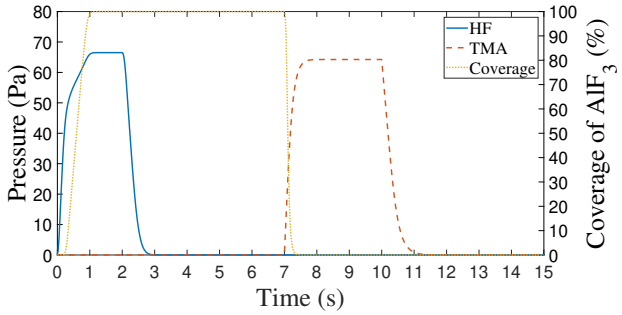
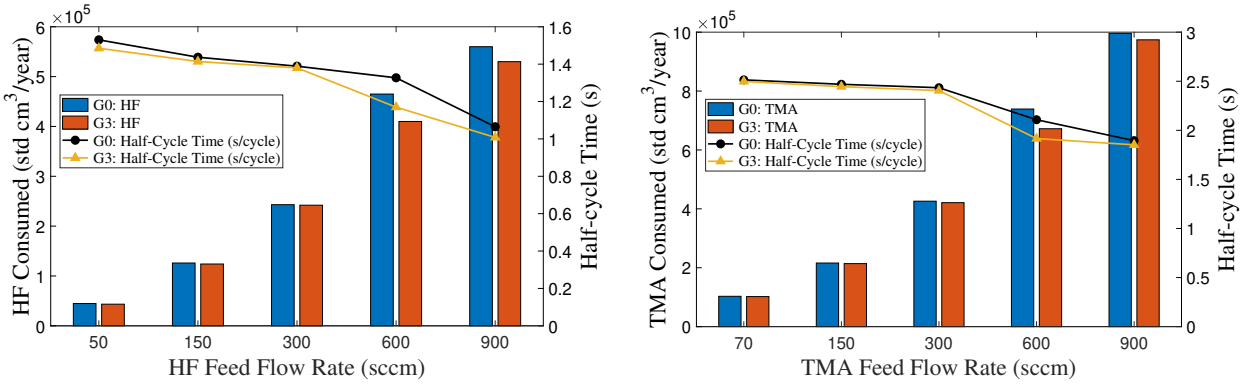


Figure 2.10: Complete cycle of G3 displaying the pressure of HF and TMA and coverage of AlF_3 for an HF and TMA feed flow rate of 150 sccm and 70 sccm, respectively. The blue solid line and the orange dashed line indicate the pressure of HF and TMA over time, respectively. The yellow solid line shows the coverage of AlF_3. The AlF_3 is formed in Step A and etched in Step B.

B are displayed in Figure 2.11b where faster etching results for Step B are observed for G3, thus, lesser TMA is needed to achieve complete etching of a mono-layer of surface substrate. Adopting the G3 model could save at least 6.5×10^2 $std\ cm^3$ and at most 6.7×10^4 $std\ cm^3$ of TMA for the simulated range of feed flow rates. The greatest amount of TMA saved for G3 in comparison to G0 occurs with a flow rate of 600 sccm. Consequently, the benefits of utilizing the inclined plate reactor not only include reduced Modification (Fluorination) and etching times but also a lesser amount of precursors is needed when compared to the typical reactor. Despite being able to maximize the amount of precursor saved for the 600 sccm flow rate, the reduction in half-cycle times for

Steps A and B suggest that increasing the precursor flow rate does not reduce the half-cycle times significantly, and thus, it may be preferable to operate under laminar-like conditions with a lower magnitude of flow rate as discussed by [30].



(a) Step A (b) Step B

Figure 2.11: Process time and consumption of HF and TMA per year comparison between the CFD simulations of the typical reactor (G0) and of the inclined plate reactor (G3). The blue and orange bar indicate the consumption of the precursors for G0 and G3, respectively, for a single reactor. The black and yellow solid lines indicate the half-cycle time of G0 and G3 for both steps, respectively.

2.4 Conclusion

The thermal atomic layer etching (ALE) process of Al_2O_3 was simulated using a multiscale 3D computational fluid dynamics (CFD) model to investigate the impact of gas-phase transport phenomena on the etching process across a wafer. The CFD simulation was performed first by constructing various reactor geometries (typical, multi-inlet, showerhead, and inclined plate) and meshing these geometries was completed until the quality criteria were met using Ansys software. Next, Ansys Fluent was used to perform the CFD simulation with the inclusion of precursor consumption via reactions corresponding to the microscopic model of the etching process. The phonon calculations to obtain thermophysical data were carried out prior to CFD calculations with some materials

requiring the use of computational chemistry software, Quantum Espresso (QE), to calculate the thermophysical data through electronic calculations. Lastly, the process data exported at every time step from Fluent were used in the kinetic Monte Carlo (kMC) microscopic model to determine the half-cycle times for Steps A and B for each reactor geometry. The results of the multiscale CFD model were validated by the experimental results from [33]. It was found that the inclined plate reactor produces a desirable distribution of precursor to the wafer and has faster cycle times for both Steps A and B compared to the other three reactor geometries studied.

Nomenclature

ALD Atomic layer deposition

ALE Atomic layer etching

ASALD Area-selective atomic layer deposition

BDEAS Bis(diethylamino)silane

CT Collision theory

DEA Diethylamine

DFT Density functional theory

EPEs Edge placement errors

FinFETs Fin field-effect transistors

GA Growth area

GUI Graphical user interface

Hacac Acetylacetone

kMC Kinetic Monte Carlo

MC Monte Carlo

NEB Nudged elastic band

NGA Non-growth area

PAW Projector augmented-wave

PWscf Plane-wave self-consistent field

QE Quantum ESPRESSO

SAM Self-assembled monomer

SCF Self-consistent field

SMI Small-molecule inhibitor

TST Transition state theory

VSSM Variable step size method

$\overrightarrow{g}$ Gravitational Acceleration Constant $9.806\,65\ \mathrm{m\,s^{-2}}$

R Universal Gas Constant$8.314\,463\ \mathrm{m^3\,Pa\,K^{-1}\,mol^{-1}}$

ν Kinematic viscosity of the fluid

$\overline{\overline{\tau}}$ Stress tensor

$\overrightarrow{F}$ External body force

ρ Density of the precursor species

$\vec{A}_i$ Area vector of the cell face

$\vec{c}_i$ Vector from the centroid of the cell to the centroid of the adjacent cell

$\vec{v}$ Velocity of the mixture

A_j Pre-exponential factor for reaction, j

E Internal energy

$E_{A,j}$ Activation energy for reaction, j

h_j Sensible enthalpy of the species, j

J_j Diffusion flux of the species, j

k_i Reaction rate constant for reaction, i

k_j Reaction rate constant for reaction, j

k_{total} Sum of reaction rate constants

N Number of reaction pathways

p Static pressure of the species

S_h Heat transfer source

S_m Mass transfer source term

T Operating temperature of the reactor

t Process time of the reaction

u_∞ Free stream velocity of the fluid

x Boundary layer starting distance from the wall

y_p Distance between the wall to the adjacent cell centroid

β_j Temperature exponent for reaction, j

γ Coefficient for reaction selection and time evolution where $\gamma \in (0, 1]$

G0 Typical reactor geometry

G1 Multi-inlet reactor geometry

G2 Showerhead reactor geometry

G3 Inclined plate reactor geometry

$Al(CH_3)_3$ Trimethylaluminum, TMA

Al_2O_3 Aluminum Oxide

$AlF(CH_3)_2$ Dimethylaluminum Fluoride, DMAF

AlF_3 Aluminum Fluoride

H_2O Water

HF Hydrogen Fluoride

Chapter 3

Multivariable Run-to-Run Control of Thermal Atomic Layer Etching of Aluminum Oxide Thin Films

3.1 Introduction

Due to the recent Covid-19 pandemic, there has been a transition from the real world to the digital world or the so-called "metaverse." As telecommuting increases, the demand for a wider range of electronic devices has surged alongside it. This trend also comes with a growing requirement for more robust and reliable computing power, which can only be obtained from high-performance semiconductors. In addition, the autonomous vehicle market is greatly expanding, and more vehicles are integrating additional safety and convenience features that require even more semiconducting materials. The combination of all of these factors has resulted in a growing demand for these valuable, ultra-high-performance semiconductors from various industries during this Fourth

Industrial Revolution. However, the semiconductor industry's fabrication output has not been able to meet this growing demand, and with the uncertainty of the Covid-19 pandemic, shortages are becoming more and more prevalent in the semiconductor market [41]. Thus, a more effective and efficient method for producing these semiconductors must be established to meet the growing consumer demand for these materials. Despite the pressure to continuing miniaturizing semiconductors, the continued shrinking of the fin dimensions of the fin field-effect transistors (FinFETs) has been obstructed by obstacles associated with the 5 *nm* node size [42]. Thus, Gate-all-around (GAA) transistors have become the most promising competitor to replace FinFET technology, as they are able to enter the sub-5 *nm* era.

Atomic layer etching (ALE), considered one of the most advanced etching techniques in semiconductor fabrication, is anticipated to be one of the key processes necessary to overcome this miniaturization issue [43]. ALE is an etching process that uses sequential precursor pulses in between purge steps to achieve self-limiting reaction behavior that etches away the substrate surface monolayers. Due to this behavior, ALE has received great attention as a promising etching process that is able to usher in the sub-5 *nm* era. Many researchers have investigated and demonstrated the technical viability of ALE processes with various materials such as Si [8], SiO_2 [44], MoS_2 [45], Si_3N_4 [46], and Al_2O_3 [40]. In particular, high dielectric materials, which can output a higher computing speed with a lower leakage current [19], have received tremendous attention. Recently, [40] has validated the self-limiting behavior of aluminum oxide in a thermal ALE process. These processes are particularly noticeable since they can ensure high uniformity and conformity of the thin film, thus resulting in ultra-smooth thin films. Nevertheless, the thermal ALE process of aluminum oxide has not been extensively studied, and thus, the process has not been fully characterized. The

thermal ALE of Al_2O_3 has been investigated microscopically [47] and from a multiscale modeling framework [1] that provides a multiscale perspective on the process, which was validated from the experimental results of [40].

Despite the advances made in ALE modeling, an optimal control scheme and guideline to perform process control for the ALE system are still lacking. For example, prior research has been conducted in the feedback control methods for silicon [48]. However, one of the most notable control methods in the semiconductor industry is run-to-run (R2R) control, which has the form of a batch system where a process recipe is modified in between the "batches" according to a predefined relationship between the inputs and outputs. Despite the fact that R2R control has been gradually incorporated into semiconductor fabrication facilities, the lack of quantitative and qualitative process information and nonlinear dynamic behaviors are some of the many reasons why semiconductor manufacturers are skeptical about integrating R2R control into ALE processes. In addition, a single R2R algorithm may not be sufficient to cover all possible disturbances such as process drift, shift, and other kinds of variability [49]. To overcome this issue, a combination of batch process control and feedback control has recently emerged in the semiconductor manufacturing industry [50]. The conjunctive strategy of R2R and feedback control has been studied by [29] and [51]. As another combined strategy, effective multi-algorithms for R2R control have emerged as robust, stable, and optimal control schemes in the semiconductor manufacturing industry [52]. In this research, multiple algorithms for multivariable R2R control are developed to ensure the conformity and uniformity of thin films and to adapt to disturbances that may impede the standard operating conditions in the context of thermal ALE of aluminum oxide thin films.

3.2 Multiscale CFD Modeling of Thermal ALE

The thermal atomic layer etching (ALE) for aluminum oxide (Al_2O_3) thin films can be simulated by multiscale computational fluid dynamics (CFD) modeling, which is an integrated form of microscopic and macroscopic modeling. This multiscale model is conducted similarly to prior research by [53] and [54]. Based on the kinetic Monte Carlo (kMC) method, the microscopic modeling composes of a general simulation of the kinetic components of the reaction under ideal conditions and is simulated through randomness to generate a kinetic model of the surface. For the gas transport domain, the previously developed inclined plate reactor is used and CFD simulations are performed using Ansys Fluent 2021R2, which is a widely used commercial CFD software. For the surface domain, the process data from the CFD model is exported and transferred to the microscopic model to calculate the coverage or etching fraction in the atomistic point of view.

In the previous research [1], the multiscale CFD simulation included the influence of transport phenomena effects to various reactor models, which were examined for their effectiveness in delivering a uniform precursor flow pattern to the substrate surface and for the time to achieve complete coverage and etching. It was revealed that the inclined plate reactor performed optimally when comparing the types of reactors that were studied. Therefore, in this chapter, the multiscale CFD model of the inclined plate reactor design is adopted to develop and optimize a Multivariable R2R control system.

3.2.1 Microscopic Surface Modeling

The thermal atomic layer etching (ALE) of Al_2O_3 uses two precursor pulses in a sequential manner. Thus, there are two reaction steps in between the purge steps: Step A and Step B. In Step A, the wafer is exposed to hydrogen fluoride (HF), which fluorinates the aluminum oxide on the surface to produce an AlF_3 layer. Next, a purge step sweeps the remaining HF in the reactor to avoid undesired reactions and to guarantee self-limiting behavior. Then, the fluorinated surface (AlF_3) undergoes a ligand-exchange and is converted into a volatile species, dimethylaluminum fluoride [DMAF, $AlF(CH_3)_2$], when trimethylaluminum (TMA), $Al(CH_3)_3$, is supplied during Step B. The volatile layer, DMAF, is desorbed and thus, a monolayer of the substrate surface is removed. Lastly, another purge step removes the remaining TMA and residual products. A cyclical operation consists of the aforementioned steps, which is repeated until the desired film thickness is achieved. The overall reaction is described by

$$Al_2O_3\ (s) + 6HF\ (g) + 4Al(CH_3)_3\ (g) \rightarrow 6AlF(CH_3)_2\ (g) + 3H_2O\ (g)$$

A microscopic surface model for etching of Al_2O_3 thin films was described in the previous work [47]. A brief description of the microscopic model is presented here. θ-Al_2O_3 ($\bar{2}$ 0 1) was found on Si(1 0 0) through the atomic layer deposition (ALD) process under annealing [55]. Thus, θ-Al_2O_3 ($\bar{2}$ 0 1) is used as the preferred lattice structure. A 300 $\times$ 300 lattice is applied to the microscopic model in which etch reactions take place. The surface kinetics in the atomistic level is modeled using the kinetic Monte-Carlo (kMC) method, in particular the variable step size method (VSSM) that is often called the *n*-fold way or BKL, which refers to the algorithm developed by

Bortz, Kalos, and Lebowitz [56]. The kMC algorithm was popularized by [57] who integrated the Monte Carlo algorithm into chemical kinetics. The kMC method is widely used to simulate individual reactions on the microscopic scale including the dependence on the lattice structure [58]. In the kMC algorithm, the total rate constant, k_{total}, is an important parameter that selects a reaction on the reaction site and calculates the time progression, which is computed as follows:

$$k_{\text{total}} = \sum_{i=1}^{N} k_i \tag{3.1}$$

where k_i is the reaction rate constant of the reaction i, and N is the number of reactions. After all reactions are defined and the total reaction rate constant is calculated, a random number, $\gamma_1 \in (0, 1]$, is chosen to determine the reaction on the site by using the criterion as follows:

$$\sum_{i=1}^{j-1} k_i \leq \gamma_1 k_{\text{total}} \leq \sum_{i=1}^{j} k_i \tag{3.2}$$

where j represents the reaction j. Finally, an additional random number, $\gamma_2 \in (0, 1]$, is generated to compute the time interval defined as:

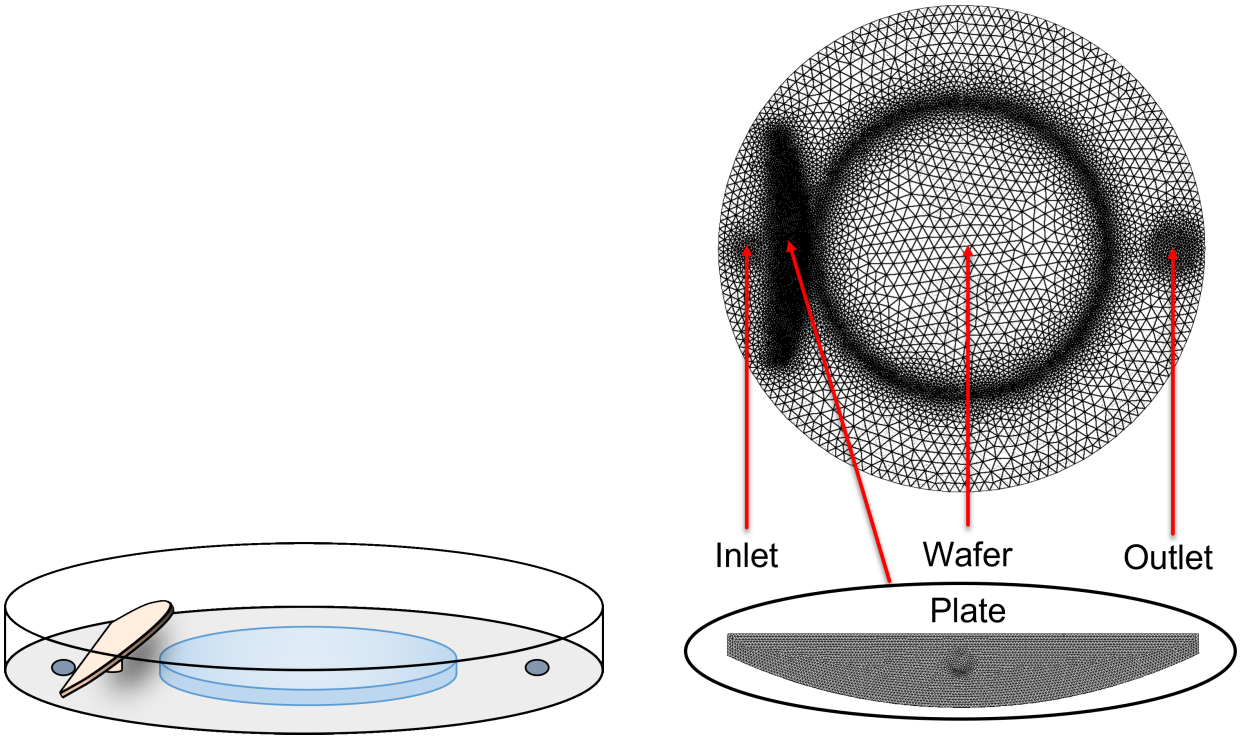
$$\Delta t = \frac{-\ln(\gamma_2)}{k_{\text{total}}} \tag{3.3}$$

Detailed kinetic mechanisms for the fluorination and ligand-exchange reaction as well as their kinetic parameters, which were calculated using electronic structure optimization methods and Density Functional Theory, can be found in the research of [47].

3.2.2 Macroscopic Modeling

In the previous work, as shown in Figure 3.1a, a 3D computational fluid dynamics (CFD) model for the inclined plate reactor design was developed and evaluated by [1]. The optimized inclined plate reactor showed the uniform precursor distribution while preserving the fastest cycle time among other geometries. Thus, in this chapter, the inclined plate reactor is adopted to formulate the control scheme. The reactor configuration consists of a cylindrical-shaped chamber with a 500 mm diameter and 10 mm height, a round-shaped inlet of 20 mm diameter, and a round-shaped outlet of 40 mm diameter. A substrate of 300 mm is placed at the center of the reactor. To optimize the precursor distribution, an arch-shaped inclined plate with 2 mm thickness and 5° angle from the horizontal is equipped near the inlet. Meshing Mode, a feature of Ansys Fluent 2021R2, was utilized to generate the mesh for the plate reactor. Tetrahedral cells were adopted to reduce simulation time while maintaining the accuracy of the numerical calculations. Mesh quality criteria ranges recommended from [34] including orthogonality, skewness, aspect ratio, and resolution were standards that were integrated into the development of the plate mesh, which is illustrated in Figure 3.1. The characteristics of the plate mesh as well as the quality criteria calculated from Ansys Fluent 2021R2 are discussed in greater detail by [1].

Ansys Fluent 2021R2 is utilized to conduct numerical computational fluid dynamics (CFD) calculations of the fluid flow and to simulate the surface kinetics of HF and TMA half-cycle in the ALE process. The pressure-based solver under transient mode in Ansys Fluent is used with time step of 0.025 s and 200 iterations per time step. The coupled algorithm is used to decrease the computation time. The pressure-based solver in Ansys Fluent solves mass and momentum



(a) (b)

Figure 3.1: A schematic diagram of the inclined plate reactor (a) and the inclined plate reactor mesh generated from Ansys Fluent (b) are illustrated from [1].

conservation equations, which are expressed as follows:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{v}) = S_m \tag{3.4}$$

$$\frac{\partial (\rho \overrightarrow{v})}{\partial t} + \nabla \cdot (\rho \overrightarrow{v} \overrightarrow{v}) = -\nabla p + \nabla \cdot (\overline{\overline{\tau}}) + \rho \overrightarrow{g} + \overrightarrow{F} \tag{3.5}$$

where ρ is the density of the mixture, $\overrightarrow{v}$ is the velocity of the mixture, S_m is the mass transfer source term, p is the static pressure, $\overline{\overline{\tau}}$ is a symmetric rank two stress tensor, $\rho \overrightarrow{g}$ is the gravitational body force, and $\overrightarrow{F}$ is the external body force. In addition, Ansys Fluent solves the conservation of energy

equation which is described by:

$$\frac{\partial}{\partial t}(\rho E) + \nabla(\vec{v}(\rho E + p)) = -\nabla(\Sigma h_j J_j) + S_h \tag{3.6}$$

where E is the internal energy, h_j is the sensible enthalpy of species j, J_j is the diffusion flux of species j, and S_h is the heat transfer source term. To ensure more accurate results, in the pressure-based coupled solver, the second-order upwind scheme is used for spatial discretization, leading to higher-order accuracy through a Taylor series expansion. It is also required to calculate gradients to compute the values of scalars at the cells on the mesh. To calculate the gradients, the Incomplete Lower Upper (ILU) method is employed, which requires more computing power but offers better smoothing properties for the pressure-based coupled solver. Based on these solver settings, aforementioned equations are solved iteratively at a given time step. Once the solution satisfies the convergence criteria, the solver advances to the next time step.

To generate a more realistic profile, the consumption of precursors and generation of products are included in the CFD model by determining the reaction rate from the modified Arrhenius equation that includes the temperature dependence of the pre-exponential factor, which is defined as follows:

$$k_j = A_j T^{\beta_j} e^{-E_{A,j}/RT} \tag{3.7}$$

In the above equation, k_j is the reaction rate constant for reaction j, T is the temperature on the surface, β_j is the temperature exponent for reaction j, $E_{A,j}$ is the activation energy for reaction j, and R is the ideal gas constant. Both half-cycle reactions are developed in Ansys Fluent, which are

described below:

$$Al_2O_3\ (s) + 6HF\ (g) \rightarrow 2AlF_3\ (s) + 3H_2O\ (g)$$

$$2AlF_3\ (s) + 4Al(CH_3)_3\ (g) \rightarrow 6AlF(CH_3)_2\ (g)$$

The monitoring of the surface temperature in real time is necessary to ensure that the temperature does not decrease, especially for Step A, which requires higher operating pressures for HF at lower temperatures [47]. Typically, the surface temperature is maintained to guarantee film quality, and thus, a PI (proportional-integral) controller is assumed to work appropriately in this process. In addition, the cyclical operation is carried out through a user-defined function (UDF) in which operating conditions and boundary conditions are specified. The detailed description for the macroscopic modeling can be found in the previous work of [1].

3.3 Multivariable R2R Control Formulation

Various algorithms have been employed in existing run-to-run (R2R) control algorithms for chemical vapor deposition (CVD) and chemical mechanical polishing (CMP) processes including the exponentially weighted moving average (EWMA), predictor-corrector control (PCC), and optimizing adaptive quadratic controller (OAQC) methods. For example, [29] utilized the EWMA and PCC methods on PEALD of HfO_2 thin films to compare their effectiveness and [54] developed an R2R controller using the EWMA algorithm for thin film Si-H solar cells. The EWMA controller, which is a linear approximation model-based controller, is widely utilized and integrated into semiconductor fabrication processes due to its versatility to compensate for process shift, drift, and noise among other disturbances [59]. Run-to-Run controllers are adjusted through multiple

batches by using statistical process control (SPC) and by tuning the controllers using engineering process control (EPC) to provide a new recipe (i.e., input) for the next batch run [60].

In this chapter, an EWMA-based multivariable R2R control scheme is developed using two inputs (the process time and the precursor flow rate) and two outputs (the coverage or etching fraction and the precursor partial pressure). In order to tune an EWMA-based R2R controller, the input-output relationship must be established first. A multi-input-multi-output (MIMO) model may be used to formulate the multivariable R2R controller. However, in this chapter, the process is locally approximated by two single-input-single-output (SISO) linear regression modelss, with each one of them being defined by an equation of the following form:

$$y_t = \alpha + \beta u_{t-1} \tag{3.8}$$

where y_t represents the output of the process for batch run t, α is the bias, β is the process gain, and u_{t-1} is defined as the recipe (or input variable) in between batch run $t-1$ and batch run t. The process parameters, α and β, are determined using the standard least squares method where α is the y-intercept and β is the slope of the linear regression model. R2R controllers are modeled under an evolutionary operation mode [61] that combines statistical results to improve efficiency and increase productivity. In other words, the control system works to adapt to changes in the process environment and tune the manipulated variables to sustain standard output conditions from the R2R controller. The first step of the control work is to update the model, which is expressed below:

$$a_t = \lambda \left(y_t - b u_{t-1} \right) + (1 - \lambda)\, a_{t-1} \tag{3.9}$$

where λ represents the weight factor that is responsible for the translation of the regression model, a_t is the updated bias, and b is the process gain that is the same as β in Eq. (3.8). Typically, λ is chosen to be $0.1 \sim 0.3$ [50]. After the model is updated, the new recipe for the next batch run is computed by the following expression:

$$u_t = \frac{T - a_t}{\beta} \tag{3.10}$$

where u_t is the new recipe for the next batch run and T represents the target or desired value of the output variable.

The stability and robustness of the EWMA-based R2R controllers have been studied; however, it has been reported that the EWMA and PCC (also known as double-EWMA) methods underperform for nonlinear systems due to the linear model-based R2R algorithm [49, 52]. Nevertheless, the EWMA method is able to be used for nonlinear systems if linear regression models are well-developed for fitting nonlinear responses, which is discussed in this chapter. In this chapter, two different R2R controllers are formulated and integrated, resulting in the Multivariable R2R control system in which each controller is modeled using a single-input-single-output (SISO) regression model as shown in Figure 3.2. The first R2R (denoted as R2R-1) tracks the reaction progression (reflected by the coverage or etching fractions) to adjust the process time. Typically, a quartz crystal microbalance (QCM) is used to monitor the etching rate through mass changes along the surface of the thin film substrate [33] in real time, which allows the etching and coverage fractions to be computed. The second R2R (denoted as R2R-2) manipulates the flow rate of the precursors by monitoring the precursor partial pressure deviation from the desired partial pressure at the standard

operating conditions for the developed inclined plate reactor outlined in Table 3.1. The following sections discuss the tuning methodology of R2R-1 and R2R-2 in greater detail.

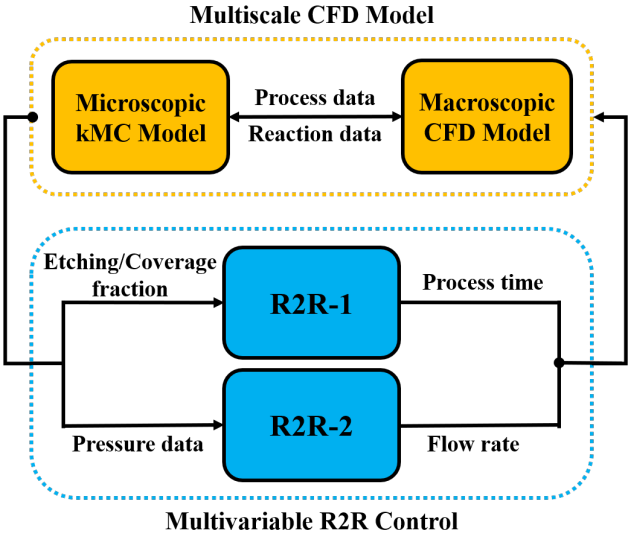


Figure 3.2: Multivariable run-to-run control of the inclined plate reactor.

Table 3.1: Standard operating conditions for the multiscale CFD simulation.

Standard Condition	Step A	Step B
Precursor Flow Rate (sccm)	150	70
Operating Pressure (Pa)	133	133
Temperature (K)	573	573
N_2 Flow Rate (sccm)	150	150
Process Time (s)	1.1	2.0

3.3.1 R2R-1 Modeling and Tuning

The schematic process diagram shown in Figure 3.3 illustrates the Multivariable R2R control system. R2R-1 is implemented to adjust the valve opening time (i.e, process time) for precursor injec-

tion to the reactor to achieve complete AlF_3 coverage and etching for Steps A and B respectively. Multiscale computational fluid dynamics (CFD) simulation data are collected at standard operating conditions for Steps A and B for the inclined plate reactor, which are summarized in Table 3.1. Process or valve opening times of 1.1 s and 2.0 s in Table 3.1 reflect the ideal time for the Al_2O_3 thin film substrate to reach complete coverage or etching respectively under the standard operating conditions. In this chapter, it is assumed that the time until the valve fully opens is negligible, and thus, the dynamics of the valve do not interfere with the process dynamics. In other words, R2R-1 only adjusts the process time; however, the flow rate is only adjusted by the upstream valve as shown in Figure 3.3, which is controlled by R2R-2. The R2R-1 controller is approximated to a single-input-single-output (SISO) model in which the process time (t) for the precursor injection serves as the recipe ($u = t$ in Eqs. 3.12 and 3.13) and the etch or coverage fraction (f) of AlF_3 serves as the output variable ($y = f$ in Eqs. 3.12 and 3.13). It is of paramount importance to determine a reasonable solution to the SISO regression model with low variance along the regression line since the solution has a great impact on the control work [52]. In this chapter, multiple regression models for R2R-1 are developed and evaluated using the standard least squares method between the result data and the solution of the SISO model. First, the standard linearization of the multiscale CFD simulation results for Steps A and B are presented in Figure 3.4. Figure 3.4 reveals that the linear regression model does not fit the simulation data for Steps A and B due to their nonlinear process responses and the larger variance of the data from the regression line, thus the linear regression of the entire data set may be difficult to accomplish. To improve the least squares values and thereby strengthen the fit of the regression model to the multiscale CFD results, several techniques are used to achieve a better fit of the model data: piecewise and modified median-effect. For the piecewise

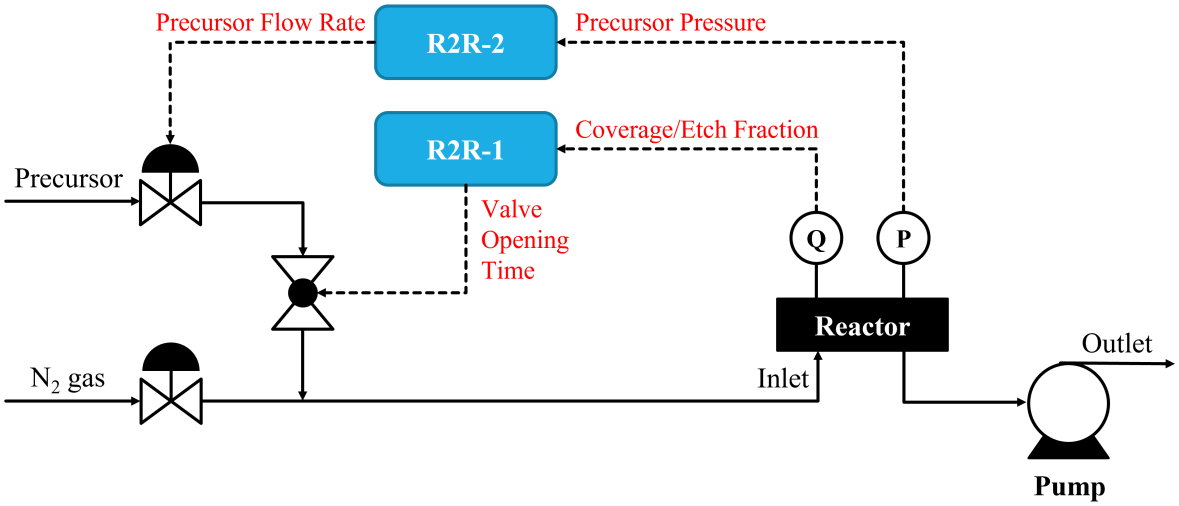


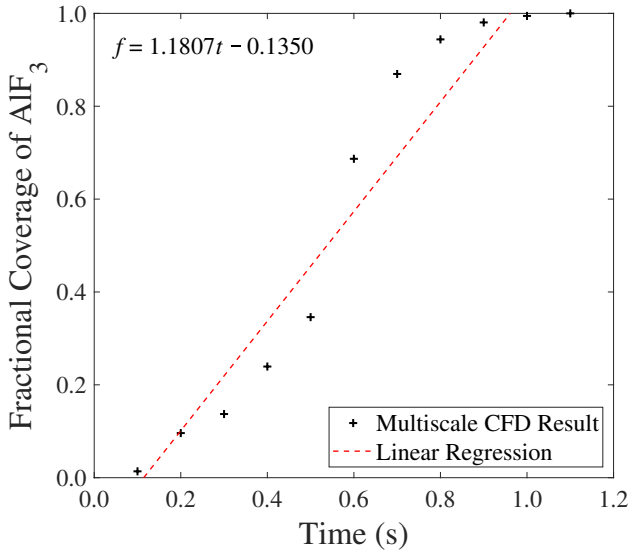
Figure 3.3: The Multivariable Run-to-Run control system.

regression model, the data curve is divided into two groups to generate two linear piecewise plots that have an intersection point (t_p, f_p) at a time of, t_p, which is expressed as follows:

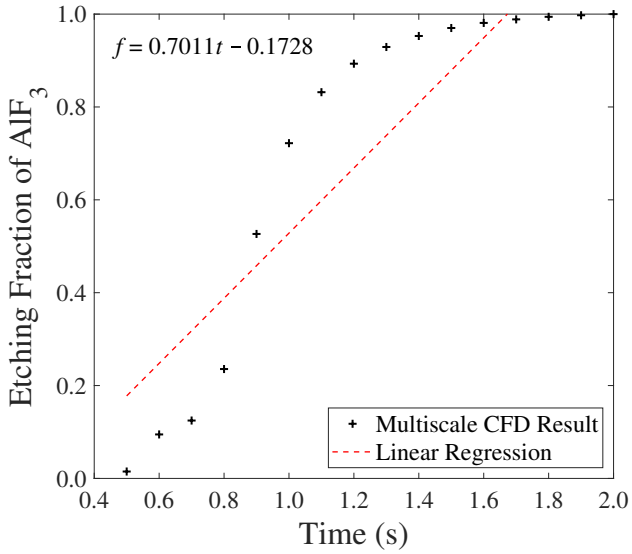
$$f_p = \frac{\alpha_2 - \alpha_1}{\beta_1 - \beta_2} \tag{3.11}$$

where α_1 and α_2 are the biases for the two plots, respectively, and β_1 and β_2 are the process gains, respectively. A conditional loop is necessary to ensure the correct piecewise regression model to use, which is based on the intersection point, f_p. The piecewise regression models are presented in Figure 3.5 and displays a marginal improvement of linearity for both precursor injection steps.

As another alternative technique to improve the curve fitting of the multiscale CFD results, the median-effect equation is adopted from prior research [62] and is applied to the formulation for the coverage and etching fraction progression with time. The median-effect equation is suitable for the EWMA method because of its logistic-like function behavior, which exemplifies the multiscale

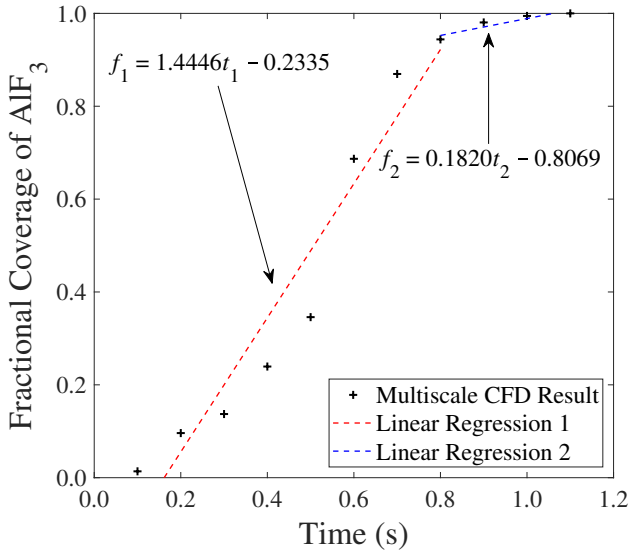


(a)

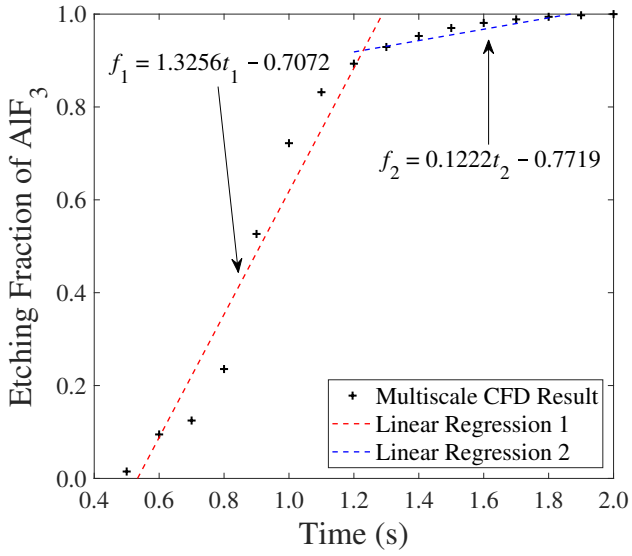


(b)

Figure 3.4: The input-output relationship between the fractional coverage of AlF_3 and the precursor valve opening time (a) for Step A and the etching fraction of AlF_3 and the precursor valve opening time (b) for Step B, which is derived from a standard linear regression model for the EWMA-based R2R controller, R2R-1. For Steps A and B, the R^2 values from Table 3.2 indicate marginal linear behavior for Step A and a lack of linear behavior for Step B.



(a)



(b)

Figure 3.5: The input-output relationship between the fractional coverage of AlF_3 and the precursor valve opening time (a) for Step A and the etching fraction of AlF_3 and the precursor valve opening time (b) for Step B, which is derived from a standard linear regression model divided into linear piecewise functions for the EWMA-based R2R controller, R2R-1. The R^2 values from Table 3.2 indicate a marginal to moderate linear relationship of multiscale CFD data.

CFD data trend and its ability to transform both the coverage (or etching) fraction and the process time into logarithmic forms. The median-effect method can also be modified into a linearized function in the form of Eq. (3.8), which will be discussed later in this section. The median-effect equation is defined as follows:

$$\frac{f}{1-f} = \left(\frac{t}{\eta}\right)^{\delta} \tag{3.12}$$

where f is the coverage or etching fraction, t is the process time, and the constants, η and δ, are obtained by regressing the median-effect equation into a linearized form to obtain the slope and y-intercept. It is also notable that f can be solved algebraically, which demonstrates the practicality of integrating the median-effect equation into the EWMA model. In this research, additional tuning constants (γ and ϵ) are introduced to reduce the variance of the data to generate a "modified" median-effect equation, which is defined below.

$$\frac{\gamma f}{1-\gamma f} = \left(\frac{t+\epsilon}{\eta}\right)^{\delta} \tag{3.13}$$

In particular, γ and ϵ are adjustable parameters for modifying the median-effect equation in Eq. (3.12) and are determined by tuning the factors until a desirable R^2 value of the linear regression is obtained. The original median-effect equation from [62] is obtained by declaring $\gamma = 1$ and $\epsilon = 0$ in Eq. (3.13). The parameter, γ, is bounded such that $\gamma \in (0, 1]$, and is employed to shift the upper horizontal asymptote of the median-effect regression while ϵ is utilized for translating the regression line along the ordinate direction. The linearized form of the modified median-effect equation

exemplified by [63] is derived below:

$$\ln\left(\frac{\gamma f}{1-\gamma f}\right) = \delta \ln(t+\epsilon) - \delta \ln(\eta) \tag{3.14}$$

Plotting the left-hand side of Eq. (3.14) as a function of $\ln(t+\epsilon)$ can generate a linearized plot to determine the values of η and δ through linear regression where $\beta = \delta$ and $\alpha = -\delta \ln(\eta)$ from Eq. (3.8). The regression models of the modified median-effect and median-effect equations for Steps A and B are presented in Figure 3.6. For Step A, values of $\gamma = 0.99$ and $\epsilon = 0.35$ are used and for Step B, values of $\gamma = 1.0$ and $\epsilon = 0$ are used, and hence, Step B is represented by the original median-effect model. The linear modified median-effect regression model in Figure 3.6a shows that the transformed multiscale CFD dataset exhibits a more linear relationship compared to that of the linear and piecewise regression models. The regression model also correlates the data from the multiscale CFD simulation reasonably, which is observed in Figure 3.6b. The modified median-effect is not required for Step B since the median-effect (i.e., $\gamma = 1$ and $\eta = 0$) has an R^2 value close to unity and the multiscale CFD results have low variance. The results from the median-effect regression model for Step B are displayed in Figures 3.6c and 3.6d. The transformation of the multiscale CFD data improves the linearization of both the linear (Figure 3.4b) and piecewise (Figure 3.5b) regression models, which is validated by an R^2 value of 0.9854 and exhibits a strong least squares fit. In the simulations, a given output variable is logarithmized to compute a logarithmized recipe for the next batch run, while a logarithmized recipe is calculated from Eq. (3.14) and the antilogarithmic form is used to simulate the process.

In conclusion, the SISO regression models of R2R-1 controllers for both half-cycles are ob-

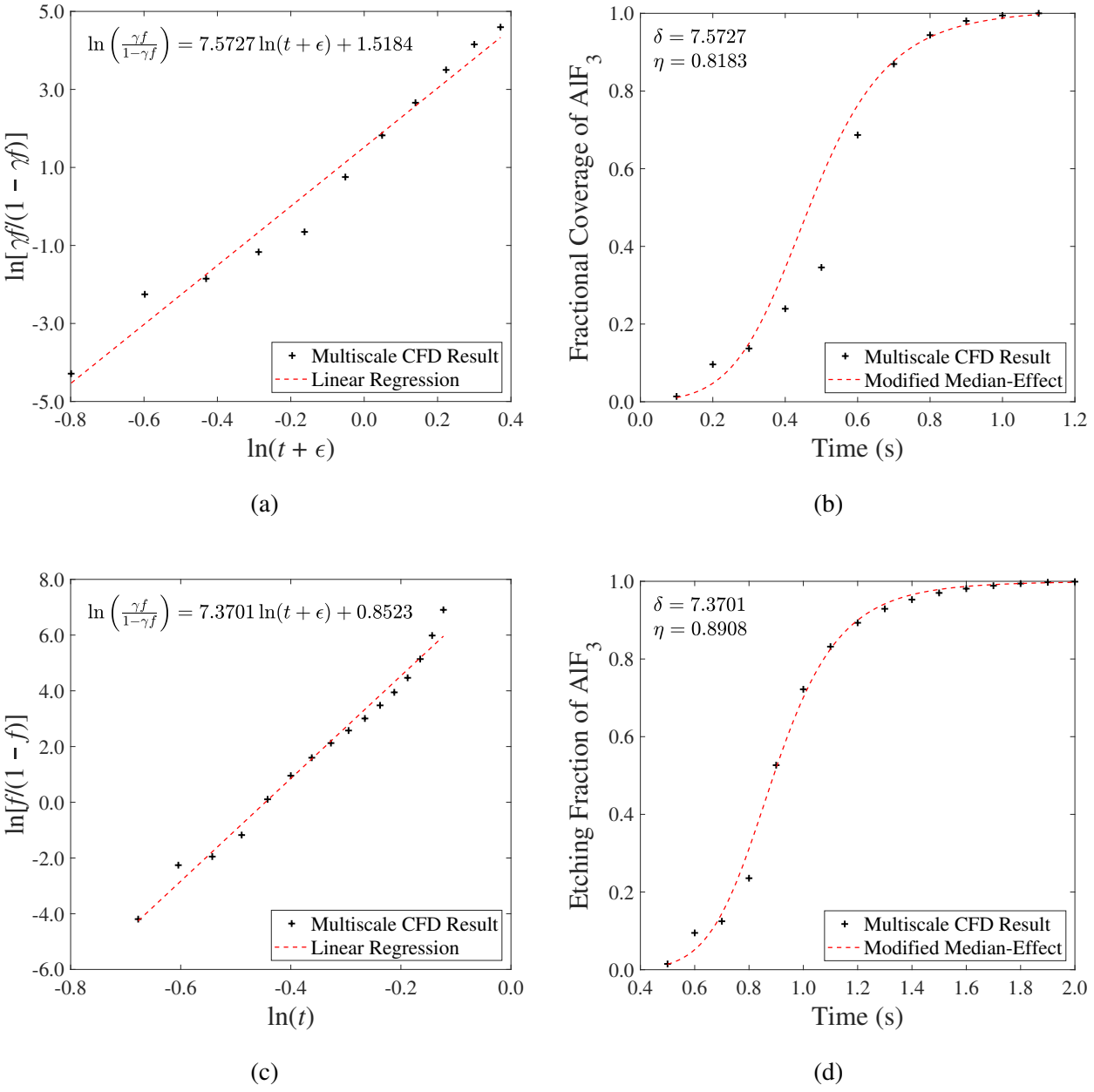


Figure 3.6: The input-output relationship with the logarithmized multiscale CFD data and logarithmized time from the modified median-effect equation for Step A (a) with $\gamma = 0.99$ and $\epsilon = 0.35$ and for Step B (c) with $\gamma = 1.00$ and $\epsilon = 0$ for the EWMA-based R2R controller, R2R-1. The R^2 values in Table 3.2 indicate a strong linear relationship for Steps A and B. The multiscale CFD results with the standard modified median-effect equation are presented in (b) and (d) for Steps A and B, respectively.

tained using three different regression methods and the parameters are presented in Table 3.2. As a result, the modified median-effect method provides better regression models for the SISO regression and overcomes the nonlinear relationship of the multiscale CFD data. The performances of the three models are evaluated in Section 3.4.

Table 3.2: A comparison of the standard linear, piecewise, and modified median-effect regression model parameters that are calculated from the standard least squares method for Steps A and B.

R2R	Half-Cycle	Regression Model	R^2	Process gain	Bias
R2R-1	Step A	Linear Model	0.9251	1.1807	−0.1350
		Piecewise-1	0.9366	1.4446	−0.2335
		Piecewise-2	0.8688	0.1820	0.8069
		Modified Median-Effect	0.9740	7.5727	1.5184
	Step B	Linear Model	0.8019	0.7011	−0.1728
		Piecewise-1	0.9520	1.3256	−0.7072
		Piecewise-2	0.8531	0.1222	0.7719
		Modified Median-Effect	0.9854	7.3701	0.8523
R2R-2	Step A	Linear Model	0.9651	0.8286	−145.6438
	Step B	Linear Model	0.9398	1.2388	−74.8118

3.3.2 R2R-2 Modeling and Tuning

R2R-2 is developed to minimize the precursor partial pressure deviation from the ideal partial pressure at the standard operating condition within the reactor by adjusting the precursor flow rate to the reactor. The R2R-2 controllers for both half-cycles are also approximated to SISO models with the precursor flow rate, V, serving as the recipe ($u = V$), and the sum of the partial pressure deviation

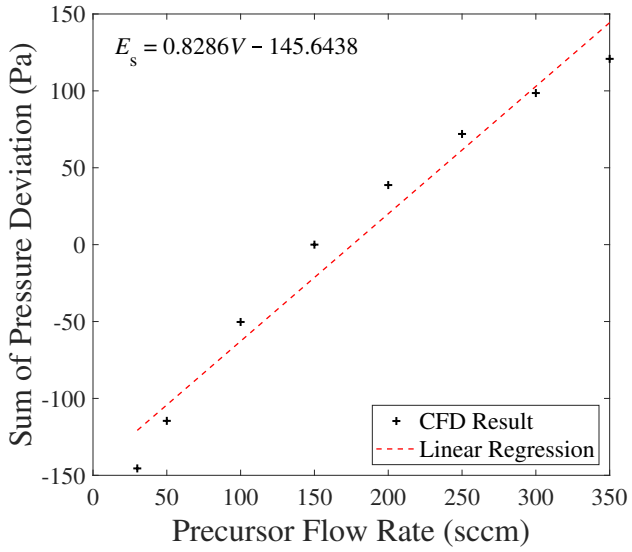
at various times, E_s, is defined as the output variable ($y = E_s$). The controlled variable for R2R-2 is expressed as the following:

$$E_s = \sum_{i=1}^{4} C_i \left[P_m(t_i) - P_d(t_i) \right] \tag{3.15}$$

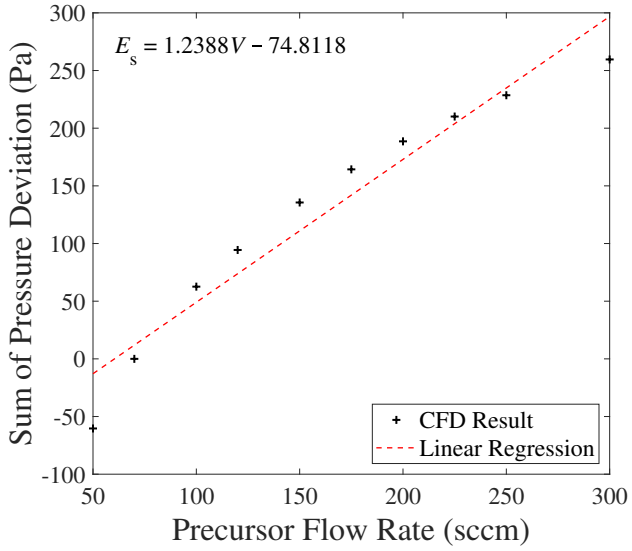
where $P_m(t_i)$ refers to the measured partial pressure at time t_i for four times of 0.1 s, 0.2 s, 0.4 s, and 0.6 s, and $P_d(t_i)$ refers to the desired partial pressure obtained from the standard operating pressure condition that is outlined in Table 3.1 without any disturbances. C_i represents a coefficient that is selectively chosen to improve the linearity of the recorded pressure deviation at the four times. A pressure measuring device such as a pressure sensor can be implemented to monitor the pressure within the reactor at various times to ensure that the sum of the partial pressure deviations is reduced to prevent further loss of control of the system and degradation of the substrate materials. Therefore, R2R-2 can effectively function with the sum of pressure deviations as the output parameter, y_t, and as the precursor (HF and TMA) flow rates serving as the recipe, u_t. As illustrated in Figure 3.7a and 3.7b, the SISO regression models fit the multiscale CFD data well using the coefficients of 1.0 for all times for Step A and 1.0 for all times for Step B except for 2.0 for the first timestep ($C_1 = 2.0$) for Step B. The solutions of the SISO model have R_2 values of 0.9651 and 0.9398 for Steps A and B, respectively.

3.4 Simulation Results and Discussion

Semiconductor manufacturing processes can be subject to some unmeasurable drifts or shifts from wall deposition and equipment aging in the semiconductor industry. In this chapter, two different



(a)



(b)

Figure 3.7: The input-output relationship between the sum of partial pressure deviations and the flow rate for Step A (a) and Step B (b), which is derived from a standard linear regression model for R2R-2. The R^2 values from Table 3.2 indicate a moderate linear relationship.

disturbances are simulated to evaluate the performance of the multivariable R2R control system.

The first disturbance is a shift (referred to as a "kinetic disturbance"), which is able to be driven

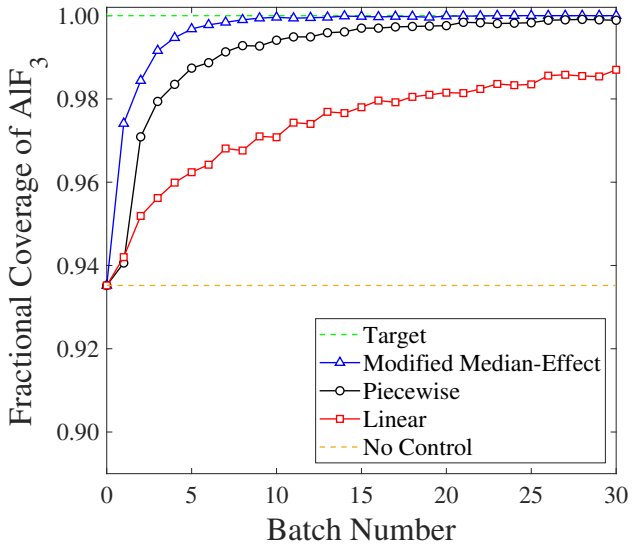
by cyclical operations, machine maintenance or changes in process settings [52]. It can be implemented simply by multiplying the reaction constants by a factor that is less than 1. The other disturbance (referred to as a "pressure disturbance") is simulated by reducing the operating pressure, which can be caused by a malfunction of the vacuum pump. In particular, R2R-2 is designed to deal with any pressure deviation from the standard conditions. A threshold value of 0.999 is set as the target for the fractional coverage for Step A and of the etching fraction for Step B. The multiscale computational fluid dynamics (CFD) simulation is performed by using 24 parallel computing processors with 384 GB memory on a compute cluster. Simulation time for a half-cycle takes half an hour on average. The batch-to-batch calculation of the control action is not demanding compared to the multiscale CFD simulation, and thus, it is considered negligible.

Initially, the R2R-1 controllers for both half-cycles are simulated under a kinetic disturbance using the multiscale CFD model to evaluate the regression models discussed in Section 3.3.1. After a regression model with a better control performance is selected, the Multivariable R2R control system is simulated, evaluated, and then compared with the single R2R control system (R2R-1 and R2R-2) under the two disturbances. As shown in Figure 3.8, the modified median-effect regression model outperforms the linear and piecewise regression models under a kinetic disturbance. The kinetic disturbance is modeled such that the reaction rate constants are multiplied by a shift factor of 0.7 in the kMC model when process time is updated, leading to a reduction in reaction rates. The median-effect regression models reach the target for Step A and Step B at batch runs of 8 and 9 respectively. However, the other two models do not reach the target value until a batch run of 30. One of the major concerns of process control is the possibility that the controllers may dramatically overshoot the target threshold. However, Figure 3.8 reveals that the modified median-effect models

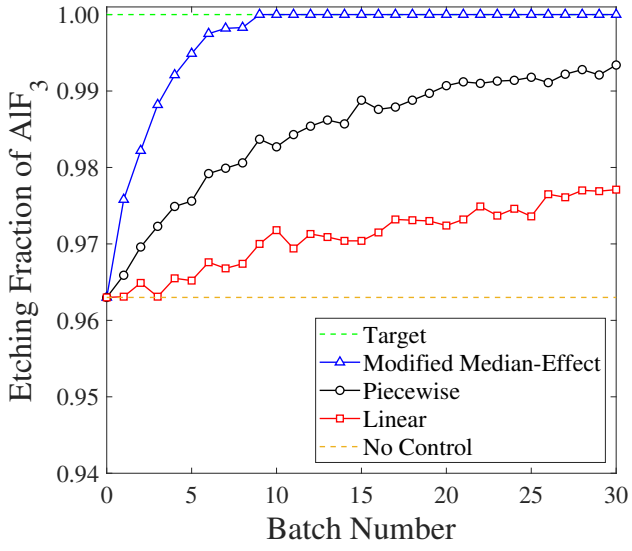
not only reach the target in a reduced number of batch runs, but also when they approach the target, the controllers add sufficiently small increments to the new recipe to prevent the possibility of overshooting the target. In other words, controller response overshoots and rapid control actions are not allowed in the process due to the best curve fit, and thus, the median-effect regression model is selected for the R2R-1 controllers.

The results of the Multivariable R2R and individual R2R control systems under the two shift disturbances for Steps A and B, respectively, are presented in Figure 3.9. To model the kinetic disturbance, the factors of 0.55 and 0.7 are multiplied to the reaction rate constants for Step A and Step B, respectively, and the pressure disturbance is simulated by reducing the operating pressure of the reactor to 40 Pa from 133 Pa. For Step A, both individual R2R configurations reveal that the effects of the disturbances are not able to be removed within 15 batch runs as shown in Figure 3.9a. For Step B, as illustrated in Figure 3.9b, R2R-1 is able to mitigate the impact of the disturbances within 7 batch runs. However, R2R-2 is not able to eliminate the effects of the disturbances within 15 batch runs. In contrast to individual R2R controllers, the multivariable R2R controller is able to successfully mitigate the effect of the disturbances within batch number 5 for Step A in Figure 3.9a and within 3 batch runs for Step B in Figure 3.9b. As a result, the multivariable R2R configuration of R2R-1 and R2R-2 performs more efficiently since the lesser number of batch runs is required to approach the complete coverage or etching, which is represented by the target line.

The multivariable R2R control algorithm is also efficient in regard to adjusting the input variables: process time and precursor flow rate. Figure 3.10 shows the corresponding input values from the control work in Figure 3.9. For both half-cycles, R2R-1 continuously increases the process time to compensate for the effect of the disturbances, as can be seen in Figures 3.10a and 3.10c. How-



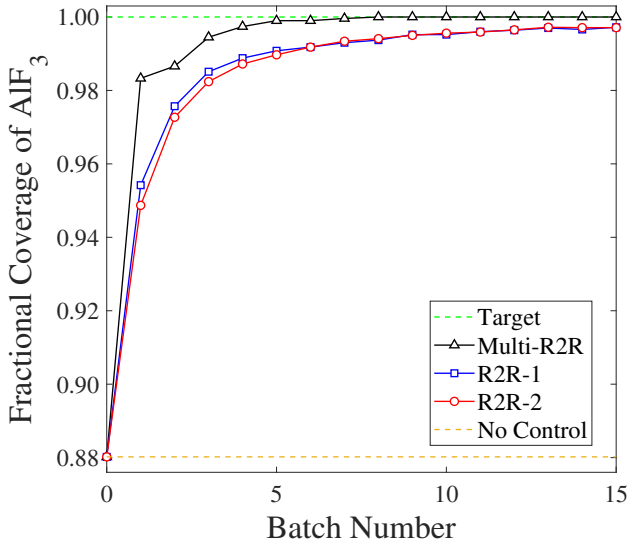
(a)



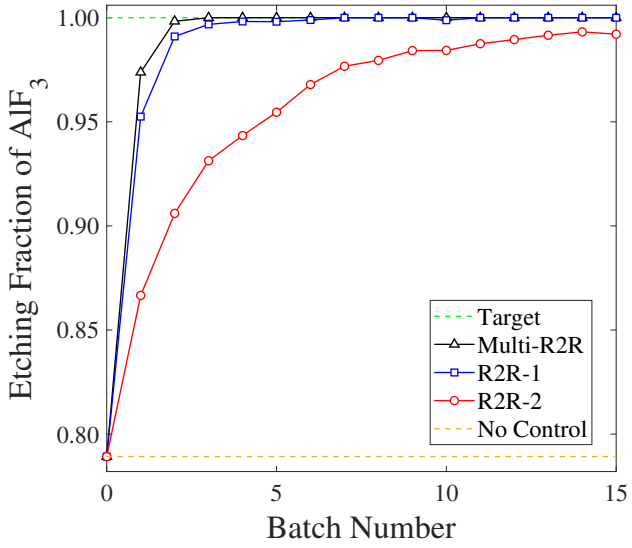
(b)

Figure 3.8: Comparison of the responses for various regression methods of R2R-1 under the presence of a kinetic disturbance for Steps A (a) and B (b). The weight factors (λ) of 0.3 and 0.1 are used for Steps A and B, respectively.

ever, In Step B, R2R-1 does not update the recipe after batch run 9 since it achieves a full etching fraction. R2R-2 also continues to increase the flow rate to get the target as shown in Figures 3.10b and 3.10d. In this chapter, there is no limit to the precursor flow rate even though there are valve



(a)



(b)

Figure 3.9: Comparison of the responses of various R2R control systems under the presence of a kinetic and pressure disturbance for Steps A (a) and B (b). The weight factor (λ) of 0.3 is chosen for all case studies. R2R-1 and the multivariable R2R system are simulated with the modified median-effect regression model.

opening constraints in the semiconductor industry. As predicted, the multivariable R2R algorithm manipulates the two input values such that both parameters (process time and flow rate) are signif-

icantly less than those of R2R-1 and R2R-2. Therefore, it is demonstrated that the multivariable R2R control system exhibits a stronger performance compared to that of the individual R2R-1 and R2R-2 control systems.

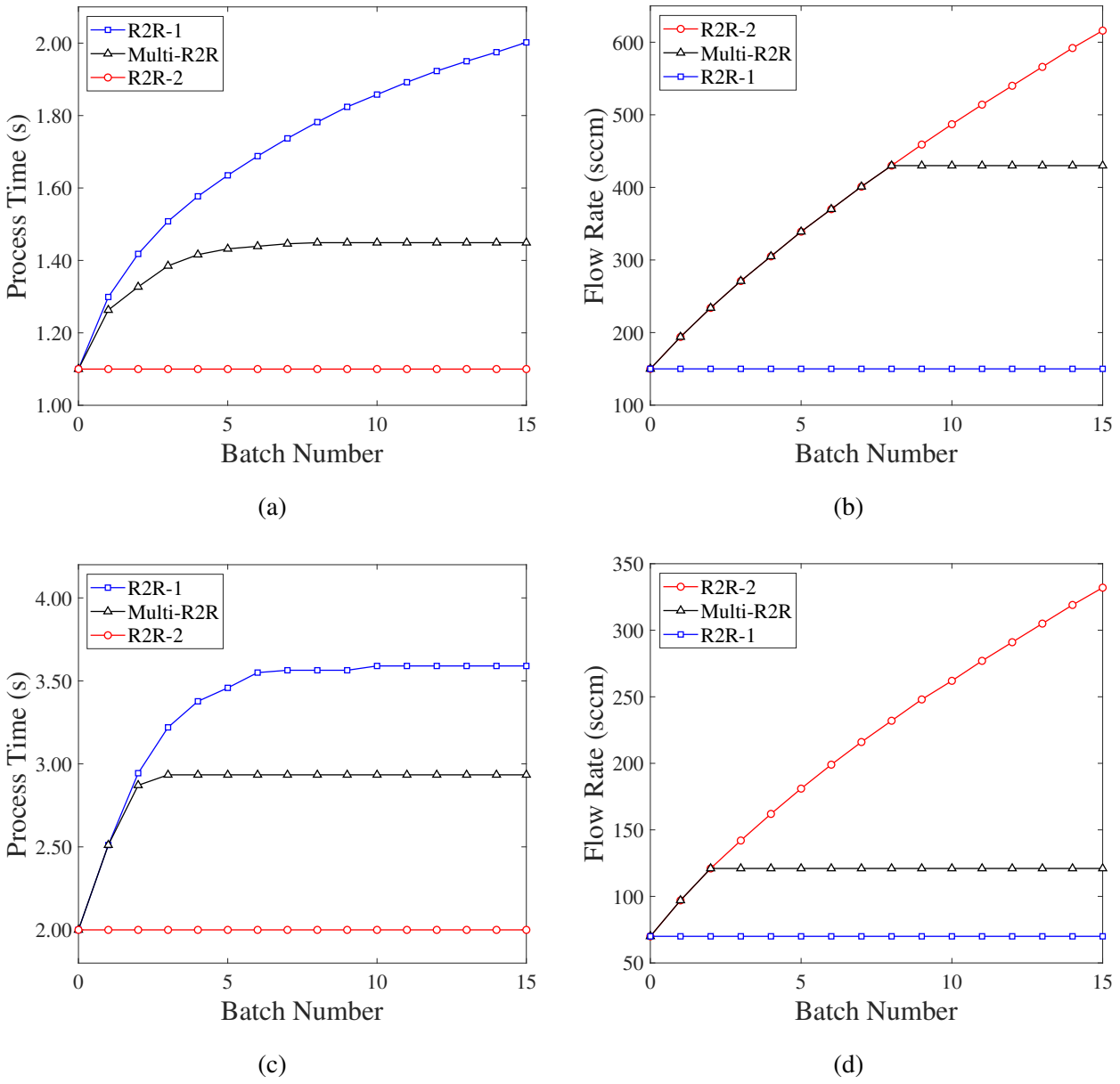


Figure 3.10: Progression of the adjustments made to the recipes (process time, precursor flow rate) in the presence of a kinetic and pressure disturbance through various EWMA-based R2R control systems for Steps A (a-b) and B (c-d).

3.5 Conclusion

In this chapter, the previously developed multiscale computational fluid dynamics (CFD) model for the thermal atomic layer etching (ALE) of aluminum oxide thin films was employed to build a multivariable R2R control system using an integration strategy. First, two individual R2R controllers (R2R-1 and R2R-2) were formulated based on the single-input-single-output (SISO) regression method using the exponentially weighted moving average (EWMA) algorithm. To significantly increase the linearity of the SISO model, a novel regression method, the modified median-effect, was implemented and compared with standard linearization and piecewise linearization. The modified median-effect method outperformed the other regression models and demonstrated the best fit of the multiscale CFD data using the EWMA method for the nonlinear system. R2R-1 was designed to adjust the valve opening time of precursor release to the reactor by measuring the coverage or etching fractions on the wafer, while R2R-2 was formulated to maintain the desired partial pressure of the precursor by manipulating the precursor flow rate into the reactor. Kinetic and pressure disturbances, which are industrially-relevant disturbances, were introduced to the system to determine the effectiveness of various R2R control algorithms. Consequently, this study substantiates that the multivariable R2R control scheme is able to successfully achieve complete coverage and etching and overcome the effects of disturbances in the least number of batch runs compared to that of the individual R2R control schemes implemented only one at the time.

Chapter 4

Sparse Identification Modeling and Predictive Control of Wafer Temperature in an Atomic Layer Etching Reactor

4.1 Introduction

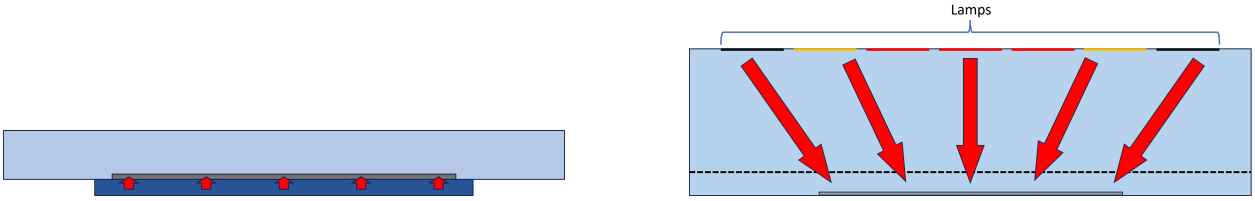
Over the last decade, there has been a rising demand for high-performance semiconductor chips such as 5-nm fin-field effect transistors (FinFETs) that are fabricated through advanced processes like extreme ultraviolet (EUV) lithography [64]. Due to their wide usage, there is a potential for a global supply shortage of advanced semiconductor chips, which are difficult to fabricate from silicon-based materials and consist of over 500 processing steps [65]. A considerable portion of these processing steps requires thermal processing under high temperatures, such as thermal atomic layer etching (ALE), to ensure conformal fabrication of these downscaled devices. Thermal ALE is a top-down fabrication procedure that improves the surface uniformity and alignment of transistors

by employing a system of reactions that produce volatile byproducts to enable surface etching. However, to ensure that complete byproduct removal is observed, the specific optimal operating temperature depends on the material of the substrate. For instance, aluminum oxide, Al_2O_3 requires high operating temperatures of 573 K, which has been proposed experimentally [40] and studied in prior *in silico* modeling work [47]. In addition to reaching the chosen operating temperatures, it is almost important that the temperature of the substrate surface is uniform. This ensures that the self-limiting surface reactions on the wafer will finish concurrently and minimize the processing time.

Prior work has extensively focused on the *in silico* modeling of thermal ALE processes through various reactor configurations [66, 67], during which a constant and uniform temperature, both on the substrate surface and operating environment, are assumed. This assumption overlooks the possibility of temperature fluctuations both on the wafer surface and in the overall reactor, which is widely reported in both laboratory experiments and numerical simulations [68]. Furthermore, these studies typically neglect the transient and dynamic nature of energy transfer among the wafer, reactor, and external environment. However, realistic thermal ALE processes often experience disturbances that disrupt the standard operating conditions of the process and introduce surface defects that result in product nonconformance, performance degradation, and lower productivity, depending on the magnitude of the disturbance. For example, a temperature fluctuation of 5 K will substantially change the half-cycle time, as observed in [47]. Therefore, precise temperature control is necessary to maintain the desired operating temperatures uniformly across the substrate surface for thermal ALE processes. In order to achieve a comprehensive understanding of the ALE process under practical industrial settings, it is critical to develop a transient model that characterizes the

physical heating process of the wafer.

Using heating lamps as the thermal source for a reactor is a popular and widely employed method in industrial practices [69]. Unlike a traditional heating plate where the primary heating mechanism is conduction, the primary heating mechanism for a heating lamp is thermal radiation, as shown in Figure 4.1. An advantageous feature of radiative heating is the high instantaneous



(a) (b)

Figure 4.1: Schematic figure that compares the (a) conventional heating plate structure and (b) novel radiative heating structure, where red arrows represent the direction of energy transfer.

changing rate in temperature that is beneficial for achieving fast controller response to mitigate disturbances. For instance, [68] determined that radiative heating can supply a maximum temperature rate of 200 K/s, which has been validated in *in silico* modeling work by [70]. Radiative heating also achieves better temperature uniformity on the wafer surface that is maintained by individually adjusting lamp powers, where in most industrial and experimental contexts, multiple groups of lamps are involved, and each group is independently controlled. Additionally, the temperature uniformity resulting from radiative heating is beneficial for the modern downscaling of semiconductors [71]. By combining the quick heating capabilities and improved temperature uniformity, radiative heating reduces the process time of the ALE processes by decreasing the time required to preheat the system and by providing more flexible control options [72].

To manifest the superior performance of radiative heating, the geometrical design of the reac-

tor is crucial. Various geometric configurations of lamp groups have been explored in prior works: a parallel configuration of individual lamps positioned directly above the wafer [68], a ring structure with varying radii above the wafer [73], and a structure where there are lamps both above the wafer and on the side of the wafer [70]. [74] also proposed a complete configuration with circular lamps positioned above, on the side, and beneath the wafer in a quartz process chamber. With the aforementioned reactor configurations, the temperature uniformity is significantly improved with individually controlled lamp power.

Nevertheless, prior studies have been primarily concentrated on optimizing the power supplied to the lamps to achieve a uniform temperature profile in a steady-state condition after thousands of seconds, which is similar to an open-loop control strategy [68, 75, 70, 73]. There is limited work focused on transient temperature analysis and real-time control. Due to the dynamic nature of industrial production and the prolonged time to reach steady-state, there is a greater need for dynamic temperature and uniformity control, especially for systems that fabricate wafers with diameters exceeding 200 mm, that operate at temperatures below 600 K, and that are sensitive to temperature changes [69]. Prior works such as [76] and [77] discussed the transient model and output feedback control of radiative heating processes, but with simpler reactor configurations and analytical transport equations. The analytical equations only work for ideal models that lack complex geometries and configurations, and for a modern industrially applicable system, a numerical computation model is required. To model the complex reactor accurately, in this study, a transient computational fluid dynamics (CFD) model incorporating physical heating devices is introduced. With the proposed model, the dynamic behavior of the temperature on the wafer surface and the ensuing energy transfer between the wafer, reactor, and surrounding environment can be analyzed.

The development of a dynamic model facilitates the application of a model predictive controller (MPC) for real-time feedback control over the wafer surface temperature to optimize the dynamic performance of the reactor. The objective of the controller is to conduct a rapid elevation of the wafer surface temperature from room temperature to the desired value, and then maintain the wafer surface temperature within a user-specified range around the target. Additionally, the controller is expected to preserve optimal surface temperature uniformity. Such a controller conforms with evolving demands of precise thermal control in *state-of-the-art* semiconductor technology.

The organization of this chapter is as follows: the computational fluid dynamics model setup is discussed in detail in Section 4.2, the dynamic modeling by the sparse identification method is examined in Section 4.3, the formulation of the model predictive controller is elucidated in Section 4.4, and the *in silico* experimental results are provided in Section 4.5.

4.2 Transient Modeling of Radiative Heating in ALE

This section analyzes the development of the radiative heating model conducted by heating lamps. In this chapter, the 2-D model for the radiative heating ALE reactor is first created. Following the construction of the 2-D reactor, a meshing procedure is performed to discretize the geometry into an appropriate mesh with sufficient quality to ensure numerical accuracy and efficiency. Lastly, the macroscopic computational fluid dynamics (CFD) simulation that is integrated with the radiation model is conducted to generate spatiotemporal temperature data on the wafer surface.

4.2.1 2-D Reactor Model and Mesh Generation

In this chapter, the macroscopic modeling framework begins by constructing the thermal ALE reactor geometry in computer aided design (CAD) software Ansys SpaceClaim. In order to conserve computational resources, a 2-D symmetrical modeling approach is adopted, which is illustrated in Figure 4.2. The wafer, with a diameter of 300 mm, is positioned at the base of the reactor, which has a cross-flow configuration [78]. Three distinct groups of lamps, the center, edge, and side lamp groups, are arranged above the reactor, and supplies heat to the surrounding environment in the form of thermal radiation. The center lamp group consists of three lamps, while the edge and side groups consist of two lamps each. The output power of each lamp group is independently adjustable, which provides a flexible control scheme with the potential to precisely modulate the wafer surface temperature. A quartz boundary with a small gap distance is positioned between the wafer and lamps, which serves as a physical barrier for internal gas flow while allowing light and energy transfer across the boundary. Additionally, external radiation from the room temperature environment is also defined as boundary conditions into the model to reflect the natural cooling process in a practical scenario [70].

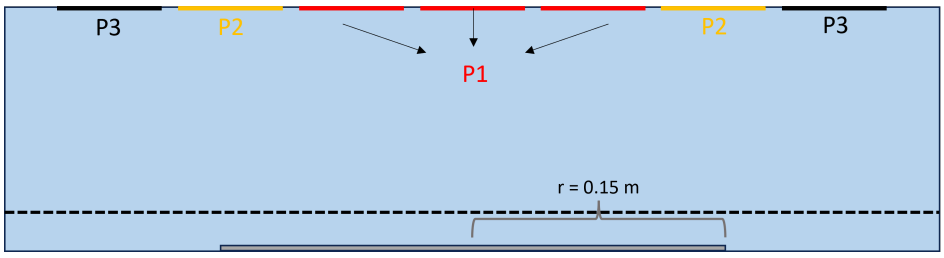


Figure 4.2: Schematic diagram of the 2-D thermal ALE, cross-flow reactor: The red lines are the Center lamp group; the yellow lines are the Edge lamp group; the solid black lines are the Side lamp group; the dashed black line is the quartz window; the gray rectangle part is the wafer. A comprehensive list of geometry parameters are displayed in Table 4.2.

The procedure to divide the CAD model into discrete cells for numerical calculation is called meshing, and the mesh quality plays a significant role in the accuracy, convergence, and stability of the numerical CFD simulation. In this chapter, Ansys Workbench Meshing is applied to create a discretized 2-D grid that is composed of rectangular cells with maximum cell distances of 0.5 mm. The mesh quality parameters, orthogonality, skewness, and aspect ratio, for rectangular cells in this chapter is within the acceptable range by the criteria recommended by [34] in all phases and boundaries, and it is summarized in Table 4.1. Additionally, the reactor was discretized into 28,000 cells to balance the computational efficiency and accuracy of the CFD simulation.

Table 4.1: The mesh quality acceptability criteria range and mesh parameters calculated from Ansys Fluent for the ALE reactor. For the orthogonality, the minimum value is presented. For the aspect ratio, the maximum value is presented.

Quality Indicator	Orthogonality	Skewness	Aspect Ratio	Number of Cells
Criteria	0.001 ~ 1*	0* ~ 0.95	1* ~ 8	N/A
ALE Reactor	0.998	0.002	1.46	28180

*Desired value for ideal mesh quality.

4.2.2 Macroscopic Energy Model

The computational fluid dynamics (CFD) simulation is conducted in the multiphysics software, Ansys Fluent. Since the energy transfer and surface temperature profile are major concerns in this study, only the heat transfer equation is calculated to save computational resources. The conservation of energy equation is described as follows:

$$\frac{\partial}{\partial t}(\rho E) + \nabla(\overrightarrow{v}(\rho E + p)) = -\nabla(\Sigma h_j J_j) + S_h \tag{4.1}$$

where ρ is the density of the fluid, E is defined as the internal energy of the system, $\overrightarrow{v}$ is the fluid velocity, p is the system pressure, h_j is the sensible enthalpy of the gas-phase species j, J_j is the diffusion flux of gaseous species j, and S_h is the heat transfer source flux rate.

The convective heat transfer originated from gas flow is also considered as a boundary condition on wafer. As demonstrated in a prior work [47], the gas flow in the thermal ALE process is laminar and has small surface flow velocities for small reactor volumes [79]. As a result, the convective heat transfer coefficient on the wafer is approximated as a fixed value where forced convection is negligible. A complete list of the parameters and constants used in the CFD model is outlined in Table 4.2. As the quartz window is thin, the amount of light absorbed by the quartz glass is neglected for simplicity, which implies that the quartz window is assumed to be fully transparent.

4.2.3 Discrete Ordinate Radiation Model

Various radiation models are provided in Ansys to calculate radiation intensity. The discrete ordinate (DO) model is one of the most reliable models to simulate radiative heat transfer with optically-thin and transparent media like air and thin glass in this study [34]. The DO model solves the radiative transfer equation (RTE) described in Equation (4.2) to obtain a spatiotemporal solution of the radiation intensity I within any media that absorbs, emits, and scatters light. The DO model considers $\overrightarrow{s}$ as a field function, and the slightly modified RTE equation is defined by Equation (4.3) below:

$$\frac{dI\left(\overrightarrow{r},\overrightarrow{s}\right)}{ds}+\left(a+\sigma_s\right)I\left(\overrightarrow{r},\overrightarrow{s}\right)=an^2\frac{\sigma T^4}{\pi}+\frac{\sigma_s}{4\pi}\int_0^{4\pi}I\left(\overrightarrow{r},\overrightarrow{s}'\right)\Phi\left(\overrightarrow{s}\cdot\overrightarrow{s}'\right)d\Omega' \tag{4.2}$$

Table 4.2: Parameters and constants of CFD model.

Parameter	Value	Units
Convective Heat Transfer Coefficient	2.5	$W/m^2/s$
Convective Flow Temperature	520	K
Wafer Diameter	300	mm
ALE Reactor Height	10	mm
ALE Reactor Diameter	460	mm
Gap Distance	83	mm
Single Lamp Group Width	60	mm
Interval Between Lamp Groups	6	mm
Maximum Lamp Power	5000	W/m^2
Minimum Lamp Power	0	W/m^2
Quartz Window Thickness	1	mm
Quartz Window Heat Capacity	964	J/kg/K
Quartz Window Absorption Coefficient	0	m^{-1}
Quartz Window Thermal Conductivity	1.67	W/m/K
Initial Temperature	298	K

$$\nabla \cdot \left(I\left(\overrightarrow{r}, \overrightarrow{s}\right)\overrightarrow{s}\right) + (a+\sigma_s)\, I\left(\overrightarrow{r}, \overrightarrow{s}\right) = an^2\frac{\sigma T^4}{\pi} + \frac{\sigma_s}{4\pi}\int_0^{4\pi} I\left(\overrightarrow{r}, \overrightarrow{s}'\right)\Phi\left(\overrightarrow{s}\cdot\overrightarrow{s}'\right) d\Omega' \quad (4.3)$$

where $\overrightarrow{r}$ is the position vector, $\overrightarrow{s}$ is the direction vector, $\overrightarrow{s}'$ is the scattering direction vector, s is the path length, a is the absorption coefficient, n is the refractive index, σ_s is the scattering coefficient, σ is the Stefan-Boltzmann constant, I is the radiation intensity, T is the local temperature, Φ is the phase function, and Ω' is the solid angle. The items on the left-hand side of the conservation equation represent the sum of the spatial derivative of radiation intensity and absorption of light, while the items on the right-hand side of the equation represent the sum of the emission and scattering of

light. A detailed description and guide to the calculation theories are further examined in [34].

The radiation-related parameters for the wafer, walls, and heating lamps are listed in Table 4.3. Additionally, radiation permeation is disabled for all wall boundaries except for the quartz window. For the DO method, it is assumed that the light beams transmit in a finite number of directions

Table 4.3: Parameters and constants of the radiation model.

Parameter	Value	Units
Wafer Emissivity	0.7	N/A
Wafer Diffuse Fraction	0.3	N/A
Side Wall Emissivity	0.3	N/A
Side Wall Diffuse Fraction	0.3	N/A
External Radiation Temperature	300	K
External Radiation Emissivity	1.0	N/A

to make the numerical calculation possible. In this 2-D system, the discretization of the angular directions is implemented by separating a space of 360° into partitions, which are further separated into angular divisions that are called control angles. The default settings in Ansys Fluent divide the angular space into eight partitions with 45° intervals, and additional control angles can be specified to improve the accuracy of the numerical methods, as demonstrated in Figure 4.3. The number of control angles defined by the user plays a substantial role in both the accuracy and computational complexity of the simulation. To find the optimal balance between numerical accuracy and computational efficiency, a grid search method is implemented through extensive test runs with the goal of finding the smallest number of control angles that will result in a minimal difference in simulation results when compared to test runs with more control angles. this chapter determined an optimized

value of 8 for the number of user-defined control angles in each default partition zone, for a total of 64 control angles. However, the discretization may encounter issues in areas with irregular geometry and cause control angles to become overhanging angles, which results in unwanted reflection and refraction of light beams [34]. To reduce the risk of overhanging angles appearing, rectangular cells are used in the meshing component of this chapter.

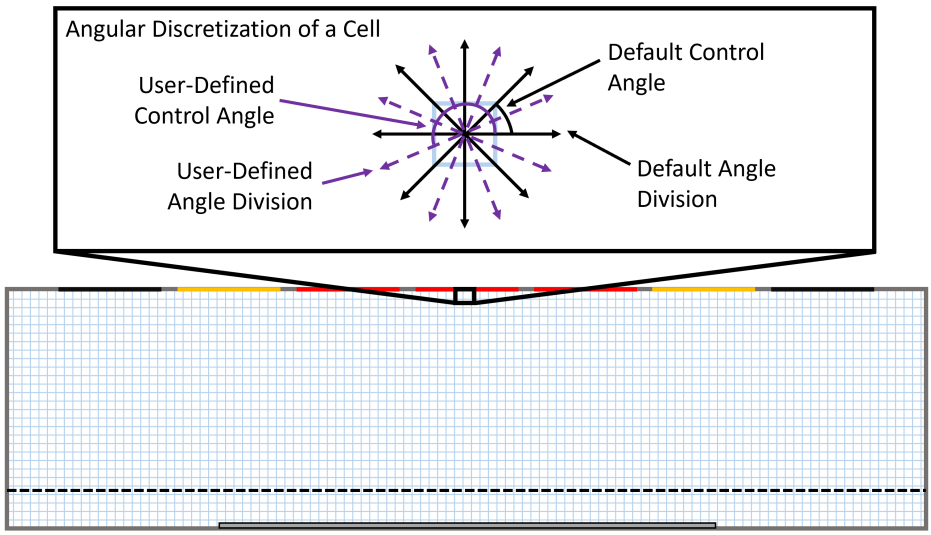


Figure 4.3: Angular discretization of an adjacent, 2-D rectangular cell on a heating lamp, where control angles are generated to reflect the light emissions produced from the heating lamps.

4.2.4 Implementation and Monitoring

Given the transient and dynamic nature of the model, the controller frequently adjusts the lamp power to adapt to perturbations in the temperature on the wafer surface. Any changes in the lamp power are formulated as a linear interpolation function between the two consecutive power levels determined by the controller to avoid potential numerical error produced from abrupt changes to the lamp power and reflect the continuously changing nature of the lamps. The calculation of the

lamp power at time t is expressed by the following equation:

$$P_i(t) = P_i(t_0) + \frac{t - t_0}{t_s}(P_i(t_1) - P_i(t_0)), \quad i = 1, 2, 3 \tag{4.4}$$

where $i = 1, 2, 3$, corresponds to the center, edge, and side lamp groups, respectively, $P_i(t)$ is the power of lamp group i at time t, $P_i(t_0)$ is the power of lamp group i at the start of the current sampling time period, $P_i(t_1)$ is the power of lamp group i determined by controller with feedback of temperature measured at t_0, and t_s is the length of one sampling time period.

To accurately represent the surface temperature profile, four temperature inspection positions are defined on the wafer surface. These locations include the center point ($r = 0$ m), the first trisection point ($r = 0.05$ m), the second trisection point ($r = 0.10$ m), and the edge point ($r = 0.135$ m), where r is the distance between the inspection points and the center of the wafer. The outermost 10% of the wafer is omitted from the region of interest due to inherent cooling effects on the periphery [68]. The controller described in the following sections will use the temperatures monitored at these designated inspection points as feedback values to maintain the temperature within an acceptable range around the target temperature. For the purpose to regulate the process time of ALE within operable range determined by [47], a target temperature over 570 K is required on wafer surface, but not exceeding too much for saving energy cost. On the other hand, the acceptable range is determined by the criteria that boundary temperatures have less than 20% half-cycle time difference with target temperature. By microscopic simulation result conducted by [47], the acceptable range within ±3 K fulfill the above criteria within temperature range of $[570, 580]$ K with half-cycle time around 0.8 s and 1.0 s. Consequently, 573 ± 3 K is determined as the goal for

temperature control.

4.3 Sparse Identification Modeling

One key aspect in accurately predicting system behavior is a dynamic model that fully captures the time derivatives of the state variables. For simple process systems with basic kinetics, such as a continuous stirred tank reactor (CSTR) or a plug-flow reactor (PFR), first-principle models can be constructed using mass and energy balances [e.g., 80]. However, these standard first-principle modeling procedures are unavailable for most complex chemical processes, such as the ALE reactor examined in this chapter. As a result, data-driven approaches have emerged as effective tools for modeling nonlinear processes; for example, there is the numerical method for subspace state-space system identification [81], the polynomial nonlinear state-space method [82], and machine learning methods like recurrent neural networks [83, 84]. Recently, a novel method for building dynamic models from open-loop data called sparse identification of nonlinear dynamics (SINDy) has gained attention for its effectiveness in several complex engineering systems [e.g., 85], such as modeling a nonisothermal CSTR reactor [86] and fluid vortex shedding behind a cylinder [18]. Of which, the latter case is a nonlinear system whose underlying dynamics were identified after nearly 30 years by field experts. SINDy has also been used directly with input/output data (known as generalized SINDy or GSINDy) and the Kalman filter to construct time-variant digital twin models [87]. The Kalman-GSINDy approach was subsequently used along with the proper orthogonal decomposition (POD) to find the lifting functions to find reduced-order Koopman linear models for incorporation into MPC. SINDy is a data-driven method that utilizes discrete measurements from a physical

system to identify a first-order ordinary differential equation (ODE) of the following form:

$$\dot{\tilde{x}}(t) = f(\tilde{x}(t), u(t)) \tag{4.5}$$

where $\tilde{x}(t)$ represents the state vector of the SINDy model, and $f(\tilde{x}(t), u(t))$ is a function of state and input variables that captures the dynamics of the underlying physical laws that govern the system. The SINDy model is applied with the goal of system identification; of note, this does not necessarily mean accurately representing the true underlying physics of the process. The key assumption of the SINDy algorithm is that $f(\tilde{x}(t), u(t))$ is sparse and only has a few nonlinear terms. Although there is a large pool of candidate nonlinear functions, only a few terms are expected to be active and contribute to system dynamic identification. The sparsity of the SINDy model facilitates calculating the nonzero coefficients, while also preventing overfitting.

4.3.1 Open-Loop Data Generation

The training, or fitting, of the SINDy model requires measurements of three quantities: the state variables, the manipulated inputs, and the time-derivative of the state variables. The state variables in this study are the temperature inspection points on the wafer: T_1 at $r = 0.00$ m, T_2 at $r = 0.05$ m, T_3 at $r = 0.10$ m, and T_4 at $r = 0.135$ m, where r is the distance from the center of the wafer. The control variables are the power settings for the three lamp groups, P_1, P_2, and P_3, as depicted in Figure 4.2.

The training data is generated via open-loop simulations that heat the reactor and wafer from an ambient temperature of 298 K using fixed lamp powers. Both uniform and nonuniform lamp

power combinations, as detailed in Table 4.4, are used to generate the spatiotemporal temperature data that is then used to create the dynamic model. The temperatures on the four inspection points and corresponding lamp powers are recorded in each simulation from $t = 0$ s to $t = 2000$ s, which is when the system is indistinguishable from its final steady-state. Variable sampling times were also used to account for the fast dynamics at the start of each run. Examples of transient profiles of open-loop data are presented in Figure 4.4.

Table 4.4: List of lamp powers used to generate open-loop data (W/m^2).

P_1	P_2	P_3
1000	1000	1000
2000	2000	2000
3000	3000	3000
4000	4000	4000
5000	5000	5000
2000	2000	4000
1500	2500	3500
1000	5000	3000
1000	1000	5000
450	450	4000

To achieve desired model performance, the range of operating conditions in training data is larger than the operating range of interest. The target temperature in the chapter is 573 ± 3 K, and the generated open-loop data set contains final steady-state temperatures from less than 540 K to over 700 K. Moreover, the coupling effects of control variables are also necessary to be taken into consideration, as 5 groups of nonuniform lamp power data with various power ratios were

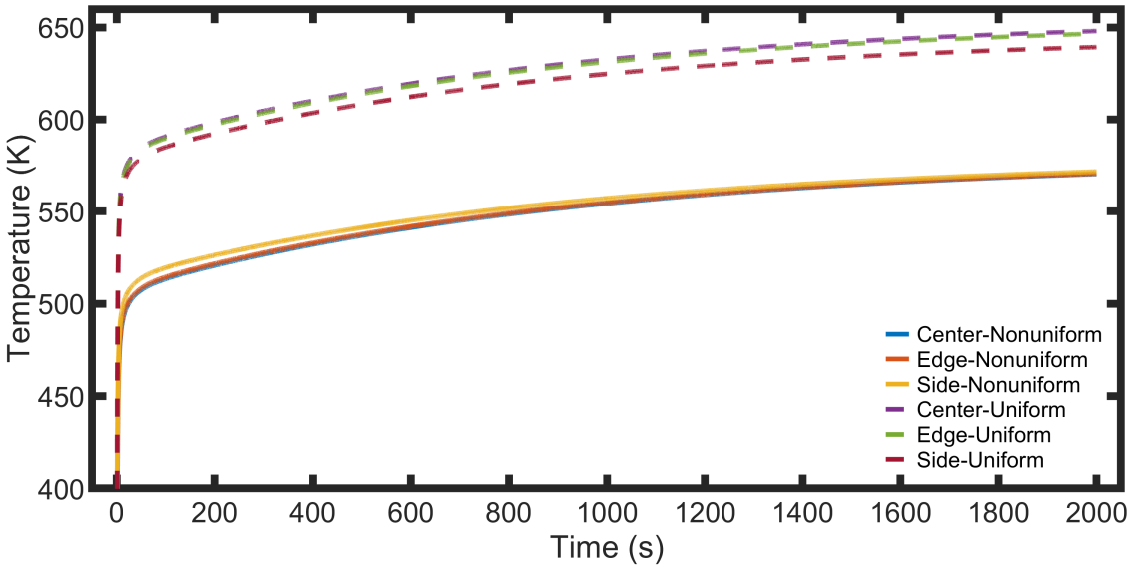


Figure 4.4: Temperature profiles of sample open-loop datasets used for model training to illustrate the surface temperature uniformity for various lamp power configurations. Dashed lines are for the uniform 2000 W/m^2 case and solid lines are for the nonuniform case where $P_1 = P_2 = 450$ W/m^2, $P_3 = 4000$ W/m^2. The side surface temperature is over 10 K lower than center temperature at $t = 2000$ s in the case of uniform lamp power; in contrast, nonuniform lamp power results in better surface temperature uniformity.

involved in model training. Additionally, process safety is an inevitable issue when obtaining data from real world experiments under the wide range of operating conditions; however, computational simulation methods such as CFD avoid the potential safety risk.

The time derivative of the temperature generated from the open-loop data is calculated using a first-order numerical method as per the equation,

$$\dot{\vec{T}}_t = \frac{\vec{T}_{t+1} - \vec{T}_{t-1}}{2\Delta t} \tag{4.6}$$

where $\vec{T}_t$ is the measurement temperature vector containing all 4 temperature measurement points at time t, $\vec{T}_{t+1}$ is the temperature vector at the subsequent time step, $\vec{T}_{t-1}$ is the temperature vector at the previous time step, and Δt is the length of the time step. In order to reduce the computational

cost while preserving simulation accuracy, a time-varying step size is implemented during the open-loop simulation. A much smaller integration time step is used for the first 5 s of the simulation duration when the temperature increases rapidly, while increasingly larger integration time steps are employed for the remainder of the simulation as the temperature increases slowly therein. The detailed distribution of the integration time step is described in Table 4.5.

Table 4.5: List of Δt applied during different time regions.

Time Region	Δt
0 s $\rightarrow$ 5 s	0.01 s
5 s $\rightarrow$ 20 s	0.02 s
20 s $\rightarrow$ 75 s	0.05 s
75 s $\rightarrow$ 2000 s	0.1 s

Analysis of Figure 4.4 shows a distinct trend: temperatures at all four points initially rise rapidly but then rise slowly for the remainder of the simulation, which leads to larger time derivatives in the first few seconds compared to the majority of the simulation. This phenomenon leads to a data imbalance problem, i.e., if the data used for modeling spans the entire time frame, then the model will accurately capture neither the fast transient nor the subsequent slow rise. Instead, selectively using only the early simulation data with the fast transient changes and discarding the remainder of the simulation is often proposed as one intuitive solution to maximize the ability of the model to capture the dynamic behavior. However, the temperature at the end of the initial fast transient step remains significantly lower than the final, steady-state temperature, which deteriorates the ability of the model to capture the final steady state when using such an approach. Therefore, an innovative data reconstructing methodology is proposed, where each trajectory is reconstructed according to

the following function in time:

$$T_{\text{new}}(t) = \frac{(T_{\text{final}} - T_{\text{init}}) \cdot t}{1 + t} + T_{\text{init}} \tag{4.7}$$

where T_{new} is an arbitrary, reconstructed temperature at any time t, T_{final} and T_{init} are the true final and initial temperatures under the designated open-loop simulation, respectively, and t is the time. This formulation guarantees that the reconstructed data will have the same initial and final temperatures as the original data since $T_{\text{new}} \rightarrow T_{\text{init}}$ as $t \rightarrow 0$, while $T_{\text{new}} \rightarrow T_{\text{final}}$ as $t \rightarrow \infty$. Importantly, the dynamic behavior of the reconstructed function is altered from the original, such that the temperatures reach their steady states earlier. Additionally, since deriving an analytical derivative function is possible for the reconstructed data, the temperature derivative is calculated as follows:

$$\dot{T}_{\text{new}}(t) = \frac{T_{\text{final}} - T_{\text{init}}}{(t + 1)^2} \tag{4.8}$$

An example of the comparison between the original and reconstructed plot is illustrated in Figure 4.5. To overcome the data imbalance problem, since the fast transient behavior still occurs within the first 5 s in the reconstructed data set, only the first 20 s of the reconstructed trajectory is used because the temperature of the reconstructed trajectory at $t = 20$ s is sufficiently close to the final, steady-state temperature, unlike the original temperature trajectory, which enables the model to capture the steady-state dynamics while retaining the shape and trend of the original data.

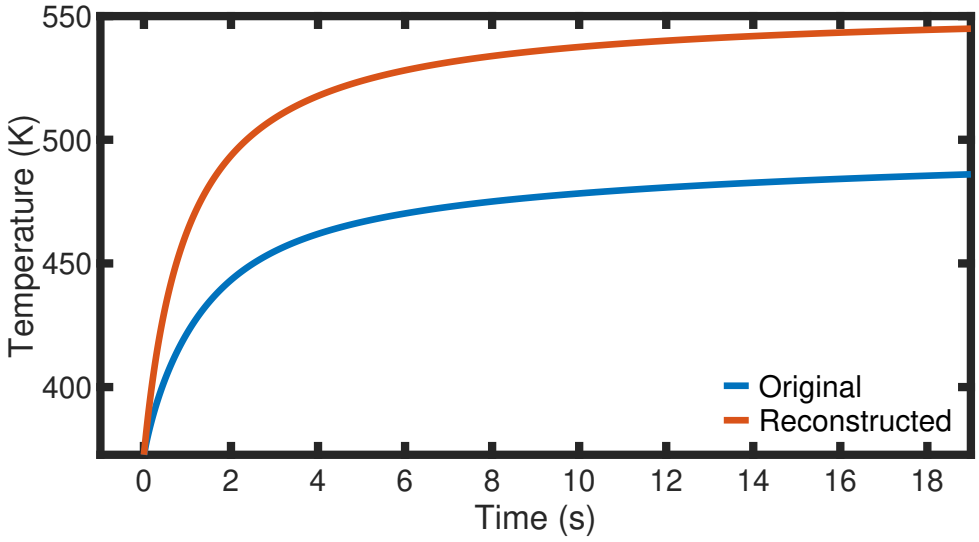


Figure 4.5: Comparison of the temperature profiles of the original and reconstructed data. The reconstructed temperature approaches the final steady-state value within 20 s, which enables the dynamic model trained using the reconstructed data to capture the correct steady state while retaining the trend of the original data.

4.3.2 Dynamic Model Development

To train the SINDy model, the open-loop data obtained from the data generation process is rearranged into two matrices: the state matrix X and the control input matrix U, both of which are defined as follows:

$$X = \begin{bmatrix} x_1(t_1) & x_2(t_1) & \dots & x_n(t_1) \\ x_1(t_2) & x_2(t_2) & \dots & x_n(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ x_1(t_m) & x_2(t_m) & \dots & x_n(t_m) \end{bmatrix} \tag{4.9}$$

$$U = \begin{bmatrix} u_1(t_1) & u_2(t_1) & \dots & u_p(t_1) \\ u_1(t_2) & u_2(t_2) & \dots & u_p(t_2) \\ \vdots & \vdots & \ddots & \vdots \\ u_1(t_m) & u_2(t_m) & \dots & u_p(t_m) \end{bmatrix} \tag{4.10}$$

where $x_i(t_k)$ is the i^{th} state variable at the k^{th} sampling time, $u_j(t_k)$ is the j^{th} control variable at the k^{th} sampling time, m is the number of time-series data points, n is the number of state variables, and p is the number of control input variables. With the state matrix and control input matrix, a function library $\Theta(X, U)$ is developed that consists of candidate linear and nonlinear features of both the state and control input variables. An example of a function library is

$$\Theta(X,U) = \begin{bmatrix} | & | & | & | & | & | & | \\ \mathbf{1} & X & U & X^2 & U^2 & \sin X & \cdots \\ | & | & | & | & | & | & | \end{bmatrix} \tag{4.11}$$

Each candidate function in $\Theta(X, U)$ is then assigned a pre-multiplying coefficient to fit the derivative of the data, as shown in Equation (4.12) below:

$$\dot{X} = \Theta(X,U)\Xi \tag{4.12}$$

where Ξ is the coefficient matrix that stores the coefficients associated with each basis function for each dependent variable. To find Ξ, Equation (4.12) is solved by Lasso regression, an optimization-based linear regression with first-order norm regularization. The Lasso optimization problem to solve for Ξ can be formulated as follows:

$$\Xi = \arg\min_{\Xi' \in \mathbb{R}} \|\Theta(X,U)\Xi' - \dot{X}\|_2 + \lambda|\Xi'|_1 \tag{4.13}$$

where Ξ' is a dummy variable that replaces Ξ for the optimization expression and λ is the regularization coefficient. The second term that includes λ is the first-order regularization term that

encourages sparsity in the Ξ matrix by zeroing all coefficients with an absolute value less than λ at each iteration. The optimization is solved using the Lasso regression function in the Python package Scikit Learn (in which the parameter α corresponds to λ as per the formulation of Equation (4.13)).

The state variables in this chapter are the four temperature measurement points on the wafer, $T_i, i \in \{1, 2, 3, 4\}$, and the control input variables are the power levels for the three lamp groups, $P_j, j \in \{1, 2, 3\}$. The four temperatures are decoupled from each other and modeled using four separate SINDy models, leading to the following sparse identification problem formulation:

$$\dot{T}_i = \Theta_i(T_i, P)\Xi_i, \qquad i \in \{1, 2, 3, 4\} \tag{4.14}$$

where $T_i \in \mathbb{R}^m$ is the i^{th} temperature state variable vector with time-derivative $\dot{T}_i$, $P = [P_1 \quad P_2 \quad P_3] \in \mathbb{R}^{m\times 3}$ is the lamp power control input matrix, and Ξ_i is the coefficient vector for each basis function for the ODE corresponding to the i^{th} temperature state. The variables T_i and P correspond to X and U in Equation (4.12), respectively. The rationale behind employing a distinct SINDy model for each state variable is to introduce sparsity during the candidate function selection process. The derivative of each state variable is presumed to be solely dependent on its own value, without the influence of other state variables. The candidate functions for each Θ_i implemented in this chapter are a bias term and a combination of linear and quadratic functions of the states and inputs, as follows:

$$\Theta_i(T_i, P) = \begin{bmatrix} | & | & | & | & | & | \\ \mathbf{1} & T_i & P & T_i \odot T_i & P \odot P & T_i \odot P \\ | & | & | & | & | & | \end{bmatrix} \tag{4.15}$$

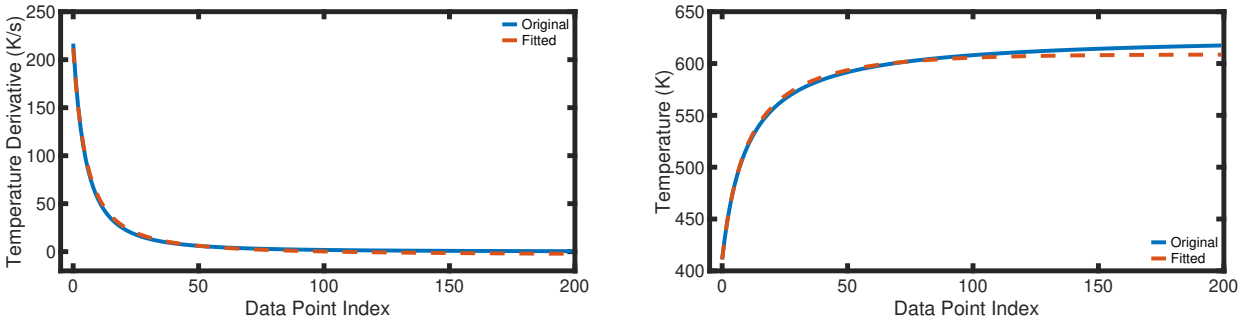
where $\mathbf{1} \in \mathbb{R}^m$ is a column vector of ones that serves as the bias term in linear regression, and

$\odot$ denotes element-wise multiplication except in the term $T_i \odot P$, where it denotes element-wise multiplication between the i^{th} temperature state variable vector and each lamp group's power control vector, i.e., $T_i \odot P = [T_i \odot P_1 \quad T_i \odot P_2 \quad T_i \odot P_3]$. As a result, $\Theta_i(T_i, P)$ is the $\mathbb{R}^{m\times 12}$ feature map matrix of $\dot{T}_i$. To improve the conditioning of the sparse identification problem, each candidate function/column in $\Theta(T, P)$ is multiplied by an additional coefficient to standardize all columns to a comparable scale. For this chapter, the regularization parameter λ is set as 0.01.

The choice of basis functions is highly dependent on the system under consideration. The natural occurrence of polynomials and trigonometric functions in engineering systems makes them an appropriate first choice for the basis functions when using SINDy [18]. However, the basis set can and should be adapted based on both performance of the method and, more importantly, any prior knowledge of the system dynamics. For example, for chemical reactors, the temperature-dependence of the reaction kinetics is based on the Arrhenius law, which suggests using exponential terms in the reactor temperature dynamic equation in the SINDy library [86]. Conversely, trigonometric functions form an appropriate basis for power systems [88]. In the ALE process studied, since quadratic polynomials were sufficient to accurately capture the system dynamics when used in conjunction with the proposed data reconstruction scheme, no other basis functions were considered.

The accuracy of the trained SINDy model can be represented by derivative fitting plots and state prediction plots. The derivative fitting plot for when the lamp power is set at a uniform 2000 W/m^2 is shown in Figure 4.6a, which demonstrates that the trained SINDy model fits well to the derivative data. The state prediction plot is illustrated in Figure 4.6b, and it compares the true and predicted temperatures. The predicted trajectory is obtained by integrating the trained SINDy

model using the explicit Euler method from the same initial temperature measurement as the true data. By examining the state prediction plot, the predicted temperature from the model is 8.81 K lower than the true temperature at the end of the trajectory at $t = 20$ s. As the total change in temperature for the true trajectory is 190 K, this represents an error of less than 5% error compared to the true trajectory. Furthermore, while the entire 20 s of the prediction in Figure 4.6b is initialized once from the temperature at t_0, in practice, when the SINDy model is incorporated into a model-based controller, it will typically predict over much smaller time periods than 20 s with a single initialization, which will yield much smaller errors. Overall, the results suggest that the dynamic model developed fulfills the requirement of the controller.



(a) (b)

Figure 4.6: (a) Derivative fitting and (b) state prediction plots for checking model performance. The fitted line in (a) closely aligns with the original data, suggesting that the SINDy model successfully captures the system dynamics. The fitted line in (b) shows strong correspondence between the predicted temperature and original data, where the maximum absolute error is 8 K when predicting 20 s of temperature.

4.4 Model Predictive Control

Model predictive control is an optimization-based advanced control strategy that utilizes a dynamic model to predict the future values of the state variables and then optimizes a cost function to find

the optimal control action. The general formulation of model predictive controller is shown in the following set of equations:

$$
\begin{aligned}
\min_{u(t)\in S(t_s)} \quad & \int_{t_k}^{t_{k+N}} L(\tilde{x}(\tau), u(\tau))\, d\tau \\
\text{s.t.} \quad & \dot{\tilde{x}}(t) = f(\tilde{x}(t), u(t)) \\
& \tilde{x}(t_k) = x(t_k) \\
& u(t) \in U,\ \forall\, t \in [t_k, t_{k+N}) \\
& g_{\text{MPC},1}(\tilde{x}(t), u(t)) = 0 \\
& g_{\text{MPC},2}(\tilde{x}(t), u(t)) \leq 0
\end{aligned}
$$

where $L(\tilde{x}, u)$ is the cost function to be optimized, $u(\tau)$ is the set of control actions and inputs that the controller receives, $f(\tilde{x}, u)$ is the dynamic model used for state prediction, $\tilde{x}(t)$ is the predicted state variable, $x(t_k)$ is the measured state variable from the physical system at time t_k where $[t_k, t_{k+N})$ represents the prediction horizon in model predictive control, and $g_{\text{MPC},1}$ and $g_{\text{MPC},2}$ denote equality and inequality constraints, respectively.

Model predictive control has received recognition for commercial applications because of its advantages over traditional controllers [13]. Unlike traditional controllers that implement control based on the feedback of offsets, such as the proportional-integral (PI) controller, an MPC foresees the future state values to make optimized control actions. PI controllers may encounter difficulties when trying to achieve a fast response with minimal overshoot due to their reliance on the continuously updated integral term to eliminate bias. On the other hand, with the predictions made by the dynamic process model, the MPC is able to optimize the control actions to quickly reach the

setpoint while eliminating severe overshoots. Moreover, the MPC is capable of controlling highly nonlinear processes with constraints in the control actions and states, which is challenging for traditional linear controllers that cannot explicitly account for the presence of constraints [89, 90]. Model predictive control also offers flexibility in the control objectives, as the cost function can be customized to optimize for a variety of objectives, such as to maximize a profit or yield function in economic MPC.

4.4.1 Implementation of Model Predictive Controller

In order to implement model predictive control, it is necessary to define the sampling time and prediction horizon and construct the cost function and constraints for the state and control variables. The sampling time t_s is set to 0.2 s in this chapter, signifying that the temperature is measured every 0.2 s, and the control action is applied and held constant for 0.2 s until the next control action is applied. The prediction horizon is set to 3 in this chapter, denoting that the controller forecasts three sampling periods (0.6 s) to calculate the cost function for optimization. The cost function and constraints used in the MPC are described below:

$$\text{Cost Function:} \quad L(T,P) = \sum_{i=1}^{4}(T_i(t) - 573)^2 + \beta \sum_{j=1}^{3}(P_j(t) - P_j^{ss})^2 \tag{4.16}$$

$$\text{Input Constraint:} \quad P_j \in [0, 5000] \text{ W/m}^2 \tag{4.17}$$

$$\text{Input Constraint:} \quad P_j(t + t_s) - P_j(t) \leq 250 \text{ W/m}^2 \tag{4.18}$$

where $T_i(t)$ is the predicted value of the measured temperature on inspection point i at time t; $P_j(t)$ denotes the power of lamp group j at time t, as the power is adjusted at the start of each

sampling period. The cost function consists of two components. The first term is the sum of squared differences between the predicted temperatures and the target temperature of 573 K. The predicted temperatures are calculated by integrating the SINDy model using the explicit Euler method starting at the state measurement at $t = t_k$. The second component is the sum of the squared differences between the optimized lamp power and the assigned steady-state values. The second term serves as a steady-state regularization on the input lamp power, where β is a tunable coefficient. A high β value drives the optimized lamp power closer to the assigned steady-state value, while a small β value encourages a lamp power that minimizes the deviation between the predicted and target temperatures. Based on extensive tests, $\beta = 1 \cdot 10^{-4}$ is used for involving moderate influence of the steady-state power in the optimization process to reach the target quickly while keeping stability at the same time. The upper bound of lamp power and maximum adjusting rate are set to 5000 W/m^2/s and 250 W/m^2/s, respectively, to account for physical constraints in practical lamps. The Python package CyIPOPT is used for solving the constrained nonlinear optimization problem to obtain the optimized lamp power.

4.4.2 Feedback-Based Time-Varying Steady State in MPC

It is important to note that both the reactor and wafer are at room temperature at $t = 0$. The components of the reactor are simultaneously heated with the wafer by the lamps via radiative heat transfer. The walls of the reactor and the lamp surfaces also contribute to additive amounts of radiation through reflection, emission, and transfer of external radiation. As a result, the temperature of the environment around the wafer changes continuously as ambient surface temperatures rise with time. This changing nature of the ambient temperature implies that the required "steady-state"

power is not constant during the process, i.e., the power required to maintain the wafer at the target temperature is unique at any given time. From a modeling perspective, the same input vectors of surface temperatures and lamp powers can produce completely different temperature derivatives as the incoming energy from the reactor and environment is continuously changing. Moreover, since reconstructed data was used during the training of the process model, deviations exist between the reconstructed derivative data and the original derivative data, particularly in the initial region, which also must be accounted for in the controller. Consequently, a novel approach of a feedback-based time-varying steady-state P_{ss} (VSS) is proposed where the steady-state lamp power is dynamically adjusted to accommodate the continuously changing temperature of the wafer environment and compensate for the derivative differences. The VSS algorithm is described in Algorithm 1.

Algorithm 1: Feedback-Based Time-Varying Steady-State Update

Data: t; T_t; S_{old}; P^{ss}_{old}

Result: S_{new}; P^{ss}_{new}

Parameters: P^{ss}_{final}; P^{ss}_{0}; ΔP

/* Read t, T_t from simulation of last sampling time */

/* Read S_{old}, P^{ss}_{old} from saved text file */

if $t \leq 50$ s **then**

  $S_{\text{new}} = 0$;

else if $\min(T_t) \leq 571.0$ K **and** $S_{old} = 0$ **then**

  $S_{\text{new}} = 1$;

else if $\max(T_t) \geq 574.5$ K **and** $S_{old} = 1$ **then**

  $S_{\text{new}} = 0$;

else

  $S_{\text{new}} = S_{old}$;

/* Switch variable S_{new} is updated */

if $P^{ss}_{old} \leq P^{ss}_{final}$ **then**

  $P^{ss}_{\text{new}} = P^{ss}_{final}$;

else if $t \leq 2.5$ s **then**

  $P^{ss}_{\text{new}} = P^{ss}_{0}$;

else if $S_{new} = 0$ **then**

  $P^{ss}_{\text{new}} = P^{ss}_{old} - \Delta P \cdot dt$;

else if $S_{new} = 1$ **then**

  $P^{ss}_{\text{new}} = P^{ss}_{old}$;

/* New steady-state power P^{ss}_{new} is updated */

/* Send P^{ss}_{new} to CFD simulation */

/* Write S_{new}, P^{ss}_{new} to saved text file */

In the VSS algorithm, the steady-state power is initialized with lamp powers of $P^{ss}_{0,1} = P^{ss}_{0,2} = 2400$ W/m^2, $P^{ss}_{0,3} = 5000$ W/m^2 and kept constant when $t \leq 2.5$ s to a achieve a quick response at the start of the time when the reactor is cool. Afterward, a switch variable S is introduced to serve as an indicator of the system status. For the first 50 s, S is set to 0 to allow the steady-state power to continuously decrease at a rate of $\Delta P_1 = \Delta P_2 = 8.1$ W/m^2/s, $\Delta P_3 = 4.6$ W/m^2/s to avoid a potential overshoot and account for the heated reactor. The temperature is constantly monitored, and if the minimum temperature is lower than 571.0 K, the system indicator variable S changes from 0 to 1, and P^{ss} stops decreasing to allow the temperature to gradually increase back towards the target temperature until the maximum temperature measured on the inspection points is higher than 574.5 K, which is when P^{ss} resumes decreasing. These two critical temperatures are within the bounds of the acceptable range of $[570, 576]$ K and are expected to enable the VSS strategy to control the temperature within the range of $T \in [571.0, 574.5]$ K at all times. The power is expected to reach $P^{ss}_{final} = [P^{ss}_{final,1}\ P^{ss}_{final,2}\ P^{ss}_{final,3}] = [400\ 400\ 3866]$ W/m^2 when the system reaches the final steady-state condition after a sufficiently long time. P^{ss}_{final} has been verified to produce a uniform temperature profile around the target temperature through an independent steady-state simulation.

4.5 Results and Discussion

4.5.1 Open-Loop Control Performance

The open-loop control strategy fixes the lamp power at P^{ss}_{final}, where the transient temperature profile is shown in Figure 4.7. With this setup, the wafer temperature only reaches the target temperature after 2000 s of process time. P^{ss}_{final} is defined as the power used in this final steady-state

scenario where the reactor is thoroughly heated up. Of note is that, in this situation, the power required to maintain a steady-state is much lower than during the initial stage of the reactor. As a result, the temperature increases rapidly during the initial stage and very slowly afterward due to the colder reactor temperature at $t = 0$. The overall response is unacceptably slow and prevents practical implementation of the open-loop control strategy. In addition, the uniformity of the temperature on the wafer surface is not within an acceptable range (±3 K) at most times. Consequently, a feedback controller must be implemented to improve the performance.

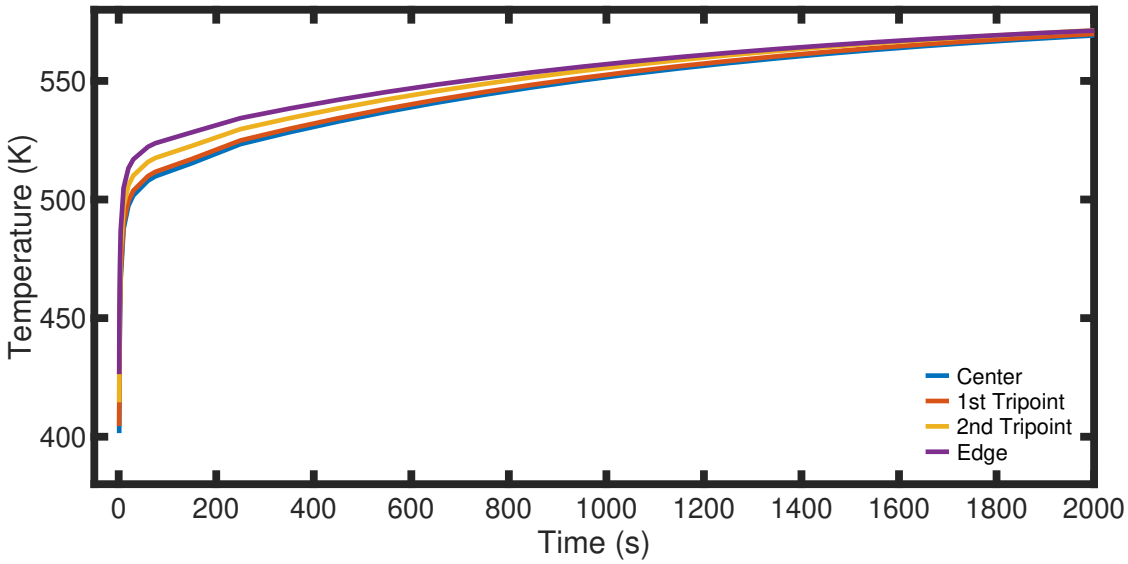


Figure 4.7: Temperature profiles with open-loop control strategy, where the lamp power is fixed at P^{ss}_{final}. Due to the low power input to the system at all times, the system needs over 2000 s to reach the target temperature, which implies that a controller is needed.

4.5.2 Closed-Loop Performance with Fixed Steady-State Power

In closed-loop control, the measured wafer surface temperature is fed to the controller, which is then used by the MPC to compute the optimal control actions to transmit to the lamp. In the current process, the controller development is challenging due to the changing environment in the reactor

and the use of reconstructed training data. Specifically, the cost function of Equation (4.16) constitutes of a state penalty, which is the first term, and an input penalty, which is the second term. We consider two cases for the steady-state power in the input penalty term, P_j^{ss}.

figs. 4.8 and 4.9 depict the closed-loop wafer temperature and lamp power profiles, respectively, under the MPC when using a fixed steady-state power of P_{final}^{ss} in the MPC cost function. From Figure 4.8, it can be observed that the wafer surface temperature is not uniform for most of the process duration as the difference between maximum and minimum temperatures measured on inspection points is over 6 K. This phenomenon is attributed to the low power of the center and edge lamps compared to the side lamps, as the ratio of $P_{final,1}^{ss}$, $P_{final,2}^{ss}$ to $P_{final,3}^{ss}$ is about 1 to 10, as shown in Figure 4.9. However, it is worth mentioning that the uniformity improves continuously with time as the reactor gradually heats up. The slow response is attributed to the fact that the final steady-state power, P_{final}^{ss}, calculated using a steady-state simulation, corresponds to the case where the whole reactor is thoroughly heated up to reach their steady-state temperature in conjunction with the wafer, which is far from the case for most of the process duration. This is because both the reactor walls and lamps start at room temperature (298 K) at $t = 0$ s, alongside the wafer. The input profile demonstrated in Figure 4.9 shows that the powers of the center and edge lamps, P_1 and P_2, are significantly lower than the power of the side lamp, P_3. The low center and edge lamp power results in the phenomenon in Figure 4.8, where the center temperature is much lower than the edge temperature, resulting in poor uniformity. Nevertheless, the closed-loop controller performance surpasses that of the open-loop control case, as the temperature of the closed-loop controller is much closer to the target temperature for any given time. Under the MPC, the temperature at two inspection points on the wafer reached a temperature of over 570 K at $t = 1000$ s, while, in open-

loop, all inspection points on the wafer are under 550 K at $t = 1000$ s. This case study demonstrates the effectiveness of an MPC over the open-loop control strategy, even if the convergence to the target temperature of 573 K occurs more slowly than desired. Additionally, an observable feature of Figure 4.8 and 4.9 the oscillatory-like behavior in particular times is originated from the numerical CFD simulation that produces converged solutions when residual tolerances are met. The resulting error generates chattering in the temperature profiles that simultaneously affects the MPC action. However, to prevent the drastic attenuation in the temperature rise after the first 20 seconds, next, we consider a second case of using a higher steady-state power in the input penalty term, P_0^{ss}.

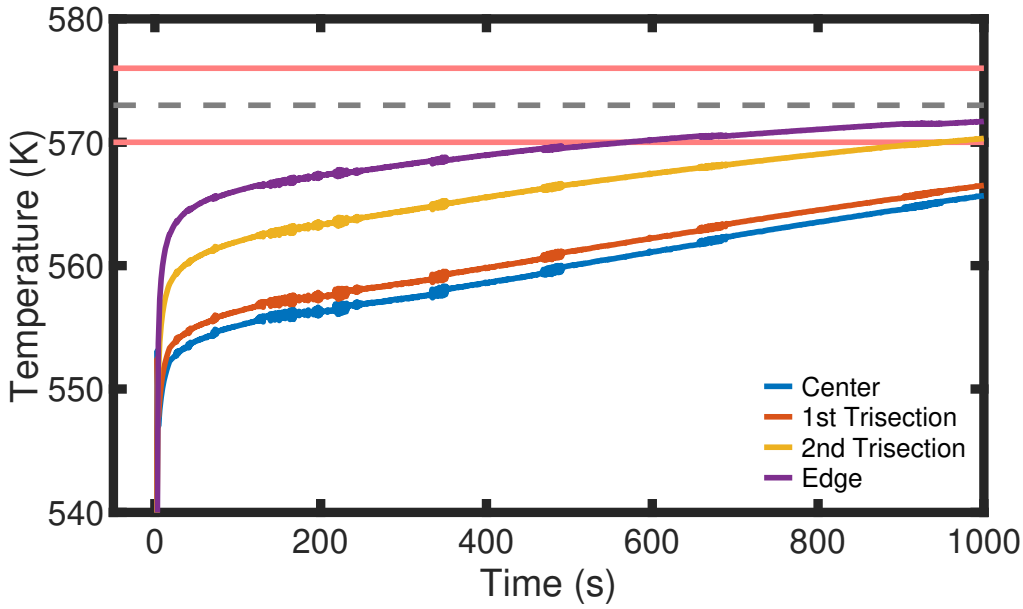


Figure 4.8: Temperature profile under closed-loop MPC with steady-state power in input penalty term fixed at P_{final}^{ss}. This performance surpasses open-loop control since the temperature is closer to target. The grey middle dashed line is the target temperature; the two red solid lines on the bottom and top are boundaries of control range around the target.

figs. 4.10 and 4.11 demonstrate the closed-loop state and input profiles, respectively, under MPC when using a fixed steady state power of P_0^{ss} in the cost function of the MPC. As expected, the

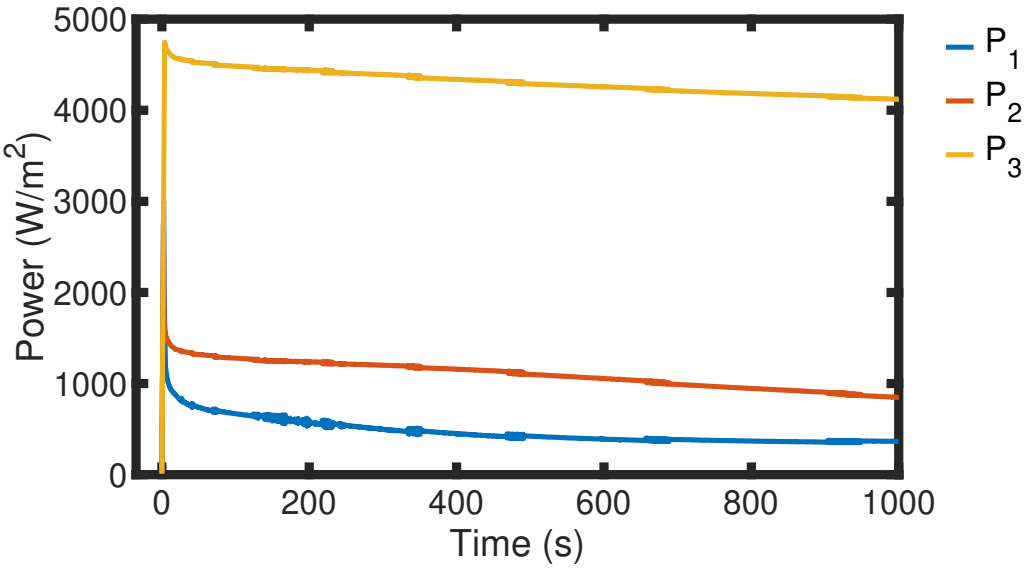


Figure 4.9: Lamp power profiles under closed-loop MPC with steady-state power in input penalty term fixed at P^{ss}_{final}. The low power for the center (P_1) and edge (P_2) lamps compared to the power of the side lamp (P_3) corresponds to the phenomenon that the center temperature is significantly lower than the edge temperature in Figure 4.8, which is attributed to a low $P^{ss}_{final,1}$ and $P^{ss}_{final,2}$.

controller with the high fixed steady-state power penalty cannot capture the target temperature and final steady-state. However, it significantly reduces the time it takes the wafer surface temperature to reach the target temperature. In addition, with the steady-state power set to P^{ss}_0, the ratios P^{ss}_1/P^{ss}_3 and P^{ss}_2/P^{ss}_3 are reduced to between 1 and 2, which causes the spatial uniformity in temperature to worsen as the system evolves, and the temperatures to vary by more than 6 K between inspection points soon after the initial rise. This phenomenon is reflected in the control actions in Figure 4.11, where the center lamp power is much higher than in the low fixed steady-state case. However, as the system continues to evolve, the high center-to-edge power ratio results in a much higher center temperature than the edge and also severe overshoot because of the high total lamp power.

By combining the observations from the low fixed steady-state power and the high fixed steady-state power cases, it can be proposed that a time-varying steady-state power is necessary to fulfill

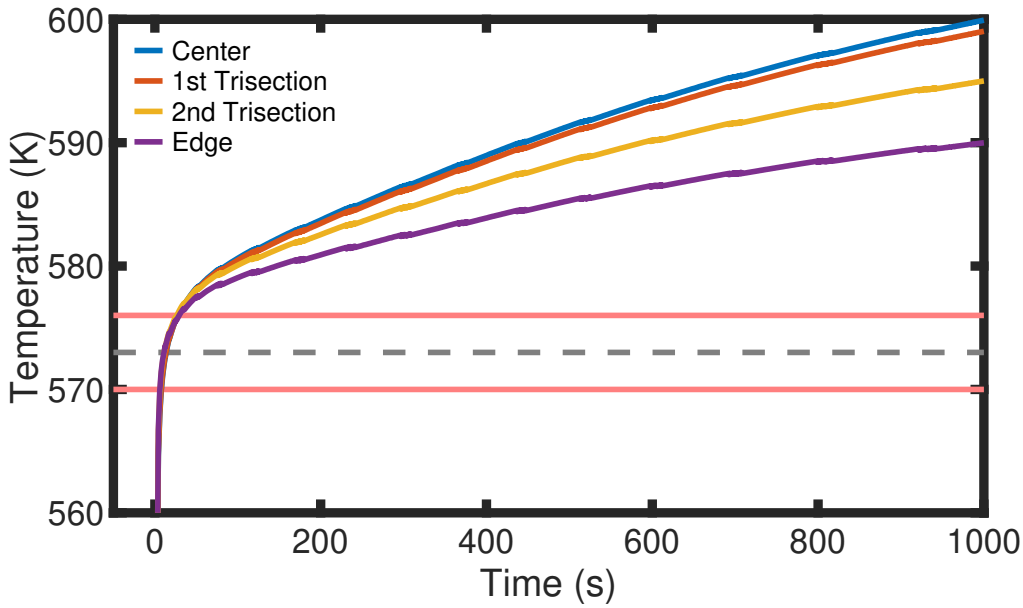


Figure 4.10: Temperature profile under closed-loop MPC with steady-state power in input penalty term fixed at P_0^{ss}. The plotting criterion is identical to that used in Figure 4.8. The system response is quicker, but severe overshoot occurs due to a higher total lamp power.

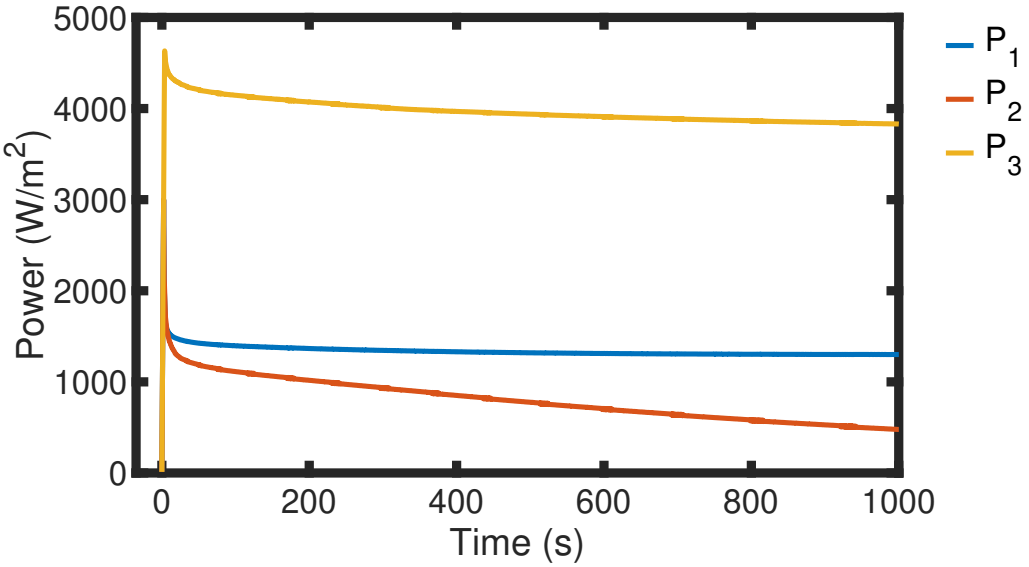


Figure 4.11: Lamp power profiles under closed-loop MPC with steady-state power in input penalty term fixed at P_0^{ss}. The center lamp power (P_1) is significantly higher than that observed in Figure 4.9 because $P_{0,1}^{ss}$ and $P_{0,2}^{ss}$ have a larger magnitude than $P_{final,1}^{ss}$ and $P_{final,2}^{ss}$, which results in a higher center temperature than the edge temperature in Figure 4.10.

the requirements of a fast response, small overshoot, and spatial temperature uniformity at all times.

4.5.3 Closed-Loop Performance with Feedback-Based Time-Varying Steady-State in MPC

The time-varying steady-steady power controller defined in Algorithm 1 has a temperature output as shown in Figure 4.12. Under the VSS approach, the temperature successfully reaches the target value of 573 K within 10 s, which is as fast as the high fixed steady-state power penalty case. After reaching the target temperature and overshooting a minor amount, which is below the upper threshold limit, the continuously decreasing steady-state power gradually drives the wafer surface temperature back toward the temperature setpoint. After the lowest measured temperature on the wafer reaches the lower temperature threshold of 571.0 K, the VSS controller stops decreasing the steady-state power and successfully moves the temperature back to target. The VSS controller then resumes decreasing the steady-state power when the highest wafer surface temperature reaches the upper temperature threshold of 574.5 K, and successfully brings the temperature back to the target. The control logic of the VSS controller is observed in the control action in Figure 4.13, where the inflection points are observed on lamp power curve when plotted against time, which indicates that the VSS algorithm does exhibit start-stop behavior when decreasing the steady-state lamp power as described above. Throughout the whole process, and especially after the initial stage, the temperature is well-controlled within the accepted region, and uniformity is also well-maintained. From fig. 4.12, it is seen that the VSS controller greatly surpasses the fixed steady-state controller in terms of controller performance.

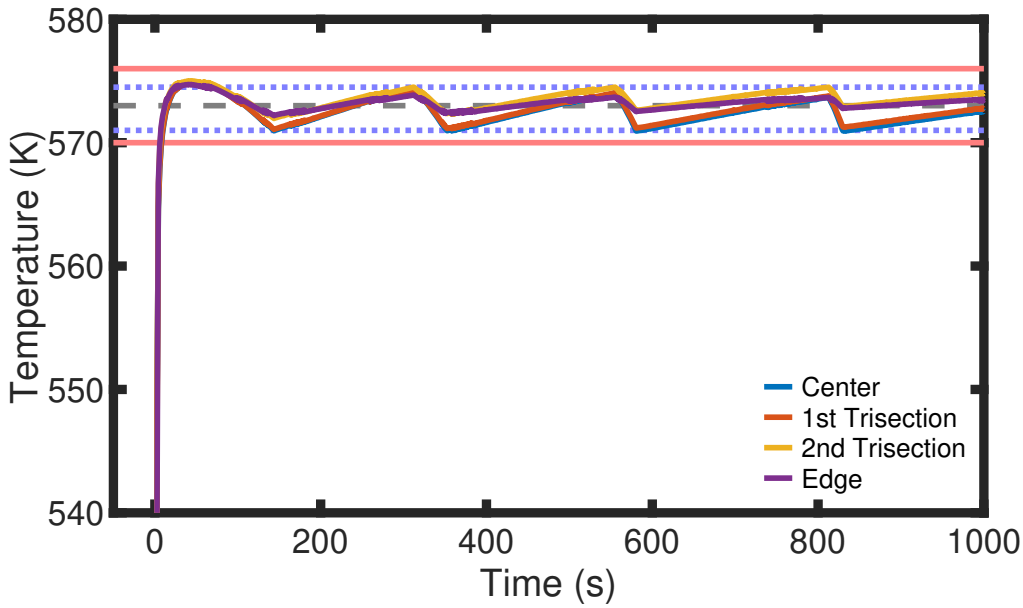


Figure 4.12: Temperature profile under closed-loop MPC with the feedback-based time-varying steady-state (VSS) power approach. The temperatures reach the target within 10 s and are kept within the acceptable range as desired. The blue dotted lines are the lower and upper temperature bounds (571.0 K, 574.5 K) for the VSS to stop or resume decreasing P^{ss} in the cost function of the MPC formulation.

4.6 Conclusion

In this chapter, we developed a 2-D transient energy model for a thermal atomic layer etching (ALE) reactor with a radiative lamp heating system that consists of three groups of independently controlled heating lamps. A data-driven dynamic model that predicts the system behavior is generated through sparse identification (SINDy) with an open-loop dataset consisting of 10 trajectories. A model predictive controller (MPC) that relies on the generated dynamic model is then applied to both the reactor and wafer, with the goal of driving the system from room temperature to the target temperature of 573 K. The open-loop controller strategy showed a slow response with poor temperature uniformity. An MPC was implemented with two settings for the input penalty term in the

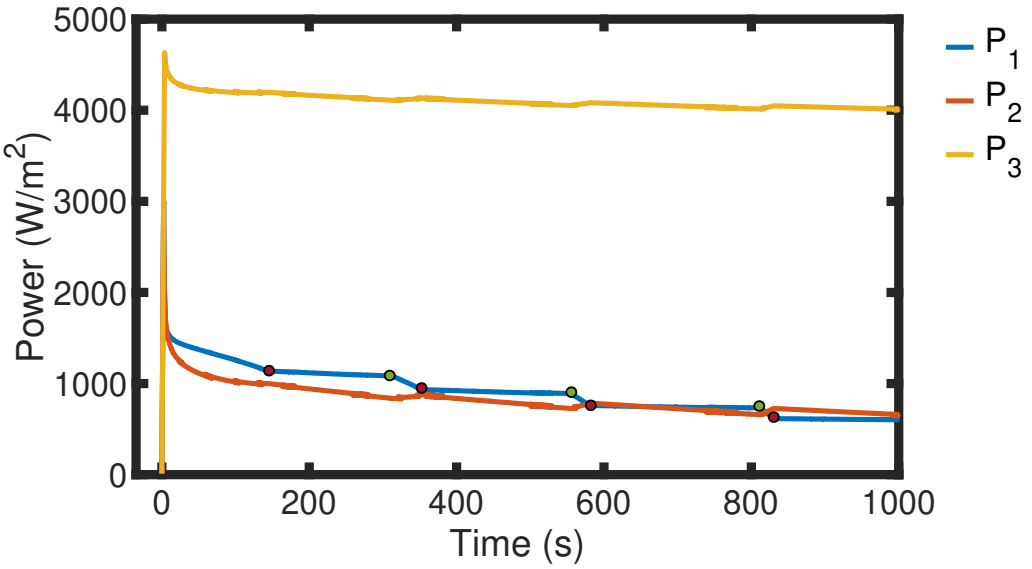


Figure 4.13: Lamp power profiles under closed-loop MPC with the feedback-based time-varying steady-state (VSS) power approach. The red points indicate that VSS stops decreasing P^{ss} in MPC cost function, while green points indicate resume decreasing.

objective function, one with a low fixed steady-state power and one with a high fixed steady-state power. While the former MPC resulted in a faster response and better uniformity than that of the open-loop, the latter produced a rapid response but overshot heavily. Therefore, a feedback-based time-varying steady-state (VSS) power penalty approach was proposed, where the MPC gradually decreases the steady-state power from an initial high value based on the wafer surface temperature and process conditions. The VSS approach successfully achieved a fast response and kept the temperature within an acceptable range for the entire duration of the process.

Chapter 5

Atomistic-Mesoscopic Modeling of Area-Selective Thermal Atomic Layer Deposition

5.1 Introduction

Currently, nanofabrication is highly dependent on top-down fabrication processes that consist of deposition, photolithography, and etching steps that are repeated to produce nanoscale devices. This conventional top-down fabrication approach, based on atomic layer deposition (ALD) and atomic layer etching (ALE), is conducive to highly conformal and ultra-thin film deposition, allowing for the construction of three-dimensional (3D) architectures in fin field-effect transistors (FinFETs) [91, 7]. Therefore, there has been a growing interest in improving and optimizing ALD processes with a plethora of research focused on this topic. ALD inherently relies on surface chemistry and ambient fluid conditions to exhibit self-limiting behavior, which is achieved by repeating

two or three reaction steps sequentially with an extended purge time between the reaction steps. ALD is a deposition process with precise thickness control capabilities with various materials, including metals, metal oxides, and organic molecules. Therefore, ALD is able to produce exceptionally uniform thin films with high-aspect ratios for a variety of materials such as hafnium oxide, HfO_2, titanium oxide, TiO_2, and aluminum oxide, Al_2O_3.

Despite the widely regarded capabilities of ALD, the semiconductor industry is facing many challenges with the continued down-scaling of nano-electronics. The primary bottleneck of the miniaturization process is the misalignment of features in the 3D architectures of the substrate while fabricating stacked, integrated circuits, which are known as edge placement errors (EPEs) [2, 92]. The conventional top-down fabrication process of the sub-5 *nm* channel uses a series of deposition, photolithography, and etching processes. During these processes, discrepancies between the desired position and the actual position emerge, causing ALD growth in undesirable areas [93]. Therefore, EPEs restrict how many well-aligned 3D structural layers can be stacked due to the misplacement between the top and the bottom layers. To address alignment challenges, area-selective atomic layer deposition (ASALD), as a bottom-up fabrication technique, has been proposed due to its potential to facilitate nano-electronic production in high-volume manufacturing [94, 10]. ASALD consists of an additional procedure that deactivates a target area, the so-called non-growth area (NGA), to prevent ALD growth on the NGA of the substrate surface while a subsequent process selectively deposits materials on the growth area (GA). This feature is achieved by chemically modifying the top surface of the NGA with inhibitors. In such a scenario, ASALD strongly depends on the chemical interactions between the substrate and of the precursors; specifically, there must be no interaction between the passivated NGA area and the deposited reagents intended for

the GA. ASALD has emerged as a possible remedy to the aforementioned issues associated with conventional ALD processes. Firstly, ASALD is able to reduce the need for additional, subsequent processing steps such as photolithography and etching applied in conventional ALD processes, which extensively reduces the overall manufacturing time and is capable of sustaining the growing demand for semiconductors, resulting in cost-effective fabrication while minimizing the toxic reagents used in photolithography and etching processes [94, 95]. However, the most attractable characteristic of ASALD when introduced into the industry, would allow for higher stacks on transistors to be produced and the potential to fabricate self-aligned nanoscale devices as mentioned above [92].

There are several approaches to creating a protective layer that allow for selective deposition. The most popular approach is to use self-assembled monomers (SAMs) as inhibitors to produce effective barriers. SAMs, such as alkylsilanes, have large aliphatic chains on their tails that block or delay ALD nucleation on the NGA while the head groups of the SAMs are connected to the NGA. Thus, SAMs act as growth inhibitors and allow film deposition to occur only on the GA of the substrate surface. The applications of ASALD using various SAMs have been investigated in several works [96, 97, 98, 99, 100]. Another advantage is that SAMs are readily removed by acetone and demineralized water after the ALD process is completed. Despite the aforementioned benefits of the SAMs, the ALD growth selectivity is limited to just a few nanometers of thickness due to potential defects in the SAMs. Furthermore, it is difficult and time-consuming to prepare defect-free SAMs [94]. In addition, most SAMs, which are based on wet chemistry, are difficult to integrate into vapor phase processes such as ALD. These issues make it challenging for SAM-based ASALD to yield high-aspect-ratio dielectric films.

To overcome the drawbacks of SAM-based ASALD, small-molecule inhibitors (SMIs) have been proposed. Unlike SAMs, which use a large inert functional group to physically block ALD growth, the reactive moieties of SMIs play a paramount role in preventing precursor adsorption onto the NGA [101]. ASALD based on SMIs effectively blocks precursors through a combination of chemical passivation and steric shielding [102]. The degradation of the blocking layer created by SMIs during the deposition processes is insignificant in contrast to that of the polymeric blocking layer of SAMs [103]. Since SMIs can be delivered through the vapor phase by adding a fixing step, the application of SMIs can be easily implemented to further improve selectivity. The protective layer formed via SMI-based ASALD allows surface destructive co-reactants such as oxygen plasma and ozone to be applicable as oxidants. An ABC-type area-selective ALD [2] has been developed based on this motivation and is also integrated into this chapter as discussed below.

The reaction mechanism for ASALD is composed of three sequential steps designated by Steps A, B, and C. In Step A, inhibitors selectively adsorb onto the NGA and form a blocking layer to prevent precursors from adsorbing onto the NGA during the entire ALD process. In Step B, precursors restrictively adsorb onto the GA due to the deactivation of the top layer of the NGA. Lastly, an oxidant oxidizes the top layer of the GA to complete a cycle of the deposition process. During the dosing of the precursor and oxidant, there is the possibility that some inhibitor molecules might desorb from the NGA, which contributes to a loss in selectivity. Thus, the correction step (Step A) is repeatedly introduced into the entire reaction cycle to re-saturate the protective layer. The ABC-type ASALD is schematically illustrated in Figure 5.1.

Despite the merits mentioned above, there is little computational or experimental research on the mesoscopic behavior of SMI-based ASALD. In particular, prior research [2, 103] demonstrated

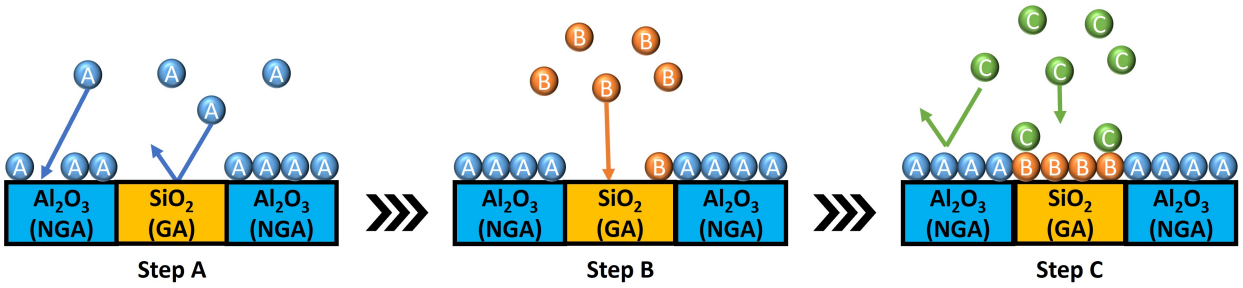


Figure 5.1: Schematic illustration of the ABC-type SMI-based ASALD. The molecules A, B, and C correspond to the inhibitor, precursor, and oxidant, respectively.

the technical viability of ASALD using SMIs, which was mainly performed from an experimental perspective. However, these previous works have not studied the surface kinetics for the entire cycle at the atomistic level in great detail. Thus, it is necessary to study how inhibitors selectively prohibit film growth on the NGA substrate with SMI-based ASALD in great detail. In this chapter, a kinetic Monte Carlo (kMC) study is performed to characterize ASALD of SiO_2 (GA)/Al_2O_3 (NGA) using small-molecule inhibitors (SMIs). Past studies have successfully employed the kMC method to simulate atomic layer etching processes [28, 104, 1, 12] to characterize the stochastic behavior of thin-film surface kinetics, which will similarly be integrated into this chapter. Acetylacetone (Hacac), bis(diethylamino)silane (BDEAS), and ozone (O_3) are selected as a SMI, precursor, and oxidant, respectively. Density functional theory (DFT) calculations are carried out to conduct an investigation into the atomistic behavior of all reaction steps. In addition, to visualize the steric effects of Hacac, a model that simulates the surface coverage for Hacac is developed. The structure of this chapter is as follows: Section 5.2 describes the methodology of mesoscopic modeling, Section 5.2.1 illustrates the ASALD process in details, Section 5.2.2 describes the calculations to define the atomistic behavior via DFT, Section 5.2.3 discusses the steric effects, Section 8.3 describes the results of the kMC study and its validation, and Section 8.4 provides a summary of this

chapter.

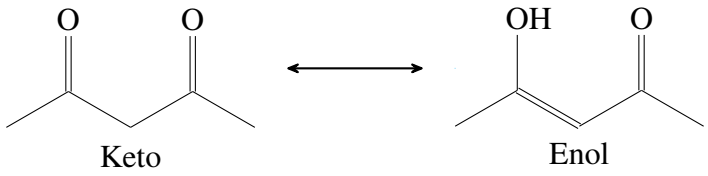
5.2 Atomistic-Mesoscopic Modeling

5.2.1 Surface Kinetics

The purpose of the ASALD process is to achieve a selective thin-film metal-oxide deposition that occurs solely on the growth area (GA) of β-SiO_2 (1 0 1) by integrating a protective surface of the non-growth area (NGA) of α-Al_2O_3 (0 0 0 1). The ASALD process consists of three sequential reactions (Steps A, B, and C) that are situated between purging steps, which are conducted in a cyclical manner. this chapter will also assume that all reactions are of bimolecular type, which only occur in elementary steps; thus, the reactions will proceed sequentially. Step A is composed of an inhibition-adsorption reaction in which the inhibitor selectively binds to Al_2O_3. Step B is a precursor modification reaction that uses bis(diethylamino)silane (BDEAS) to produce a modified surface layer on SiO_2, which possesses a self-limiting nature. Step C is an oxidation process that introduces ozone (O_3) onto the modified surface layer, thereby depositing a monolayer. These three sequential reactions are components of the so-called ABC-type reaction cycle, which is elucidated in this section.

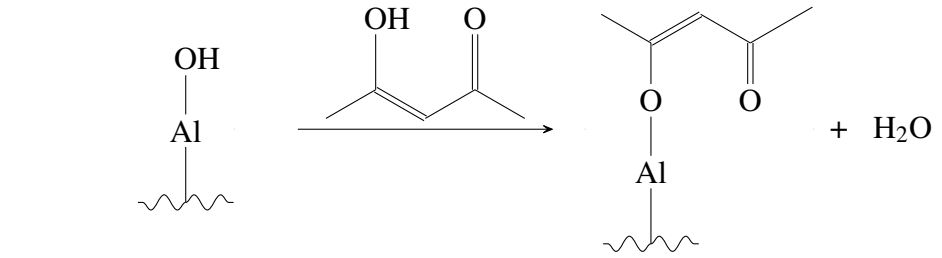
Inhibitors are designed to reduce the occurrence of a reaction, which makes inhibitor selection an issue of great importance for ASALD processes. Ideally, the inhibitor for an ASALD process must be chosen such that the inhibitor selectively reacts with the NGA; thus, this implies that there needs to be a substantial difference in the magnitude of the activation energy barriers between the NGA and GA, which is characterized by chemoselective behavior. Inhibitors must also have a low

decay rate so that the protective layer can sustain cycles of Steps B and C without requiring numerous doses of inhibition treatment between the deposition steps (Steps B and C). Thus, this chapter employs a small molecular inhibitor (SMI) with acetylacetone (Hacac) to fabricate an inhibition layer on the NGA of Al_2O_3, which exploits the conjugated structure of the tautomerized enol isomer to deprive the active oxygen sites on the NGA of subsequent adsorption reactions (Steps B and C). Hacac can exist in a keto isomer that can undergo a tautomerization process to produce a stable enol isomer composed of conjugated π bonds, which is the driving force that allows Hacac to bind to an Al atom on the NGA.

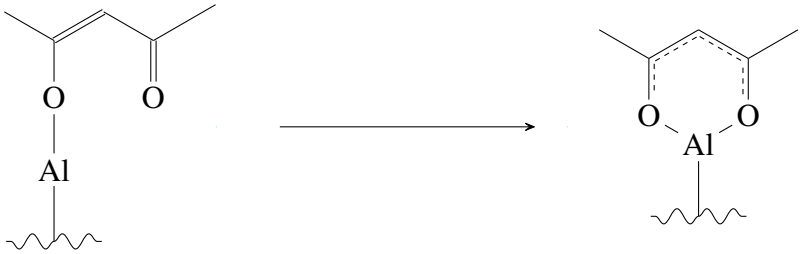


The keto and enol structural isomers that are formed through resonance allow the two resonance structures to coexist in basic and acidic forms, respectively. However, the stability of the enol form is more preferential and stable than the keto structure in the gas phase [105]. this chapter first characterizes the surface of Al_2O_3 as being uniformly composed of hydroxyl-terminated ligands (Al-OH) in a vicinal diol structure. This hydroxylation of the Al_2O_3 surface promotes basicity and interacts with the acidic Hacac, giving rise to a thermodynamically feasible neutralization reaction [2]. The reaction between the hydroxyl-terminated Al and Hacac molecule produces H_2O

vapor as a byproduct, which is defined in the following reaction:



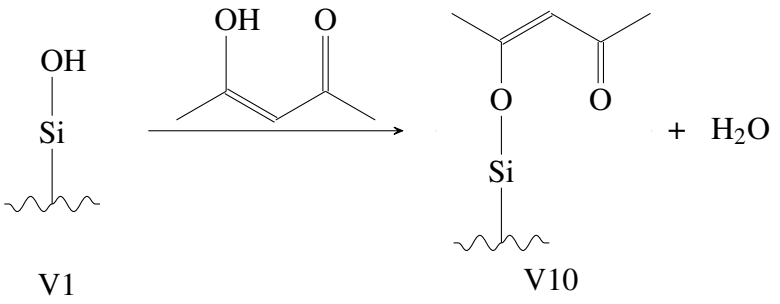
A1 Monodentate (A3) (Step A-1)



Monodentate (A3) Chelate (A4) (Step A-2)

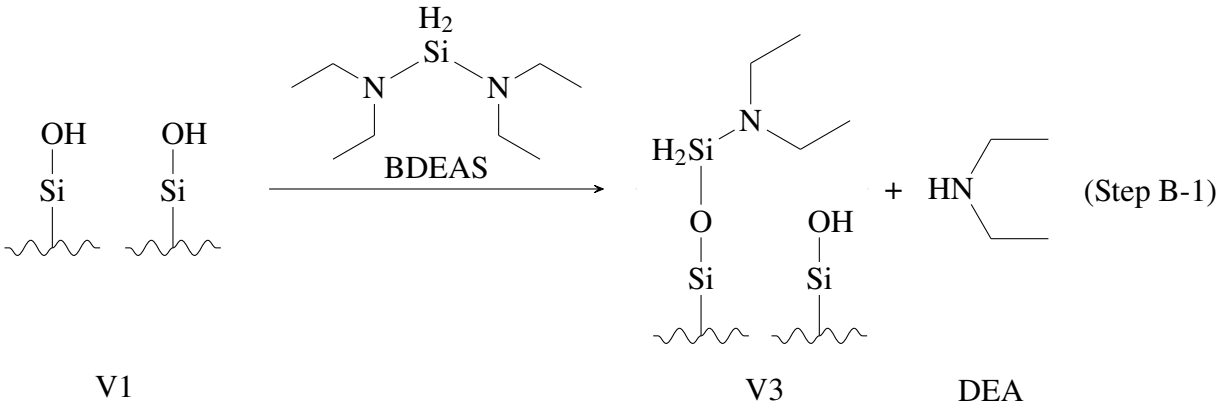
In Step A-1, an enol Hacac molecule withdraws its acidic hydrogen proton from the hydroxyl group, which is donated to the hydroxyl-terminated Al atom to produce a H_2O leaving group. The deprotonated enol Hacac molecule adsorbs onto the Al site to form a monodentate structure. Then, in Step A-2, the deprotonated Hacac adsorbate forms a protective six-member cyclic ring, also called its chelate form, through a chelation process by adsorbing onto the surface Al atom with the non-adsorbed keto group. The delocalized electrons in the conjugated π bonds of the chelate ring is illustrated with a dotted line [106]. Also, note that the wavy bonds from the Al atoms do not necessarily imply that the terminal is a single bond as shown in the reaction mechanisms; instead, they are used to represent the ligand structure that is unimportant to the mechanism and to illustrate that the reactions are occurring on the surface of the substrate. Although Hacac is intended to selectively adsorb only to Al-OH active sites, there is also potential for Hacac to adsorb onto Si-OH active sites

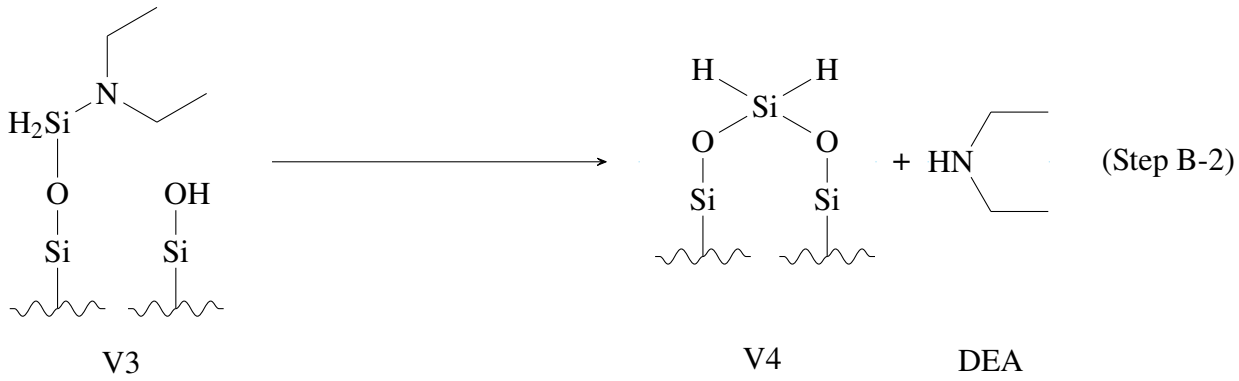
that may be generated during the hydroxylation step.



This undesirable reaction is the main source for nonuniform deposition in the SiO_2 growth region and can impede substrate quality and conformity by deflecting precursor adsorbates in Steps B and C of the ASALD process.

Precursors are intended to undergo surface reactions on the GA while having inherent properties that prevent further permeation beyond the substrate surface. Typically, bulky precursors are integrated into this modification step to ensure that a transport-limited boundary is established. The precursor, bis(diethylamino)silane (BDEAS), is used as an adsorbate on SiO_2 to produce Si lattice sites. The adsorption of BDEAS on the GA exemplifies self-limiting behavior such that single atomic monolayers of Si are deposited and byproducts are easily volatized. This Step B reaction mechanism is simplified by the following overall reaction:

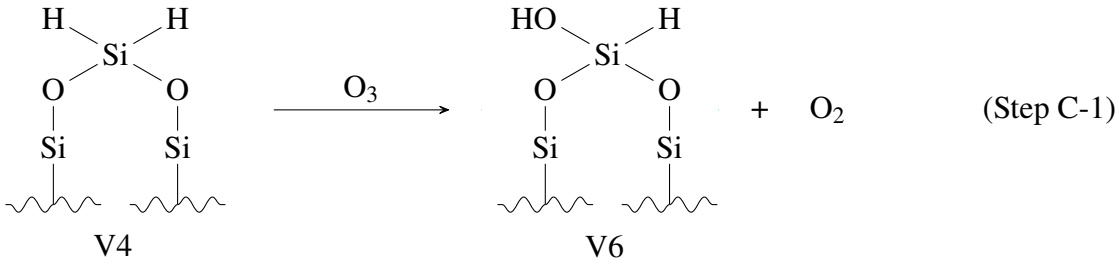


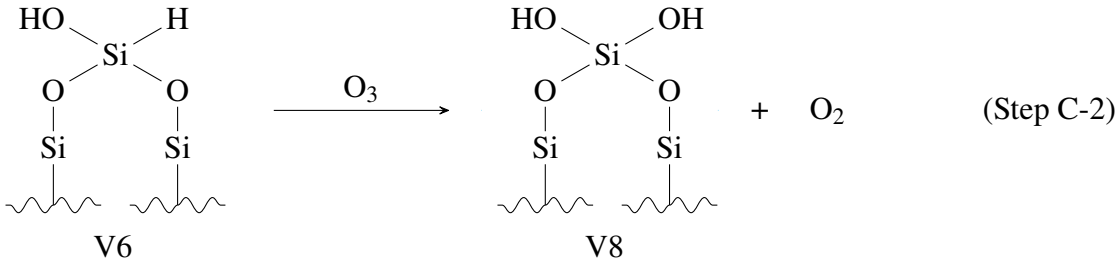


In Step B-1, BDEAS first adsorbs to an active hydroxyl surface site while simultaneously withdrawing a molecule of diethylamine (DEA). It is assumed that the hydroxylated SiO_2 surface consists of a homogeneous composition of vicinal hydroxyl ligands. Following the initial adsorption of BDEAS in Step B-2, the adsorbate binds to an adjacent hydroxyl surface site while simultaneously removing an additional DEA molecule, which results in a deposited monolayer of silicon atoms.

Following Step B, an oxidant, ozone, is introduced into the ASALD process to convert the hydride-terminated ligands of the newly deposited Si surface to hydroxyl-terminated ligands through an oxidation process. Ozone, which exhibits both electronegative and electropositive characteristics, performs a simultaneous and spontaneous ligand-exchange mechanism that substitutes a hydroxyl group for the hydride-terminated group while producing a leaving group of O_2, oxygen gas. This oxidation step is incredibly spontaneous to the point where the reaction can occur simultaneously for both hydride-terminated groups on the Si atom. The result completes the final monolayer of SiO_2 deposition and concludes the cycle of the ASALD process. The overall reaction for Step C

is described as follows:





The resulting surface layer becomes composed of a homogeneous array of geminal hydroxylated ligands that can be recrystallized through an annealing cycle after the ASALD process [107] to reform the initial vicinal diol surface structure. It is, however, worth mentioning that after each step (Steps A, B, and C) in the ASALD process, a purging cycle is introduced with an inert gas such as nitrogen gas, N_2, to ensure that the reagent species (Hacac, BDEAS, and O_3) do not intermix and disrupt the self-limiting behavior, which is essential to ensuring high surface uniformity.

5.2.2 DFT Calculation

Electronic Structure Calculation framework

With the reactions defined in Section 5.2.1, kinetic rate constants for each reaction must be defined for these elementary reaction steps. Mesoscopic simulation requires the knowledge of important reaction parameters to compute such reaction rate constants including the activation energies, the

pre-exponential factors, vibrational frequencies, and sticking coefficients. These parameters may be difficult to measure experimentally and generally require various computation methods that combine *ab initio* molecular dynamics, first principles quantum mechanics, and statistical chemistry. The open-source software for materials modeling, Quantum ESPRESSO (QE), is used in this chapter to calculate such kinetic parameters, which are needed to define the mesoscopic model. QE uses theoretical concepts including density functional theory (DFT), which computes the energy of phononic interactions using pseudopotential data that define each element by wavelengths to facilitate the computations [108, 109]. Previous works have employed QE for computing electronic structure optimization as well as for performing computations of kinetic parameters [110, 28, 1]. this chapter will integrate DFT computations via QE to compute an optimal (minimal electronic energy) electronic structure for each species produced in each step of the ASALD process.

The initial stage of the electronic structure optimization process consists of modeling the atomic structures using an open-source, graphical user interface (GUI) program that supports QE called BURAI. The surface crystal structures of α-Al_2O_3 and β-SiO_2 are constructed by defining a Bravais lattice of (0 0 0 1) [2] with a trigonal crystal and vicinal (1 0 1) [111] with a triclinic crystal structure, respectively. Also, lattice model unit cells for α-Al_2O_3 and β-SiO_2 are constrained within a $2 \times 2 \times 2$ supercell. Precursor and byproduct species including BDEAS, DEA, H_2O, O_3, and O_2 are developed using a free crystal lattice that is self-determined using a unit cell size that contains the entirety of the molecular structure. This method prevents divergence by defining boundary conditions for the "particle-in-a-box" model. Projector Augmented-Wave (PAW) pseudopotential data are used in conjunction with the Plane-Wave Self-Consistent Field (PWscf) package. In the program, the "relaxed" calculation method is utilized to optimize electronic forces that allow the

translation of atoms in space to generate an electronic structure of minimum electronic energy. The use of the aforementioned pseudopotential data facilitates the solving of the position and time-dependent Schrödinger equation, which is defined by the following expression:

$$\hat{H}\psi(\overrightarrow{x},t) = E\psi(\overrightarrow{x},t) \text{ where } \hat{H} = \frac{\hbar}{2m_e}\nabla^2 + \hat{V}(\overrightarrow{x},t) \tag{5.1}$$

where $\hat{H}$ is the Hamiltonian operator, which is a function of the sum of the kinetic and potential energies that are defined by the reduced Planck constant, $\hbar$, the particle mass, m_e, the Laplacian operator, ∇, and the potential energy, $\hat{V}$, respectively. Also, E refers to the total electronic energy of the system, and $\psi(\overrightarrow{x},t)$ is the wave function that depends on the displacement vector, $\overrightarrow{x}$, and time, t. A complete summary of the variables and their definitions used throughout this chapter is provided in Table 5.1. Self-determined parameters including the *degauss*, the kinetic energy cutoff for wave functions (*ecutwfc*), the kinetic energy cutoff for charge density and potential (*ecutrho*), and the k-points are determined through the SCF (self-consistent field) computation in QE and summarized in Table 5.2. The optimized electronic structures exported from QE and compiled in BURAI are visualized in Figure 5.2.

Computation of Kinetic Parameters

Various first principles theoretical concepts derived from quantum mechanics and chemistry are integrated into this chapter to compute reaction rate parameters that include the reaction rate constant, activation energy, and pre-exponential factor. The aforementioned parameters are fundamental for the mesoscopic modeling via the kinetic Monte Carlo method, which is discussed in Section 5.2.4. This section examines the methodological approaches to computing these kinetic parameters for

adsorption and nonadsorption reactions that are described in Section 5.2.1.

Adsorption reactions occur during the initial bombardment of the inhibitor (Step A), precursor (Step B), and oxidant (Step C) on the surfaces of the substrate. These adsorption reactions can be modeled as bimolecular reactions through Maxwell-Boltzmann statistics and Collision Theory (CT). The probability of a successful adsorption reaction is modeled based on a sticking coefficient factor, σ, which is deduced from experimental works. The following equation provides the definition of CT:

$$k_{ads} = \frac{PA_{site}\sigma}{Z\sqrt{2\pi m k_B T}} \tag{5.2}$$

where k_{ads} is the reaction rate constant for the adsorption reaction, P is the pressure of the gaseous reagent (Hacac, BDEAS, or O_3), A_{site} is the surface area of a single active site, Z is the coordination number of the gas, m is the atomic mass of the gas, k_B is the Boltzmann constant, and T is the absolute temperature of the ambient environment. this chapter defines a sticking coefficient, σ, for Hacac as 1.0×10^{-4} [7], BDEAS as 2.0×10^{-5} [112], and O_3 as 4.5×10^{-5} [113]. It is noteworthy that the sticking coefficient of Hacac was determined from surface adsorption using precursor species for ALD and was assumed to be similar in magnitude to that of Hacac adsorption onto Al_2O_3.

A majority of reactions discussed in this chapter involve chemisorption, desorption, and other surface reactions that cannot be computed via CT. Nonadsorption reaction rate constants can be calculated using the Arrhenius equation, which is defined by the following equation:

$$k_{nonad} = \nu \exp\left(-\frac{E_A}{RT}\right) \tag{5.3}$$

where k_{nonad} is the reaction rate constant for the nonadsorption reaction, ν is the pre-exponential

factor, E_A is the activation energy, R is the universal gas constant, and T is the absolute temperature of the reaction. One challenge arises with the computation of the pre-exponential factor term, which is modeled on the frequency of collisions where the reactants overcome the activation energy barrier. Thus, Transition State Theory (TST) is often employed to reduce the computational requirements to collect such frequency data and is described by the following expression that computes the pre-exponential factor:

$$\nu = \frac{k_B T}{h} \frac{Q^{\ddagger}}{Q} \tag{5.4}$$

and $Q^{\ddagger}$ and Q are the products of the electronic, rotational, translational, and vibrational partition functions for the transition state species and reactant, respectively. To reduce the computational requirements for evaluating the partition functions, the ratios between the transition state and reactant partition functions are often simplified to unity [114]. Thus, the pre-exponential factor is largely dependent on the temperature of the reaction. Lastly, the activation energy of the reactions should also be established in Eq. (5.3), which are computed using the nudged elastic band (NEB) method in this chapter. QE contains a NEB package that is designated for carrying out such computations that essentially construct a minimum energy path between reactants and products. The NEB method requires optimized structures, which are obtained from the PWscf package described in Section 5.2.2. With the optimized structure data, the NEB computation requests a self-determined number of electronic steps that computes an energy for each step to ultimately determine the activation energy barrier. this chapter defines seven electronic steps to reduce the computational demand while establishing a robust procedure for computing an accurate activation energy barrier to establish the reaction rate constants. Figure 5.2 illustrates the results produced from the application of

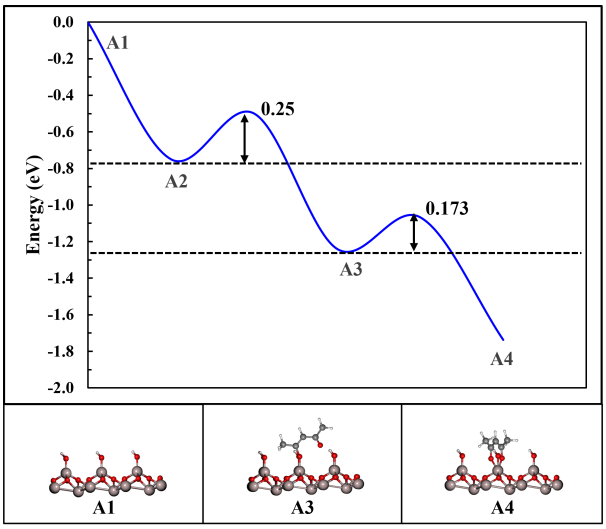
ab initio and first principles modeling concepts as well as simulation results generated from QE. The activation energies essential for the mesoscopic model are shown in Figure 5.2. It is notable that the energy on the *y*-axis does not denote an absolute energy, but rather the energy difference between the starting structure and the ending structure in each step. Geometries for the reaction paths on Al_2O_3 and on SiO_2 are numbered as **A1** through **A4** and **V1** through **V10**, respectively, and they are summarized in Table 5.3.

5.2.3 Steric Hindrance

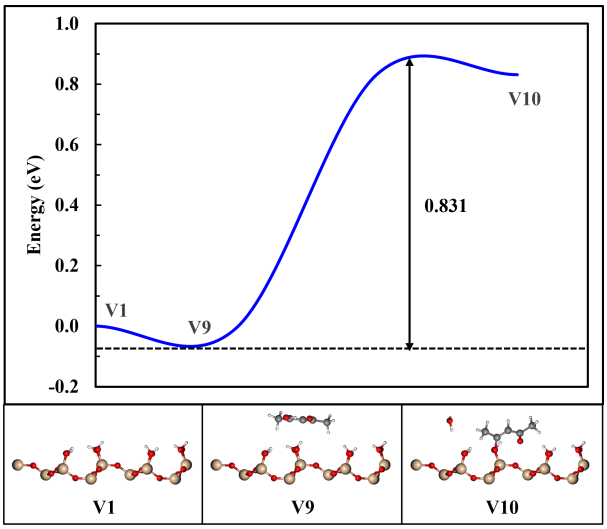
Fundamentals of Steric Effects

Steric effects play a substantial role in the frequency of adsorption reactions discussed above by blocking other molecules from approaching the surface, thus hindering specific surface reactions. The regioselective adsorption of the SMI, Hacac, and bulky precursor, BDEAS, depends on how much space is available for these reactions to occur. In other words, Hacac and BDEAS cannot be densely packed on the surface due to their large molecular size, and the bulky adsorbates that have already bonded to the surface can prevent surrounding molecules from interacting with the substrate surface because of steric effects. Thus, the repulsive effect that results from the steric hindrance of the bulky reagents used in ASALD must be investigated and accounted for to accurately depict the kinetic behavior of the surface reactions. The computational methods used to simulate the steric effects in conjunction with the kinetic Monte Carlo model are elucidated in this section.

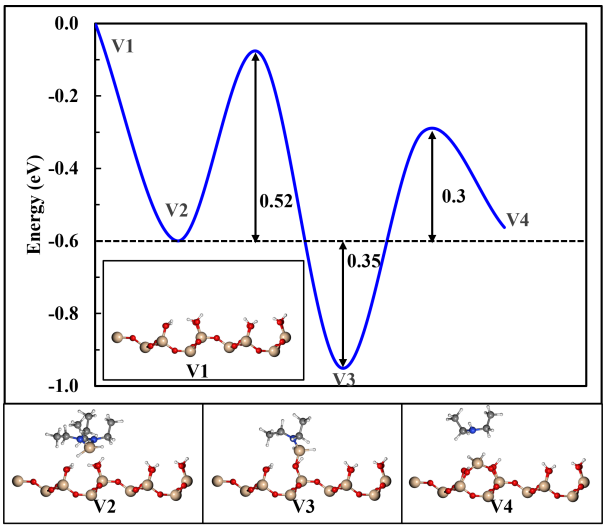
There are several studies that have reproduced the repulsive effects of steric hindrance in surface adsorption reactions through computational approaches [115, 116, 117, 118]. For example, [115] considered the molecular interactions of Si-S, Si-Se, and Si-Te bonds between the



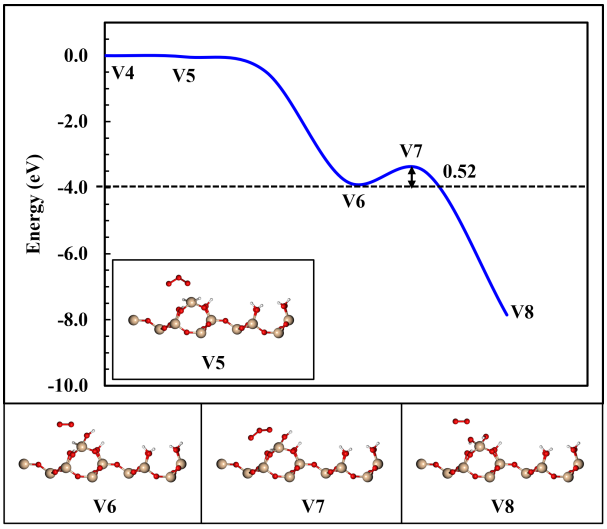
(a) Hacac cycle on Al_2O_3



(b) Hacac cycle on SiO_2



(c) BDEAS cycle on SiO_2



(d) O_3 cycle on SiO_2

Figure 5.2: Minimum energy paths from the DFT calculations for the Hacac adsorption on (a) Al_2O_3 and (b) SiO_2, for the BDEAS adsorption on (c) SiO_2, and for the O_3 adsorption on (d) SiO_2. Color code for atoms: aluminum, dark gray; silicon, light brown; oxygen, red; hydrogen, white; nitrogen, dark blue; carbon, gray. The paths for the Hacac adsorption from **A1** to **A3** was sourced from [2]. Specifically, **A2**, **V2**, **V5** and **V7**, and **V9** indicate the physisorption reactions for Hacac on the NGA, BDEAS on the GA, ozone on the GA, and Hacac on the GA, respectively.

substrate and bulky organic functional ligands in adsorption reactions, which were simulated in a two-dimensional (2D) stochastic model. In this model, the adsorbates were represented by disks

defined by their van der Waals radii, which are defined as the minimum approach distance between two atoms that are represented as solid spheres. The van der Waals radii of some organic functional groups are presented in [119], which are applied in the simulation for Step A and summarized in Table 5.4. This section will discuss the steric hindrance involved in the Hacac adsorption on Al_2O_3 in Step A and the BDEAS adsorption on SiO_2 in Step B (due to the small van der Waals radius of ozone, steric hindrance effects are not expected in Step C). In regard with the BDEAS adsorption, a detailed review of the open literature indicates that it is difficult to find either experimental or simulation studies that discuss the steric hindrance effects taking place in the ALD process with BDEAS as precursor. Prior work presented important results from macroscopic measurements and observations [111, 2], but there is no mesoscopic explanation for the phenomena from a steric hindrance perspective. Thus, this surface modeling work accounting for steric effects provides a potential approach to reveal how steric hindrance influences the ALD nucleation of BDEAS.

Hacac Adsorption on Al_2O_3

Hacac is a bulky molecule compared to the active reaction sites on α-Al_2O_3, which are defined as OH ligands. Thus, the deprotonated Hacac adsorbates on the Al_2O_3 surface can block other Hacac molecules from adsorbing onto neighboring reaction sites. There are two molecular configurations for deprotonated Hacac that has adsorbed onto Al_2O_3: monodentate (**A3**) and chelate (**A4**), as shown in Figure 5.2a and described in Table 5.3. The reaction (Step A-2) from the monodentate to the chelate configuration is exothermic and has a low activation energy barrier as illustrated in Figure 5.2a, which implies that the monodentate ligand can easily overcome the activation energy barrier of 0.173 *eV* and transform into the chelate molecule through a chelation process. As a

result, the monodentate structure spontaneously converts to the chelate structure from a kinetics perspective, which is in agreement with [2]. However, [103] reported that part of deprotonated Hacac exists as monodentates on the Al_2O_3 surface through IR inspection.

To investigate the Hacac adsorption process in greater detail, this chapter constructs a 2D model with a 100×100 surface grid, which has been proposed by [115]. In the grid, all active sites, denoted by small black dots in Figure 5.7, are randomly searched to determine their possibility for Hacac adsorption. In the simulation, the chelate adsorption is performed first and then followed by the monodentate adsorption due to the difference in magnitude between the sizes of the two molecules. A horizontal chelate adsorbate exists on the surface as revealed in Figure 5.3. The chelate structure has two methyl, CH_3, functional groups with a van der Waals radius of 2.0 Å and a spacing of 4.8 Å between the two functional groups, which is denoted as an empty capsule with two circles in Figure 5.7. Conversely, a monodentate adsorbate is placed diagonally on the surface as shown in Figure 5.3. To simulate the adsorption of deprotonated Hacac monodentates onto the 2D grid, it is assumed that the CH_3 functional group does not affect other adsorption reactions at adjacent reaction sites due to its high position. Instead, the free CO functional group is solely responsible for the steric repulsions of other molecules. Therefore, in the 2D model, monodentate structures have CH_3 and CO functional groups with van der Waals radii of 2.0 and 1.7 Å, respectively, and a spacing of 3.8 Å; they are depicted as two conjoined circles in blue in Figure 5.7. The simulation procedure is as follows:

1. The chelate adsorbate is placed on the site throughout the grid if there is no steric hindrance. The orientation of the adsorbate is stochastically selected from 360° with intervals of 10°.

2. Next, the monodentate adsorption selection is performed, filling the empty sites with adsorbates in viable orientations in the same manner with the chelate adsorption.

Even after Step 1 (chelate adsorption) is performed, vacant active sites still exist due to the steric hindrance of the chelate configurations. Thus, Hacac molecules can nonetheless still adsorb onto the surface in the monodentate configuration. To reduce large deviations attributable to the stochastic nature of the algorithm, 100 simulations are carried out, and the surface densities are averaged. In addition, the standard deviation is calculated from the set of computations to evaluate the developed model for steric effects.

A prior study developed a 2D surface model to investigate the effects of steric hindrance on Hacac adsorption on Al_2O_3 [103]. The surface model simulated chelate and physisorbed monodentate configurations; however, the model lacked chemisorbed monodentate products on the active reaction sites. To advance the model proposed by [103] and to obtain the final surface structure, both chemisorbed chelate and monodentate molecules are considered in this research. This improvement enables the model to obtain more precise and realistic surface information, which provides a deeper explanation of the hinderance effect. The approach of this chapter, which is adopted from [115] and advanced from [103], can be extended to other surface kinetics in which the surface can be approximated into a 2D lattice model simply with the calculations of van der Waals radii of adsorbed molecules. The simulation results from the surface model of the Step A (Hacac) cycle are discussed in Section 5.3.1. After the simulation, the molecular density on the surface is calculated by Eq. (5.5).

$$\rho = \frac{n}{D^2 \cdot s \cdot x} \tag{5.5}$$

where ρ is the molecular density in $molecules/nm^2$, n is the number of molecules on the 100×100 grid, D is the dimension of the grid, which is 100 in this simulation, s is the distance between two adjacent sites, and x is the distance between two sites in the horizontal direction, which is calculated by the following equation:

$$x = \frac{\sqrt{3}}{2} s \tag{5.6}$$

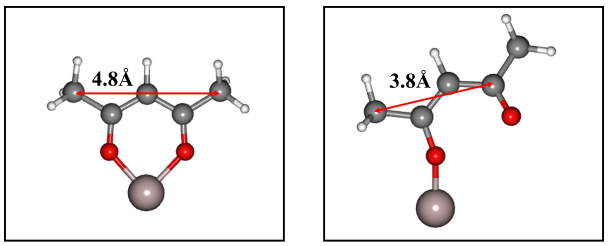


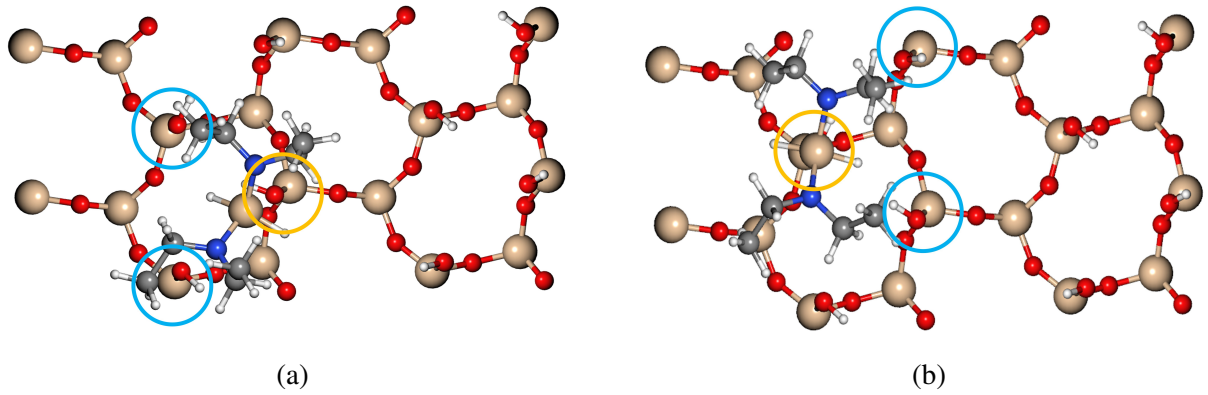
Figure 5.3: The intervals between two functional groups of the chelate and the monodentate are 4.8 and 3.8 Å, respectively. For the monodentate configuration, the CO group is considered as the group that is potentially able to hinder neighboring adsorption reactions. It is assumed that the CH_3 group at the far right of the monodentate product does not contribute to steric effects on adjacent reaction sites. Color code for atoms: aluminum, dark gray; oxygen, red; hydrogen, white; carbon, gray.

BDEAS Adsorption on SiO_2

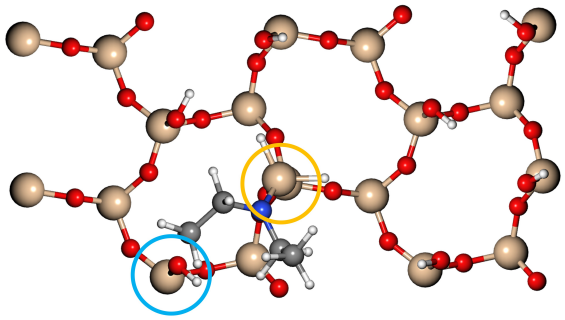
A BDEAS adsorbate is capable of preventing other BDEAS molecules from adsorbing on the GA, SiO_2. Figure 5.5 illustrates the top view of the GA surface, which is expressed by Geometry **V1** in Table 5.3, and it consists of 8 active reaction sites, which are shown as OH functional groups. If a BDEAS molecule adsorbs onto one of these sites (Geometry **V2**), the two functional groups of diethylamine (DEA) may protect the two neighboring reaction sites from the physisorption of other BDEAS molecules as shown in Figures 5.4a and 5.4b. When the physisorbed BDEAS reacts with the OH group by releasing H_2O, the BDEAS molecule turns into a DEA-deprived BDEAS

(Geometry **V3**) on the GA as illustrated in Figure 5.4c. However, the remaining DEA group may block an adjacent site so that additional BDEAS physisorptions on the blocked site are hindered. When forming SiH_2 (Geometry **V4** in Table 5.3) by discharging the DEA group from Geometry **V3**, the adsorbate must bind to two adjacent sites on the surface. There are 5 potential reaction sites from which SiH_2 can be formed on the GA. In Figure 5.5, the distances between Sites 1 and the 4 adjacent reaction sites, denoted as Sites 2 through 5, are calculated from the optimized structure computed through electronic structure calculations. The distances between Sites 1 and 2, Sites 1 and 3, Sites 1 and 4, and Sites 1 and 5 are 5.03, 4.01, 4.02, 6.00, and 5.92 Å, respectively. It is reasonable to assume that Site 1 preferentially binds with Sites 3 and 4 to deposit SiH_2 films (Geometry **V4**) due to these sites being the closest, leading to those binding reactions to have the lowest activation energy. Therefore, the reactions between Sites 1 and 2, Sites 1 and 5, and Sites 1 and 6 can be reasonably ignored in the kinetic Monte Carlo (kMC) simulation. According to the periodical and symmetrical nature of the surface grid, Site 3 can only react with Sites 1 or 2. In other words, the reactions are constrained in the "bicolumn" system from a modeling perspective.

In addition to the steric effects, another issue to consider is that some reaction sites can be deactivated through isolation effects when two adjoining sites are already occupied. As a result, it is nearly impossible to reach full coverage for Step B (BDEAS cycle on the GA). In the experimental data of the BDEAS adsorption from [111], coverages of 94% and 86% were observed both within and outside the studied ALD window, respectively. In this mesoscopic model, the deactivated sites are ignored when calculating the surface coverage in the kMC simulation so that the simulation stops appropriately. With this modification, the computation model can reach 100% coverage by disabling the deactivated sites. On the basis of these steric and kinetic constraints, the kinetic



(a) (b)



(c)

Figure 5.4: Top view of (a) Case 1 for Geometry **V2** (b) Case 2 for Geometry **V2** (c) Geometry **V3**. The blue circled reaction sites are hindered and deactivated by the current reaction site in yellow. Color code for atoms: silicon, light brown; oxygen, red; hydrogen, white; nitrogen, dark blue; carbon, gray.

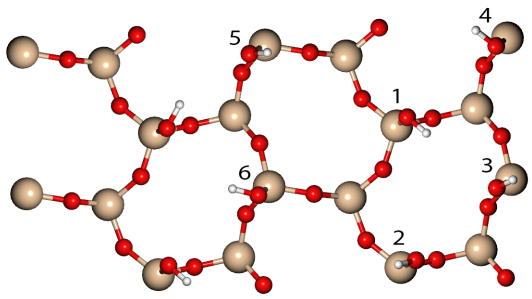


Figure 5.5: Top view of a OH terminated SiO_2 surface, expressed by Geometry **V1** in Table 5.3. Sites 2 through 5 are numbered with Site 1 as the center. Color code for atoms: silicon, light brown; oxygen, red; hydrogen, white.

Monte Carlo (kMC) model is built to simulate the surface kinetics, including the blocking effect of adsorbed molecules and the interactions between reaction sites. The detailed description of the kMC model is discussed in Section 5.2.4 in depth.

5.2.4 Kinetic Monte Carlo Simulation

The conformity of the substrate is dictated by various quality specifications including the uniformity and the amount of deposition on the growth area (GA). However, the latter presents a challenge for modeling in the mesoscopic surface domain due to the disorganization of the reaction kinetics. While it is theoretically possible to simulate the complete reaction mechanism on the surface of a 200 or 300 mm substrate, such a task is extremely computationally demanding to the point of infeasibility. As a popular approach to simulate surface kinetics at the mesoscopic level, statistical Monte Carlo (MC) methods use random sampling in a constrained domain to accurately estimate the effects of a much larger stochastic system. Such atomistic-mesoscopic simulation using MC methods is applicable to nanoscale material systems and minimizes the variability of timescale prediction as described by [120]. These algorithms are the most reliable when applied to complex simulations that maintain their probability distribution. Additionally, the kinetics of multiple reaction systems will evolve as the system progresses; thus, there must also be a temporal element to the algorithm. The reaction systems occur at predetermined reaction sites on the surface of the wafer substrate, which indicates that the kinetic Monte Carlo (kMC) simulation method is an appropriate algorithm to integrate into this chapter [114]. There are various kinetic Monte Carlo (kMC) algorithms that have been investigated; however, this chapter applies the variable step size method (VSSM), also called the Gillespie algorithm, which was advanced by Bortz, Kalos, and

Lebowitz [121]. This algorithm is ideal because it is more computationally efficient when there are numerous reaction pathways [122], which is exactly the case for the ASALD reactions. The VSSM method converts the surface structure of the substrate into a matrix of identifiable numbers, which is illustrated in Figure 5.6. The basic formulation of the VSSM algorithm can be separated into 5 steps as described in [28, 122]: 1) identification of possible reactions, 2) rate constant summation, 3) reaction selection, 4) time step evolution, and 5) process continuation.

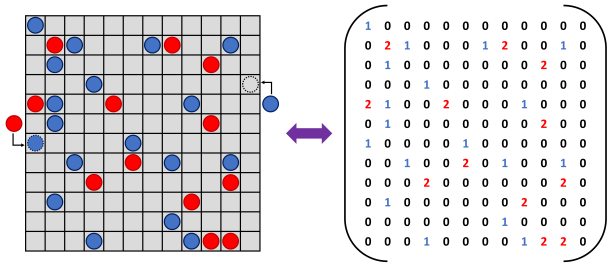


Figure 5.6: Conversion of the mesoscopic surface kinetics into a matrix of identifiable numbers. The blue and red circles in the lattice, corresponding to 1 and 2 in the matrix, indicate different reaction statuses.

The first step of the VSSM algorithm is to identify all possible reactions for each active site, which is identified by conditions in the simulation. Next, all the possible reaction rate constants are summed, which establishes the bounded conditions for the kMC simulation and is calculated from the following equation:

$$k_{total} = \sum_{i-1}^{N} k_i \tag{5.7}$$

where k_{total} is the sum of all reaction rate constants, k_i is the reaction rate constant for the potential reaction i, which was evaluated in Section 5.2.2, and N is the total number of possible reaction pathways. Secondly, the algorithm employs a randomly generated number, $\gamma_1 \in (0, 1]$, to stochastically

select a reaction, j, that fulfills the following criterion:

$$\sum_{i=1}^{j-1} k_i \leq \gamma_1 k_{total} \leq \sum_{i=1}^{j} k_i \tag{5.8}$$

where j is the index of the reaction in N that satisfies the equation. As a reaction path is determined depending on the product of γ_1 and k_{total}, it is observed that the larger the individual rate constant, the more likely it is to be chosen. Of note, the random number is computed using the Python "*random*()" function from the *random* package. Then, the following algorithm is used to determine a time interval in which no reactions occur [114].

$$\Delta t = \frac{-\ln \gamma_2}{k_{total}} \tag{5.9}$$

where Δt is the aforementioned time interval and $\gamma_2 \in (0, 1]$ is another randomly generated number between 0 and 1 that is independent of γ_1. Finally, the simulation is conditionalized with an ending criterion to determine the continuation or termination of the simulation. If the former is chosen, the algorithm is repeated starting from the first step of the VSSM algorithm as the time progresses from $t_{old} \rightarrow t_{new} + \Delta t$. In addition, k_{total} is recalculated with each iteration of the algorithm since the possible reaction pathways across the substrate surface change as the system evolves. For the simulation of ASALD, the kMC simulation is used to estimate the time necessary for the surface of the wafer substrate to reach complete, 100%, coverage; thus, the terminal condition for the simulation is defined for the latter. The simulation is conducted using Python scripting language in serial processing and various packages dedicated to the randomization of the simulation and for carrying out numerical computations.

For generality, the procedural steps of the atomistic *ab initio* quantum mechanics and mesoscopic simulations are summarized as follows:

1. Elementary reaction pathways are defined and limited to rate-determining reactions from the electronic structure calculations of the surface structures.

2. *Ab initio* quantum mechanics simulations are simulated via Quantum ESPRESSO for individual species generated for surface reactions.

3. Reaction rate constants are calculated using CT and the Arrhenius Equation for adsorption and nonadsorption reactions, respectively.

4. The kMC method considering hinderance effects is exploited following the characterization of the kinetics parameters.

5.3 Simulation Results and Discussion

In this chapter, surface modeling results for the simulation of Hacac adsorption on the NGA, described in Section 5.2.3, are discussed. In addition to the surface modeling, the temperature and pressure dependence of the area-selective atomic layer deposition (ALD) of SiO_2/Al_2O_3 are investigated in a pressure range of 10 to 500 Pa with a temperature range of 423 to 573 K. As mentioned above, SiO_2 is defined as the growth area (GA); on the other hand, Al_2O_3 is considered to be the non-growth area (NGA) where the film deposition is not required. It is also noted that the ALD process has a self-limiting behavior in that only a single layer of SiO_2 is deposited on the top layer of the surface. Thus, the kinetic Monte Carlo (kMC) simulations considering steric effects are performed

until each step in the ABC cycle reaches full coverage. It is noted that the kinetic interaction between SiO_2 and Al_2O_3 on the boundary is ignored in this study as described in Section 5.2.4. Thus, two separate lattices for SiO_2 and Al_2O_3 of 100×100 are simulated for surface kinetics. A number of simulations with a lattice size of 100 through 900 were performed to explore size-dependence of the kMC simulation. It was observed that there was no significant effect of the lattice size on the simulation results while requiring higher computing power with increasing lattice size, which is also supported by previous studies [123, 28]. To minimize the effect of stochastic sampling, 100 kMC computations are performed and averaged to obtain a data point for the process time under the operating conditions. To provide a benchmark, the computation time for the entire cycle is about 30 *min* for 100 trials using a constant temperature of 523 K and pressure of 300 Pa, while simulated using similar computational resources.

5.3.1 Surface Modeling for Protective layer

Figure 5.7 visualizes the surface of Al_2O_3 where chelate and monodentate products are randomly distributed at different angles. Due to the small molecular size, chelate molecules occupy the space between monodentate molecules. The surface density of each chelate and monodentate adsorbate is 1.45 ± 0.02 $molecules/nm^2$, and 0.35 ± 0.03 $molecules/nm^2$, respectively. 1.80 ± 0.01 $molecules/nm^2$ is thus calculated to be the combined surface density from the simulation. The fraction of monodentate configurations on the surface is 19.37 ± 1.40%. The reported experimental value of the monodentate fraction on the surface by IR inspection is 20 ± 5% [103], and thus, the simulation result conforms with the experimental value. However, the molecular density simulated from the model developed in this chapter is lower than the density from [103] for the

chelate configuration, which is 1.7 ± 0.1 $molecules/nm^2$. The deviation of the molecular density is attributed to differences in the atomic distances calculated in Section 5.2.2 using Quantum ESPRESSO and the atomic distances evaluated by [103] who reported a lower atomic distance.

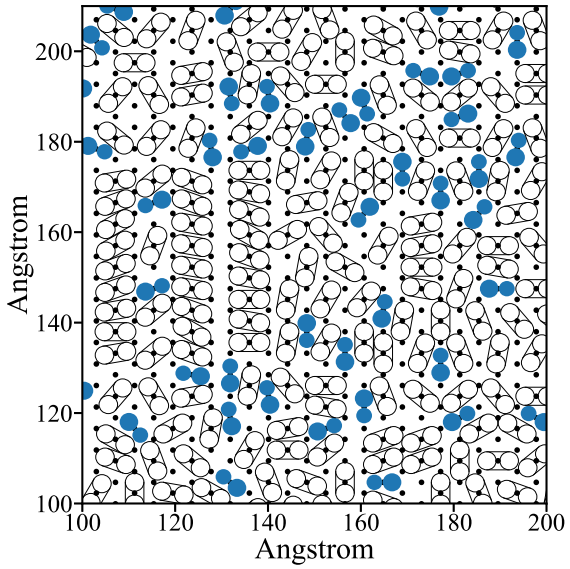


Figure 5.7: The adsorption pattern of Hacac chelate (represented by unfilled geometry) and monodentate (represented by filled geometry) configurations on Al_2O_3 sites (blue dots) from the 2D stochastic simulation model. The empty capsule with two circles in black and the two conjoined blue circles symbolize monodentate and chelate molecules, respectively.

5.3.2 Selectivity of ASALD

Figure 5.8 illustrates the selectivity of Hacac adsorption on Al_2O_3 (NGA) as opposed to SiO_2 (GA) in Step A (Hacac cycle) at T = 423 K and P = 400 Pa. No Hacac adsorption is observed on the NGA, which is also detected throughout the entire operating window, which is in agreement with the experimental results [2, 103]. The selective Hacac adsorption forms a protective layer on the NGA, leading to SiO_2 being deposited on the GA in a selective manner. The selectivity of ASALD

can be influenced by two factors: chemical passivation and steric hindrance. In contrast to the GA, the NGA is chemically deactivated through the adsorption of Hacac as a result of the difference in the magnitude of the activation energies of the adsorption. This chemoselectivity is supported by the DFT calculations as discussed in Section 5.2.2. As shown in Figures 5.2a and 5.2b, the Hacac adsorption on SiO_2 is endothermic and has a high activation energy barrier of 0.831 *eV*. Meanwhile, the Hacac adsorption on Al_2O_3 is exothermic and has a low activation energy of 0.25 *eV*. Therefore, the adsorption predominantly occurs on Al_2O_3, resulting in the localized formation of a protective layer on the NGA. In addition to the chemoselective deposition behavior, the regioselective behavior is accounted for by the inclusion of steric repulsion effects. As discussed in Sections 5.2.3 and 5.3.1, complete Hacac coverage is not possible on the NGA since some reaction sites are sterically hindered by neighboring deprotonated Hacac adsorbates. As a result, the blocked reaction sites are physically deactivated. In conclusion, the NGA is deactivated both chemically and sterically, leading to high effective selectivity for ASALD.

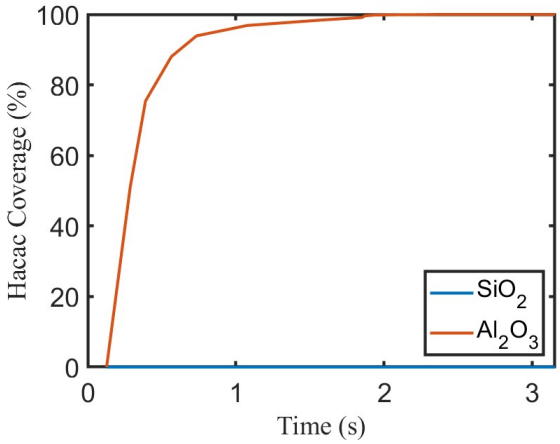


Figure 5.8: Hacac coverage versus time at $T = 423\ K$ and $P = 400\ Pa$. The red and blue solid lines denote the Hacac adsorption on Al_2O_3 and SiO_2, respectively.

5.3.3 Impact of Operating Conditions

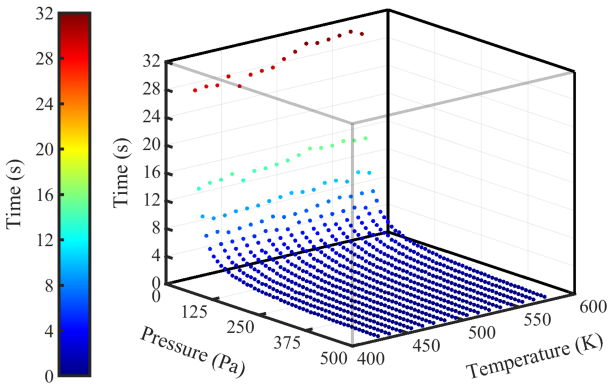
The simulation results in the studied operating window (423 $K \leq T \leq 573$ K and 10 $Pa \leq P \leq 500$ Pa) from the mesoscopic model based on the kMC simulation with steric hindrance are shown in Figure 5.9. 800 data points are collected with a temperature and pressure interval of 10 K and 10 Pa, respectively. Depending on the operating temperature and pressure, the overall process time significantly varies. Specifically, a Hacac dose of more than 3 s is calculated for saturation at a pressure of below 100 Pa. Hacac pulse times from the kMC simulations are comparable to the experiment results, where the Hacac pulse of 5 s per cycle was reported to be used to reach saturation [2]. In the BDEAS cycle, a BDEAS dosage time of 2.3 s at $P = 400Pa$ is computed to reach full coverage, which is consistent with prior experimental research from [103]. In addition, [111] reported that dosing BDEAS for around 2 s was required for saturation on SiO_2 in a typical ALD operating window.

Figure 5.10 shows the process time dependence on the operating temperature and pressure on 2D graphs by presenting isobaric lines produced from Figure 5.9. As expected, a higher operating pressure causes a shorter process time for all steps due to the higher pressure accelerating the physisorption reactions, which can be explained by the kMC algorithm. Surface reactions and desorption reactions, which are based on the Arrhenius equation, are mostly dependent on temperature. On the contrary, the pressure only affects physisorption reactions. Therefore, high pressure solely has a substantial impact on the physisorption step. Specifically, when pressure is increased from 100 to 500 Pa, the process time decreases by a factor of 5.03 for Step A, 4.89 for Step B, and 5.01 for Step C throughout the temperature window. Therefore, the operating pressure linearly affects

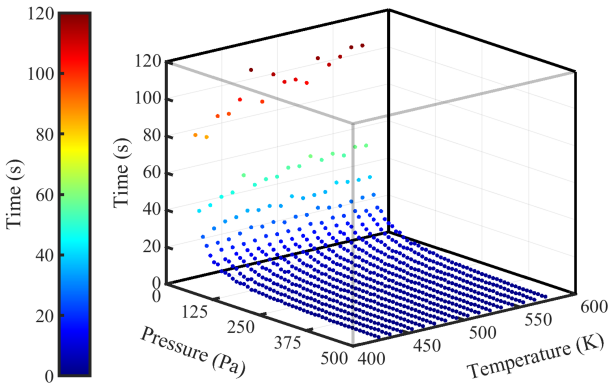
the process time for all steps, which indicates that the physisorption step is the key component in determining the process time.

As shown in Figure 5.10, changing the temperature does not cause any significant impact on the process time in the aforementioned range. Due to the randomness, the process time slightly fluctuates with increasing temperature. Although an increase in temperature accelerates all surface reactions based on the simulation algorithm, Figure 5.10 reveals that the process time is rarely reliant on temperature. Prior research reported that the deposition of BDEAS was nearly constant [111]. With regard to the kMC algorithm, the reaction rate constants of the surface reactions are based on the Arrhenius equation in Eq. (5.3) increase with increasing temperature. However, increasing temperature decelerates the physisorption reaction rate. With the independence of temperature in the ALD growth process, the reduction in the physisorption reaction rates and the increase in the surface reaction rates with increasing temperature offset each other.

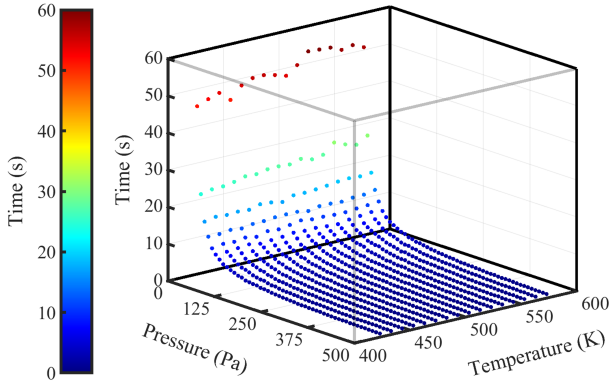
The stochastic behavior of the kMC algorithm may present a variation in dosage times for reaching full coverage. Thus, 100 simulations of the same condition were simulated and averaged to compensate for the randomness and deviation in the dosage times. Histograms depicting the distribution of the dosage times for Steps A, B, and C, are presented in Figure 5.11 and illustrate that most dosage times computed were within a single standard deviation, σ, from the average, μ, dosage time. Therefore, the stochastic behavior has a moderate impact on the precision of the dosage times, which is the reason why averaging the dosage time is an effective method for producing consistent results.



(a) Step A (Hacac cycle)



(b) Step B (BDEAS cycle)

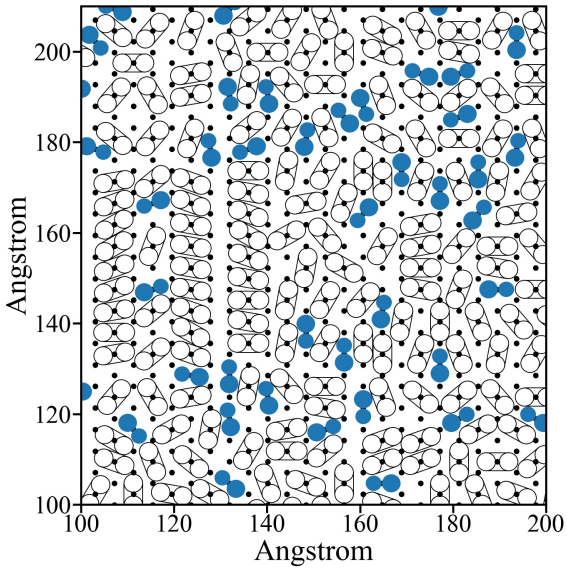


(c) Step C (O_3 cycle)

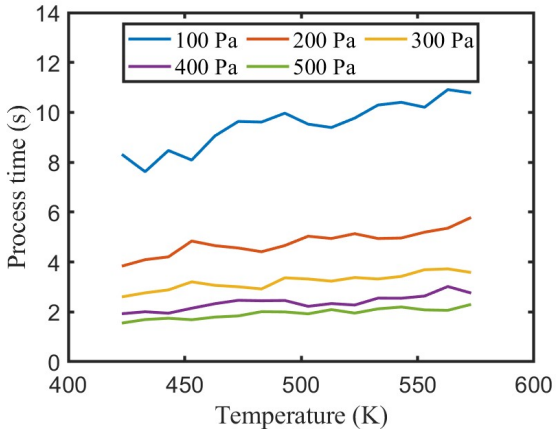
Figure 5.9: Scatter charts for process time collected from kMC simulations under the operating window ($423 \leq T \leq 573\ K$ and $10 \leq P \leq 500\ Pa$) in (a) Step A, (b) Step B, and (c) Step C, respectively.

5.4 Conclusion

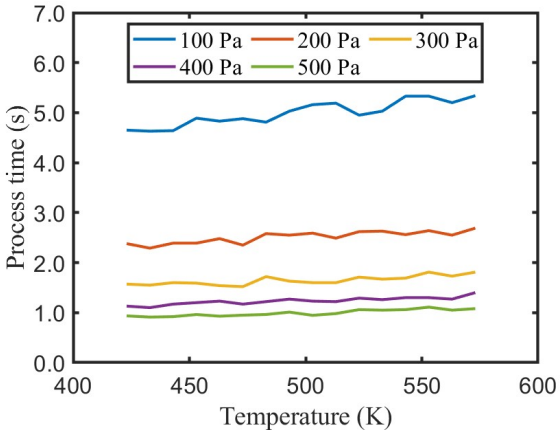
In this chapter, a combined atomistic and mesoscopic model considering the effects of chemoselectivity and regioselectivity (steric hindrance) based on a kinetic Monte Carlo (kMC) simulation was developed to investigate the surface kinetics of area-selective atomic layer deposition (ASALD) of SiO_2/Al_2O_3. Acetylacetone (Hacac), bis(diethylamino)silane (BDEAS), and ozone were used for Steps A through C as small molecular inhibitors (SMI), precursors, and oxidants, respectively.



(a) Step A (Hacac cycle)



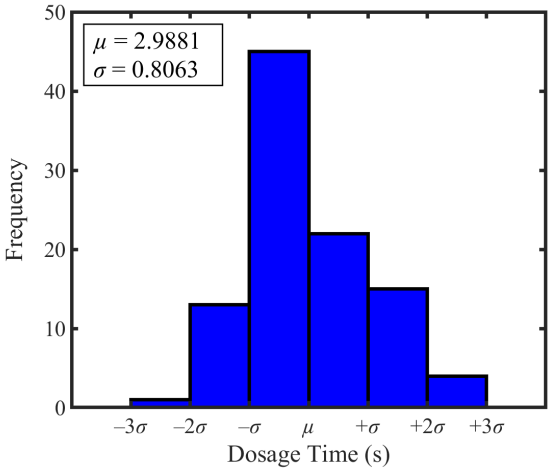
(b) Step B (BDEAS cycle)



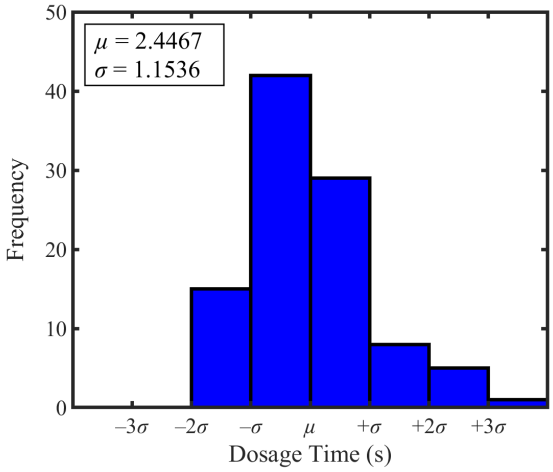
(c) Step C (O_3 cycle)

Figure 5.10: 2D plots of process time as a function of temperature with different pressures in (a) Step A, (b) Step B, and (c) Step C, respectively.

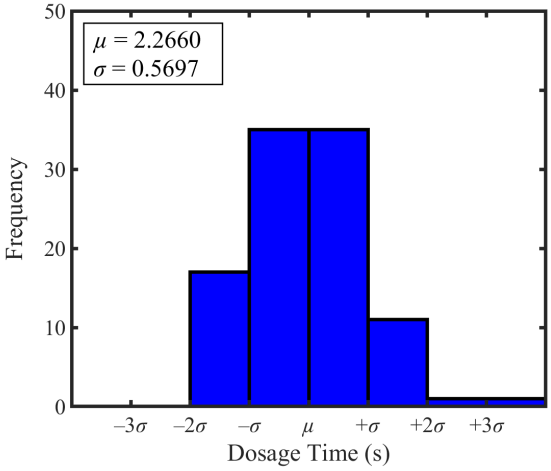
Density functional theory (DFT) calculations including the nudged elastic band (NEB) method were performed to obtain kinetic parameters for the basis of the kinetic mechanisms. This study revealed that the inhibitor forms a protective layer on the non-growth area (NGA), Al_2O_3, as a preferential reaction path as opposed to the growth area (GA), SiO_2, by visualizing the surface



(a) Step A (Hacac cycle)



(b) Step B (BDEAS cycle)



(c) Step C (O_3 cycle)

Figure 5.11: Histograms depicting the distribution of dosage times for 100 iterations while at constant operating conditions for (a) Step A ($T = 523\ K, P = 100\ Pa$), (b) Step B ($T = 523\ K, P = 350\ Pa$), and (c) Step C ($T = 523\ K, P = 200\ Pa$).

coverage of Hacac, demonstrating the chemoselectivity of the Hacac adsorption. Additionally, the regioselective behavior of bulky species, Hacac and BDEAS, was simulated, which determined the preference for particular configurations (chelate configuration) that would provide the least steric hindrance to maximize the surface coverage. It was also observed that regioselective dependence of BDEAS adsorption prevented the precursor from adsorbing onto the Hacac protective layer as

a consequence of steric repulsion effects. In addition, the temperature and pressure dependence on the ALD growth on the GA was discussed. The ALD growth was linearly dependent on pressure, while there was no impact of temperature on the ALD growth on the GA. This simulation work was supported and demonstrated by several experimental studies as discussed in Section 8.3. Therefore, the developed mesoscopic model greatly exemplifies the surface kinetics at the atomic level, providing insight into ASALD as a brand-new technology for bottom-up nanofabrication. This research can be integrated with not only other experimental research, but also a computational fluid dynamics model to optimize the industrial reactor design and the operating conditions of the ASALD process in future research.

Table 5.1: Summary and description of variables.

Variable	Definition
A_{site}	Surface area of an active reaction site
D	Dimension size of a $D \times D$ 2D grid
E	Total electronic energy
E_a	Activation energy
h	Planck constant
$\hbar$	Reduced Planck constant
$\hat{H}$	Hamiltonian operator
k	Reaction rate constant
k_{ads}	Reaction rate constant of adsorption reaction
k_{nonad}	Reaction rate constant of nonadsorption reaction
k_B	Boltzmann constant
k_{sum}	Sum of the reaction constants
m	Atomic mass of adsorption species
m_e	Atomic mass of particle
n	Number of molecules
P	Adsorption species partial pressure
Q	Partition function for the reactant
$Q^{\ddagger}$	Partition function for the transition state
R	Universal gas constant
s	Distance between two adjacent active sites
t	Reaction time progress
T	Operating temperature (absolute)
$\hat{V}$	Potential energy
x	Horizontal distance between two active sites
$\overrightarrow{x}$	Displacement vector
Z	Coordination number of adsorption species
Δt	Time interval
γ_1, γ_2	Random numbers where $\gamma_1, \gamma_2 \in (0, 1]$
ν	Pre-exponential factor
ρ	Molecule density
σ	Sticking coefficient for adsorption species
ψ	Time and position-dependent wave function

Table 5.2: Self-determined parameters for DFT calculations.

Material	Variable	Value
SiO_2	*degauss*	0.02
	$ecutwfc^{\dagger}$	50 *Ry*
	$ecutrho^{\ddagger}$	200 *Ry*
	k-points	2 2 2
Al_2O_3	*degauss*	0.01
	$ecutwfc^{\dagger}$	50 *Ry*
	$ecutrho^{\ddagger}$	200 *Ry*
	k-points	3 3 2

† Kinetic energy cutoff for wave functions.
‡ Kinetic energy cutoff for charge density and potential

Table 5.3: Geometry description for reaction paths

Substrate	Step	Number	Description
Al_2O_3	Step A	A1	OH-terminated Al_2O_3
		A2	Hacac physisorption on Al_2O_3
		A3	Monodentate configuration (deprotonated Hacac on Al_2O_3)
		A4	Chelate configuration (deprotonated Hacac on Al_2O_3)
SiO_2	Step A	V1	OH-terminated SiO_2
		V9	Hacac physisorption on SiO_2
		V10	Monodentate configuration (deprotonated Hacac on SiO_2)
	Step B	V2	BDEAS physisorption on SiO_2
		V3	DEA-deprived BDEAS adsorbate on SiO_2
		V4	SiH_2 adsorbate on SiO_2
	Step C	V5	O_3 physisorption on SiH_2
		V6	H-Si-OH on SiO_2
		V7	O_3 physisorption on H-Si-OH
		V8	Final configuration ($Si(OH)_2$)

Table 5.4: Van der Waals radii of functional groups in Hacac molecule

Functional group	Van der Waals radius (Å)
—CH_3	2.0
—CH_2—	2.0
C═O	1.7

Sourced from [119]

Nomenclature

ALD Atomic layer deposition

ALE Atomic layer etching

ASALD Area-selective atomic layer deposition

BDEAS Bis(diethylamino)silane

CT Collision theory

DEA Diethylamine

DFT Density functional theory

EPEs Edge placement errors

FinFETs Fin field-effect transistors

GA Growth area

GUI Graphical user interface

Hacac Acetylacetone

kMC Kinetic Monte Carlo

MC Monte Carlo

NEB Nudged elastic band

NGA Non-growth area

PAW Projector augmented-wave

PWscf Plane-wave self-consistent field

QE Quantum ESPRESSO

SAM Self-assembled monomer

SCF Self-consistent field

SMI Small-molecule inhibitor

TST Transition state theory

VSSM Variable step size method

$\overrightarrow{g}$ Gravitational Acceleration Constant $9.806\,65\ \mathrm{m\,s^{-2}}$

R Universal Gas Constant$8.314\,463\ \mathrm{m^3\,Pa\,K^{-1}\,mol^{-1}}$

ν Kinematic viscosity of the fluid

$\overline{\overline{\tau}}$ Stress tensor

$\overrightarrow{F}$ External body force

ρ Density of the precursor species

$\vec{A}_i$ Area vector of the cell face

$\vec{c}_i$ Vector from the centroid of the cell to the centroid of the adjacent cell

$\vec{v}$ Velocity of the mixture

A_j Pre-exponential factor for reaction, j

E Internal energy

$E_{A,j}$ Activation energy for reaction, j

h_j Sensible enthalpy of the species, j

J_j Diffusion flux of the species, j

k_i Reaction rate constant for reaction, i

k_j Reaction rate constant for reaction, j

k_{total} Sum of reaction rate constants

N Number of reaction pathways

p Static pressure of the species

S_h Heat transfer source

S_m Mass transfer source term

T Operating temperature of the reactor

t Process time of the reaction

u_∞ Free stream velocity of the fluid

x Boundary layer starting distance from the wall

y_p Distance between the wall to the adjacent cell centroid

β_j Temperature exponent for reaction, j

γ Coefficient for reaction selection and time evolution where $\gamma \in (0, 1]$

G0 Typical reactor geometry

G1 Multi-inlet reactor geometry

G2 Showerhead reactor geometry

G3 Inclined plate reactor geometry

$Al(CH_3)_3$ Trimethylaluminum, TMA

Al_2O_3 Aluminum Oxide

$AlF(CH_3)_2$ Dimethylaluminum Fluoride, DMAF

AlF_3 Aluminum Fluoride

H_2O Water

HF Hydrogen Fluoride

Chapter 6

Multiscale Modeling of Area-Selective Atomic Layer Deposition of SiO_2 on Al_2O_3 and SiO_2 Surfaces with Intermediate Etching Steps

6.1 Introduction

For decades, nanoscale semiconductor manufacturing has relied predominantly on top-down fabrication processes comprising multiple lithography, etching, and deposition steps. Traditional top-down approaches employing Atomic Layer Deposition (ALD) and Atomic Layer Etching (ALE) enable atomic-level precision in film thickness, allowing the formation of highly conformal and ultrathin layers essential for complex three-dimensional architectures such as Fin Field-Effect Transistors (FinFETs) [91, 7] and Gate-All-Around (GAA) [124] structures. These designs substantially increase transistor density, thereby enhancing computational performance while reducing power consumption. Consequently, significant research effort has focused on improving and optimiz-

ing atomic layer-based operations to achieve higher yields and tighter quality control. Both ALD and ALE are characterized by sequential, self-limiting surface reactions, typically involving two half-cycles separated by inert gas purging, that afford sub-nanometer control of material growth or removal. This cyclic mechanism has been applied to a wide range of materials, including metals and metal oxides commonly used as gate dielectrics, such as SiO_2, Al_2O_3, HfO_2, TiO_2, and Ta_2O_5, enabling the precise and uniform deposition of functional thin films critical to advanced semiconductor device fabrication.

With shrinking device dimensions, the semiconductor industry faces mounting challenges in achieving reliable pattern fidelity at the nanoscale. One of the most critical issues is edge placement error (EPE) [2, 92], which is the cumulative misalignment that arises during the sequential deposition, lithography, and etching steps used in conventional top-down fabrication. These small positional deviations between the intended and actual feature locations can lead to unwanted material growth or etching at undesired areas [93]. Although such errors were tolerable in earlier technology nodes with larger gate widths, they become increasingly detrimental in sub-5 nm manufacturing, where even nano-scale misalignments can severely degrade device performance. EPE ultimately limits the achievable stacking complexity of three-dimensional architectures, such as multilayer GAA transistors.

To address these limitations, Area-Selective Atomic Layer Deposition (ASALD) has emerged as a promising bottom-up self-alignment technique for high-volume manufacturing [94, 10]. In ASALD, the substrate is chemically divided into a growth area (GA) where deposition is desired, and a non-growth area (NGA) which is passivated to inhibit nucleation. This selectivity is achieved through surface chemical modification and the careful choice of small-molecule inhibitors (SMIs)

that preferentially adsorb and deactivate the NGA without interfering with reactions on the GA. When the inhibitor effectively blocks precursor adsorption on the NGA, ASALD enables self-aligned, layer-by-layer film growth that reduces the need for conventional lithography and etching steps. Consequently, this approach not only improves process yield and lowers fabrication cost but it also facilitates the realization of more complex three-dimensional device architectures with higher stacking layers [92].

Several strategies have been proposed to achieve selective inhibition on the NGA, among which the use of self-assembled monolayers (SAMs) has become particularly prevalent in academic studies. SAMs consist of long aliphatic tail chains that self-organize through van der Waals interactions, forming a densely packed organic layer in which the molecular heads are chemically anchored to the substrate surface. This configuration provides an effective diffusion barrier that suppresses precursor nucleation on the NGAs, allowing deposition to proceed primarily on the desired GA. The successful application of SAM-based inhibitors has been reported in multiple studies [96, 97, 98, 99, 100]. However, SAMs exhibit several intrinsic limitations. First, their coverage is rarely defect-free, restricting reliable inhibition to only a few nanometers of film thickness on GA. More critically, SAM formation typically relies on wet-chemical processing and requires extended reaction times on the order of tens of minutes to achieve sufficient surface passivation. These constraints hinder their integration with conventional vapor-phase ALD and ALE processes, where rapid cycling and compatibility with high aspect ratio features are essential. As a result, SAM-based inhibition is unsuitable for high-volume manufacturing of dielectric films in advanced gate structures that require both high selectivity and conformal coverage.

To overcome the limitations associated with SAM-based inhibition, small-molecule inhibitors

(SMIs) have been proposed as a more versatile alternative. Unlike SAMs, which rely on long-chain organic molecules that block precursor adsorption primarily through hydrophobicity and physical steric hindrance, SMIs employ short, volatile molecules delivered in the vapor phase to passivate the NGA via a combination of chemical bonding and steric shielding [102]. The key advantage of SMIs lies in their compatibility with existing ALD process conditions: they can be rapidly introduced and reacted in the gas phase, enabling straightforward integration into standard reactor configurations. By incorporating an additional inhibition step, the conventional AB-type ALD cycle can be extended into an ABC-type ASALD cycle [2], where:

- Step A involves selective adsorption of the inhibitor onto the NGA to prevent precursor nucleation.
- Step B introduces the precursor, which reacts exclusively on the unblocked GA.
- Step C supplies the co-reactant to complete the surface reaction on the GA.

This three-step ASALD structure not only ensures selective film growth but it also allows the use of reactive oxidant co-reactants in Step C. Such co-reactants can both finalize the desired deposition on the GA and simultaneously remove residual inhibitors from the NGA, thereby regenerating the surface for the next cycle. This vapor-phase inhibition strategy has been successfully demonstrated in SiO_2 ASALD systems employing SiO_2 (GA) and Al_2O_3 (NGA) substrates, with validation from both experimental studies [2] and density functional theory (DFT) assisted kinetic Monte Carlo (kMC) simulations [3]. In these implementations, the inhibitor layer is cyclically removed during the ozone oxidation step (Step C) and re-adsorbed in the subsequent inhibition step (Step A) to maintain selectivity over multiple deposition cycles.

However, prior experimental work has shown that the inhibitor is not always perfect. For example, in area-selective ALD of SiO_2 on a SiO_2 growth surface with an Al_2O_3 non-growth surface, the acetylacetone (Hacac) inhibitor does not completely block precursor nucleation on the NGA [103]. In fact, up to about 8% of the BDEAS precursor can still adsorb on the Al_2O_3 NGA despite the inhibitor [103], indicating that the blocking is imperfect and the selectivity can be easily lost. This unwanted deposition will accumulate with continued cycling, and eventually the growth rate in NGA becomes identical to that in GA after an initial nucleation delay of roughly 10–20 ALD cycles [93, 125, 126]. One possible solution to mitigate this loss of selectivity is to add an etching step, after the deposition steps, forming an ABCD-type ALD sequence with step D serving as a selective etch [92]. The 'step D' here denotes an arbatrily comprehensive etching procedure that may actually contain multiple internal steps to complete the etching process. The etching process can selectively remove the unwanted nuclei on the NGA while incurring only a minimal loss of material on the GA. Indeed, such ABCD-type (ALD-etch supercycle) processes have been demonstrated for a variety of material systems, for instance, area-selective deposition of Ruthenium [125], Titanium Nitride [93]and of Ta_2O_5 [126] with an additional thermal ALE etching step whereby periodic removal of nuclei on the NGA maintains high selectivity without significantly compromising the deposition efficiency on GA. In this chapter, the term "batch" refers to one minimal repeatable process unit, consisting of a complete sequence of elementary steps. Depending on the process configuration, a batch may include different numbers of steps: two steps for conventional ALD, three steps for traditional AS-ALD, and five steps (ABCDE) for the AS-ALD with integrated etching considered here.

Despite the aforementioned merits, there remains a notable gap in both computational and ex-

perimental studies analyzing the multi-batch performance of SMI-based ASALD processes—particularly those employing an ABCD-type cycle with integrated etching steps to suppress unwanted nucleation on the NGA. Prior computational work by [3], assumes ideal inhibitor performance and neglects precursor nucleation on the NGA, a simplification that contradicts experimental observations indicating incomplete blocking and progressive selectivity loss. To address this, the present study introduces a Monte Carlo–based collision model that captures the stochastic surface interactions of inhibitor adsorption, precursor deposition, co-reactant oxidation, and exposure-limited etching. This microscopic model enables a detailed investigation into how inhibitors adsorb, how nucleation events initiate and accumulate on the NGA under imperfect blocking conditions, and how etching parameters such as reaction time and exposure conditions can be tuned to recover and sustain high selectivity during multi-batch, high-thickness ASALD. Building on this, a computational fluid dynamics (CFD) model of the ALD reactor that is developed in collaboration with NIST and validated by optical $MoCl_5$ flow experiments is coupled with the Monte Carlo simulation via PyFluent to construct a multiscale digital twin of the ASALD process. This integrated framework captures the real-time reactor-scale pressure dynamics and enables cycle-accurate simulation of surface coverage evolution, providing a predictive, application-ready platform for optimizing ASALD systems in realistic fabrication environments.

6.2 Microscopic Monte-Carlo Based Collision Model

The objective of the ASALD process is to enable selective thin-film metal oxide deposition on the GA that is represented by the $\beta - SiO_2$ (101) surface, while suppressing deposition on the

NGA which is typically $\alpha - Al_2O_3$ (0001). The full ASALD cycle in this chapter consists of five steps. Step A involves the adsorption of the small-molecule inhibitor acetylacetone (Hacac) as it has significant difference in activation energy barriers in GA and NGA [3], which can be selectively chemisorbed onto the NGA surface to passivate it and prevent precursor nucleation. Step B introduces the silicon precursor bis(diethylamino)silane (BDEAS), which reacts with surface hydroxyl groups to form Si–H terminations on the GA, but it also leads to limited and undesired nucleation on the NGA due to imperfect inhibition. In Step C, the co-reactant ozone oxidizes the precursor-modified surface, converting Si-H terminations to Si-OH to complete both ideal and unwanted SiO_2 deposition on both surfaces. Simultaneously, the ozone step also removes other species, including residual inhibitors, from both surfaces, preparing them for the next cycle. To restore and maintain high selectivity over multiple cycles, the process incorporates a fully thermal, HF-based atomic layer etching (ALE) sub-sequence that is compatible with vapor-phase integration and is consistent with established TMA/HF conversion–etch mechanisms for oxide materials [127, 128]. In this scheme, Step D is a TMA exposure that serves two coupled roles at the microscopic level: (i) it removes any AlF_3-terminated surface segments generated during the HF step of the previous batch by forming volatile Al-containing products, and (ii) it converts exposed SiO_2 segments (including undesired nuclei on the NGA) into an Al-O-Si / Al_2O_3-like modified state via a conversion reaction, without immediate geometric thickness removal. Step E is a thermal HF exposure that fluorinated these Al-containing modified segments to an AlF_3-like state, which is then removed during the subsequent batch's Step D.

To simulate the microscopic surface reactions of the ASALD process, a Monte Carlo collision model was developed by discretizing the surface into a two-dimensional grid of reactive sites. This

model allows for stochastic simulation of molecular adsorption, steric interactions, and nucleation events. Each surface representing either SiO_2 as the GA or Al_2O_3 as the NGA is initialized as a regular lattice, where each site may host an adsorbed species depending on the reaction conditions and spatial constraints. Molecules such as inhibitors (e.g., acetylacetone), precursors (e.g., BDEAS), and reaction intermediates are represented with geometric footprints defined by spatial structure and the van der Waals radii of the functional groups of molecules. This geometric representation ensures that new adsorbates cannot overlap with existing ones, thereby introducing steric exclusion effects that influence local surface coverage. The resulting grid-based simulation provides physically meaningful outputs, such as coverage fractions, inhibitor retention, and precursor breakthrough events, which are critical for quantifying selectivity loss and guiding etch process integration. An illustration of the initialized empty grids for both GA and NGA surfaces is shown below in Figure 6.1 and Figure 6.2 to demonstrate the layout before molecular adsorption.

The general Monte Carlo simulation scheme proceeds by repeatedly sampling adsorption and reaction events on the discretized surface. In each trial, a site is randomly selected from the surface grid, and a physisorption time increment is added to the cumulative simulation time. If the chosen site is already occupied, the algorithm immediately moves on to the next trial. If the site is unoccupied, the simulation evaluates whether the target molecule (e.g., inhibitor, precursor) can be adsorbed at the location without overlapping any nearby molecule, determined by checking steric compatibility using van der Waals radii.

To resolve steric interactions between adsorbed species and enforce non-overlapping placement in the surface Monte Carlo simulation, all molecular templates are constructed using a computational geometry framework based on convex polygonal approximations. Circular functional

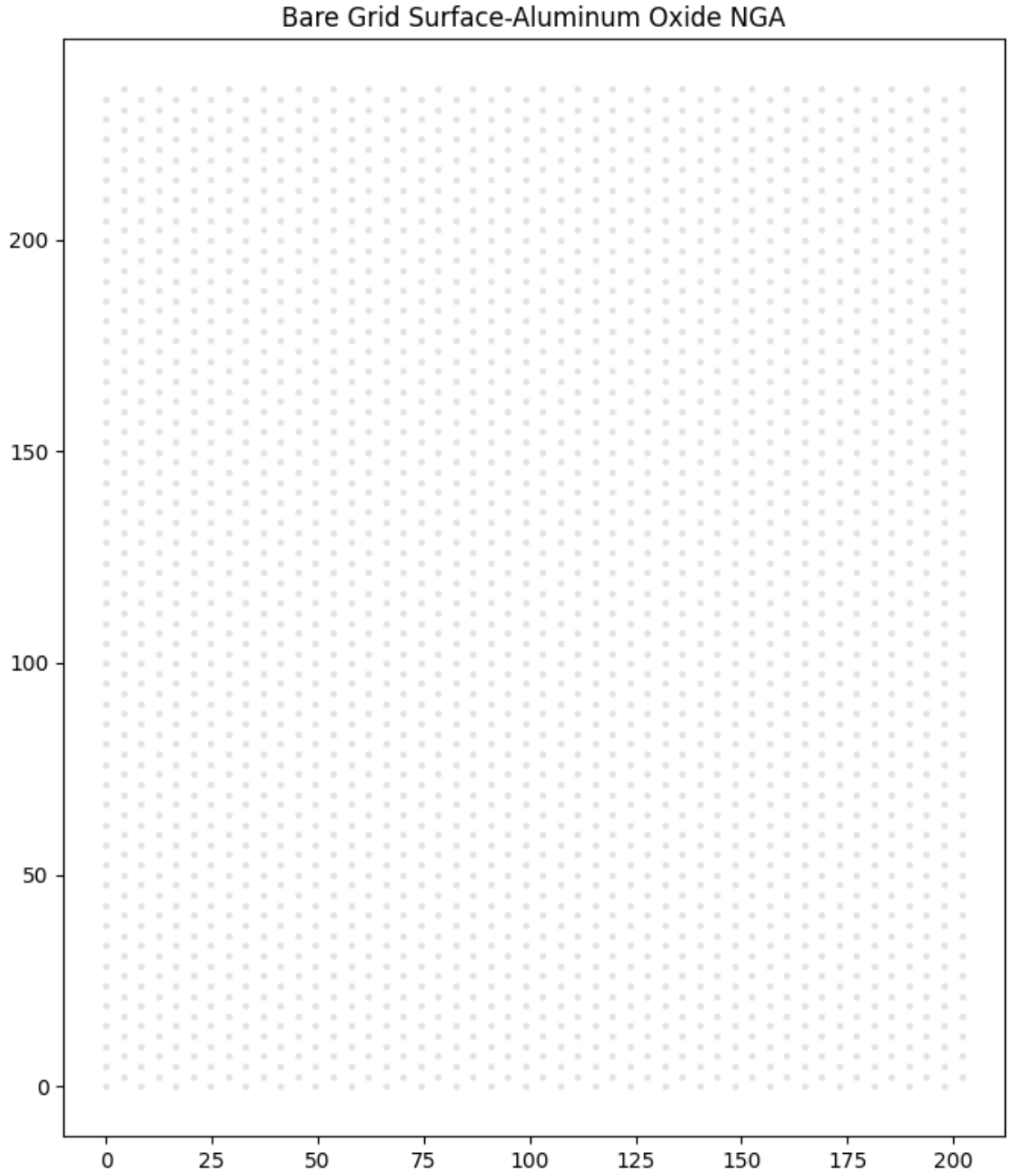


Figure 6.1: Bare Al_2O_3 staggered arrangement grid surface with uniform 4.76Å site distance.

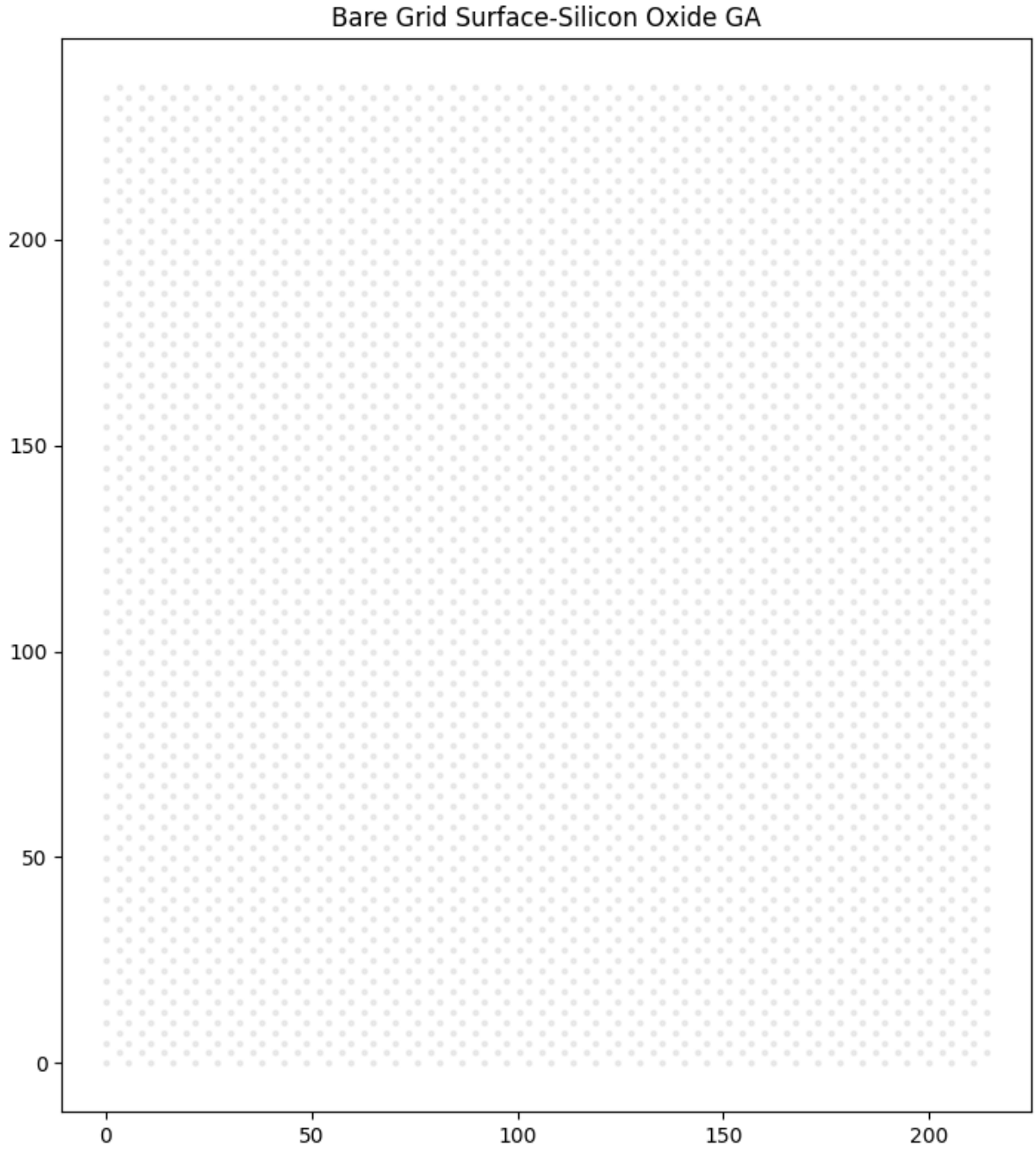


Figure 6.2: Bare SiO_2 nonuniform staggered arrangement. Described in [3], surface grid separated into 2-column groups, within each group the vertical site distance 4.99Å, horizontal distance 3.13Å, between-group distance 5.40Å.

groups—such as methyl (CH_3) or silane (SiH_2) units—are represented as regular polygons constructed from uniformly spaced points along the circumference. Specifically, each circle is approximated by a 32-sided polygon, generated by discretizing the boundary using angular subdivision. These convex polygons are analytically tractable and allow robust geometric operations such as unions, intersections, and transformations. Composite molecular geometries are assembled by uniting multiple convex subcomponents into a single surface footprint.

Explicit electronic polarization of adsorbed molecules is not treated in the present model. Instead, molecular sizes and steric exclusion are represented using effective van der Waals–based geometries that capture the average spatial extent of adsorbed species on the surface. These effective geometries inherently incorporate, in a mean-field sense, the influence of electronic effects such as polarization and charge redistribution on intermolecular spacing. Importantly, the chosen molecular footprints are not arbitrary: they are validated by reproducing experimentally reported inhibitor coverages, precursor nucleation fractions, and growth behavior on both growth and non-growth areas. Within the scope of this chapter, the dominant factors governing selectivity and film evolution are steric hindrance, site availability, and exposure-limited surface kinetics, rather than subtle polarization-induced deformation of adsorbed molecules. Explicit treatment of polarization would require quantum-mechanical resolution of charge redistribution and geometry relaxation and is therefore incompatible with the lattice-based kinetic Monte Carlo framework and wafer-scale simulation objectives pursued in the present work. Neglecting explicit polarization thus represents a controlled approximation that preserves physical fidelity at the process-relevant scale while enabling tractable multiscale simulation.

Collision detection between any two molecular geometries A and B is based on evaluating the

overlap of their occupied spatial regions. A steric conflict is defined mathematically as

$$A \cap B \neq \emptyset, \tag{6.1}$$

and the corresponding overlap area is computed as

$$A_{\text{overlap}} = \text{Area}(A \cap B). \tag{6.2}$$

If $A_{\text{overlap}} > 0$, the candidate placement is rejected due to geometric interference. Otherwise, the molecule can be safely placed on the surface. This check is applied between every candidate template and all currently placed geometries to maintain steric exclusion throughout the simulation.

To improve computational efficiency, especially in high-coverage surface regimes, a spatial indexing strategy is employed using a sort-tile-recursive tree (STRtree) structure. The STRtree is a spatial search tree that organizes the currently placed geometries by their bounding boxes, allowing rapid querying of nearby features. When a new molecule is proposed for placement, the tree is queried to retrieve only those existing geometries whose bounding boxes intersect a local search window around the candidate site. Exact polygonal overlap checks are then performed only within this reduced set, substantially reducing the number of pairwise intersection tests per Monte Carlo trial.

The overall surface is managed using a structured grid system initialized by a grid-construction routine. Each site contains spatial coordinates, state variables indicating its current chemical or physical condition, and (if occupied) the geometry associated with the adsorbed molecule. Geometry templates for each molecule type and orientation are precomputed to enable fast trial place-

ment. At initialization, the grid includes empty containers for bidentate, monodentate, and trilate geometries; a list of all lattice site positions; and a uniform state vector tracking adsorption and transformation events. The spatial index tree is rebuilt as needed to reflect the evolving surface configuration. Together, this geometric and grid-based framework enables rigorous steric enforcement and scalable simulation of surface dynamics during the ASALD process.

If spatial placement is possible, the simulation then checks whether the molecule is eligible for further reaction steps. If subsequent surface reactions are defined for the molecule, each reaction is evaluated sequentially for feasibility and executed accordingly. Each successfully executed reaction contributes an additional reaction time increment to the total time counter.

The physisorption and surface reaction time steps are governed by kinetic models. Specifically, the reaction time increment $\frac{\Delta t}{N}$ (N is the number of sites in the simulation grid) for any event is computed as a stochastic sample from the uniform distribution using the general reaction rate k as shown in eq. (6.3).

$$\Delta t = -\frac{\ln \gamma}{k} \tag{6.3}$$

where γ is a uniformly distributed random number in $(0, 1]$, and k is either k_{phys} for physisorption reactions or k_{surf} for surface reactions. The reaction rate k_{phys} for physisorption processes is computed using *collision theory* in eq. (6.4).

$$k_{phys} = \frac{P A_{\text{site}} \sigma Z}{\sqrt{2\pi m k_B T}} \tag{6.4}$$

where P is the gas-phase pressure, A_{site} is the surface area per adsorption site, σ is the sticking

coefficient, Z is the coordination number, k_B is the boltzmann constant, m is the molecular mass, and T is the temperature. The sticking coefficient used for Hacac is 1.0×10^{-4} [7], for BDEAS is 2.0×10^{-5} [112], for O_3 is 4.5×10^{-5} [129], and for TMA and HF, because of a lack of experimental data on exact surfaces used in this chapter, the TMA sticking coefficient is set to be the same as the BDEAS precursor, and the HF applies the O_3 sticking coefficient. For thermally activated surface reactions (e.g., ligand exchange or bridge formation), the reaction rate is calculated based on *transition state theory* (TST) in eq. (6.5).

$$k_{surf} = \frac{k_B T}{h} \frac{Q^{\ddagger}}{Q} \exp\left(-\frac{E_a}{RT}\right) \tag{6.5}$$

Here, $Q^{\ddagger}$ and Q are the partition functions for the transition state and reactant, respectively, E_a is the activation energy, and h is Planck's constant. These kinetic formulations ensure that surface reactions are both spatially constrained and temporally resolved with physical accuracy. For simplicity, the $\frac{Q^{\ddagger}}{Q}$ is assumed to be 1 in this chapter. The activation energies of ASALD reactions calculated by Density Functional Theory (DFT) are demonstrated in [3].

In the present microscopic Monte Carlo framework, neither spontaneous thermal desorption nor lateral surface diffusion of chemisorbed species is explicitly modeled. Thermal desorption is neglected based on physical timescale considerations: experimental studies of small-molecule inhibitors and adsorbed precursor fragments in ASALD systems indicate that spontaneous desorption occurs on timescales of hours to days under typical ALD and ALE operating temperatures [103], whereas adsorption, surface reactions, and etching steps occur on timescales of seconds. As a result, natural desorption does not measurably affect surface coverage evolution within a single ALD

or ALE half-cycle and can be safely neglected for the exposure-limited regimes investigated here. Surface diffusion of chemisorbed species is also not included explicitly. In the present model, adsorption events are stochastic in both spatial location and molecular orientation, and repeated adsorption across cycles already produces an effectively randomized surface configuration. From a statistical perspective, explicit surface diffusion would primarily serve to redistribute adsorbates without altering overall coverage or reaction probability, while substantially increasing computational complexity due to repeated geometric relocation and collision checks. Because the focus of this chapter is on multi-batch selectivity trends and exposure-limited growth and etching behavior rather than equilibrium surface rearrangement, neglecting surface diffusion represents a controlled and reasonable approximation consistent with prior lattice-based and kinetic Monte Carlo ALD modeling approaches.

6.2.1 Step A: Inhibitor Adsorption

The inhibitor employed in this chapter is acetylacetone (Hacac), a molecule that exists in equilibrium between keto and enol tautomeric forms. In the gas phase, the enol configuration is known to be more thermodynamically stable than the keto form due to the stabilization provided by intramolecular hydrogen bonding [105]. Upon introduction to the Al_2O_3 non-growth surface, which features surface hydroxyl groups in a vicinal diol arrangement, the acidic enol form of Hacac undergoes a condensation reaction with the basic –OH terminated surface. This results in monodentate adsorption, where the molecule binds via a single Al–O bond and releases H_2O as a byproduct.

Following monodentate adsorption, the Hacac molecule can undergo an energetically favorable transition to form a second Al–O bond, producing a chelate (or bidentate) configuration. This

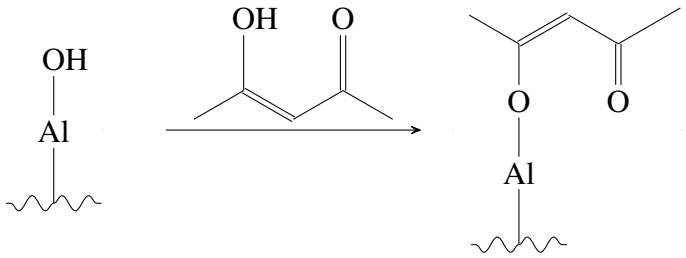


Figure 6.3: Orange molecules are monodentates consisting of two circles, blue molecules are chelates consisting of a rectangle and semicircles.

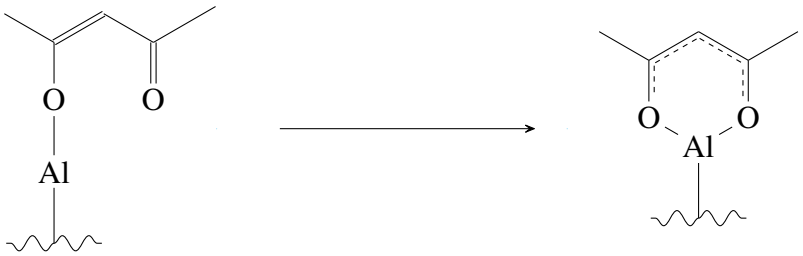
chelation results in a six-membered ring in which the π-electrons are delocalized, creating a stable conjugated system. To capture steric effects in simulation, the chelate geometry is represented as two circles (diameter: 4.0Å, corresponding to the van der Waals radius of CH_3 groups) spaced 4.8Å apart. The intermediate space, comprising the CH_2 bridge, is approximated as a convex shape—effectively modeled as a rectangle with semicircular ends. The sample shapes that are attached to the NGA is shown below in Figure 6.3.

The monodentate configuration, which lacks the second Al–O bond, is modeled differently due to the out-of-plane displacement of one of the CH_3 groups [3]. It is represented by two circles: one with a 4.0Å diameter (CH_3) and the other with a projected effective diameter of 3.4Å, reflecting the horizontal projection of the non-planar methyl group. These circles are spaced 3.8Å apart to reflect the molecular geometry derived from quantum chemical calculations and steric mapping. This geometric abstraction enables accurate modeling of surface coverage and steric hindrance effects during inhibitor adsorption. The schematic diagrams of the modeling of chelates and monodentates are shown in Figure 6.4 and Figure 6.5, and the reaction mechanisms and molecule structures are

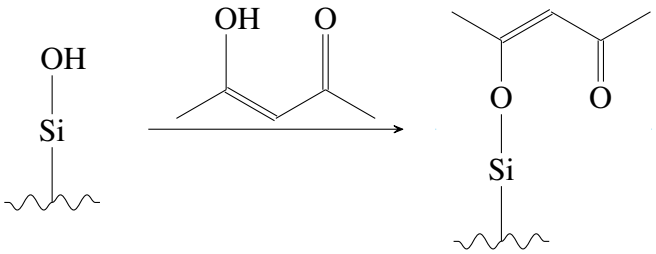
demonstrated below as well to help understanding.



+ H_2O (Step A-1)



(Step A-2)



+ H_2O

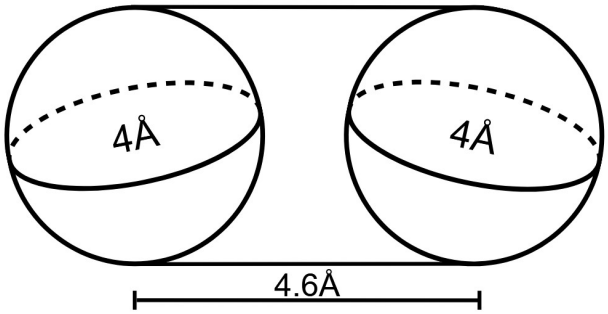
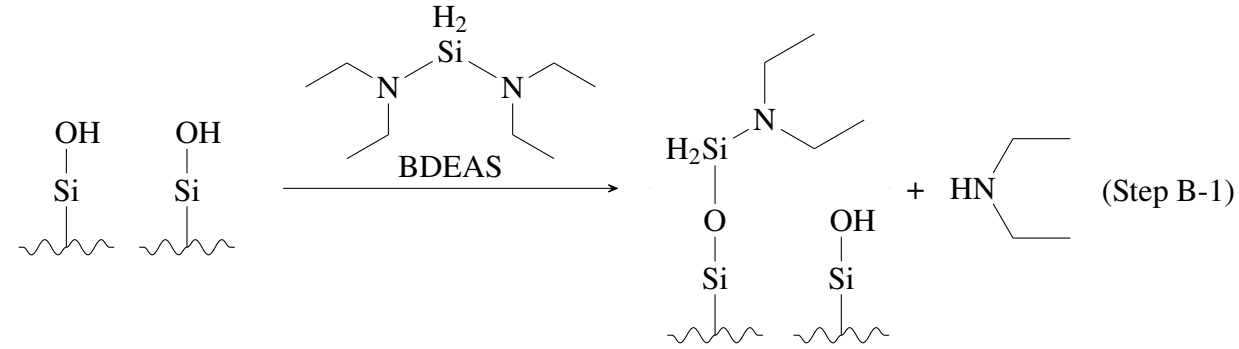
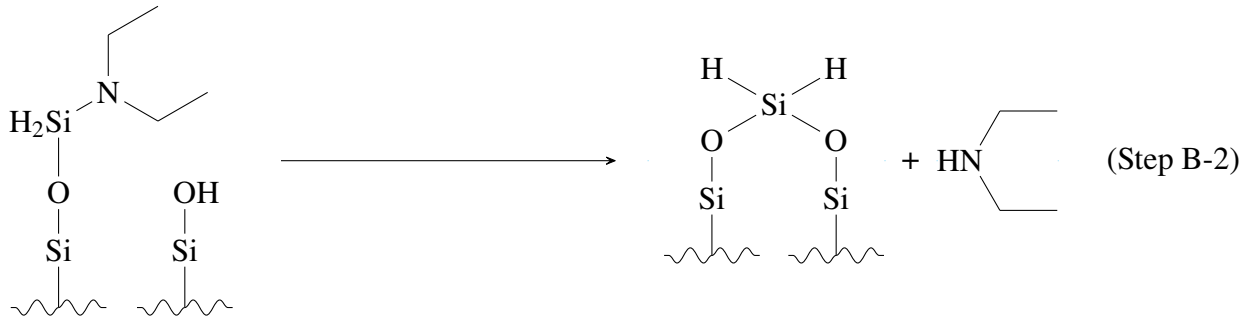


Figure 6.4: Diagram of Chelate molecule. Each 4Å circle denotes a CH_3 group on the same horizon.





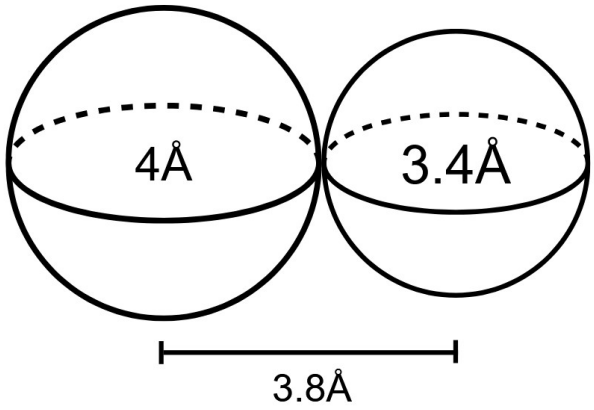


Figure 6.5: Diagram of Monodentate molecule with two circles of 4Å diameter denoting a CH_3 group and 3.4Å diameter denoting projection of CH_3 on another side.

During the inhibitor adsorption process, the placement of each acetylacetone (Hacac) molecule is governed by a stochastic sampling approach. For each trial, a surface site is randomly selected, and a list of candidate rotation angles ranging from 0° to 345° in 15° increments is generated and randomly shuffled to ensure variability in orientation attempts. Each angle in the shuffled list is sequentially tested to determine whether a monodentate Hacac configuration can be accommodated at the selected site without overlapping adjacent molecules, based on van der Waals exclusion criteria. If none of the candidate orientations result in a sterically viable placement, the site remains unoccupied in that trial. The discretization with 15° step size is a trade-off between computational complexity and accuracy, as with thorough computational testing, it was found that step sizes below 15° do not lead to significant differences in the computed results. If a monodentate configuration fits successfully, the simulation then attempts to place a chelate (bidentate) configuration at the same site, starting from a reshuffled list of angles. This second angle list accounts for the fact that the single Al–O bond in monodentate species permits rotational flexibility, so the feasible orientations for chelate attachment are not constrained by the previously chosen monodentate

angle. The process checks again for steric collisions in all attempted orientations and selects the first valid placement. If there are no available angles without collision with adjacent molecules, then it keeps the monodantate placed before. To reduce computational complexity, once either a monodentate or a chelate is successfully adsorbed, the molecule is considered geometrically fixed. That is, no further adjustment or rotation is permitted after placement. This assumption reflects a quasi-static treatment of adsorbed configurations and is consistent with other lattice-based surface Monte Carlo models that prioritize configurational diversity during trial selection rather than after placement [130].

6.2.2 Step B: Precursor Adsorption

After the adsorption of acetylacetone inhibitors in NGA, the bis(diethylamino)silane (BDEAS) precursor is introduced in Step B to initiate deposition on both surfaces. For simplicity, the model assumes that the NGA is fully covered by inhibitor at the start of Step B with sufficiently long reaction time for Step A, and the GA remains completely available for precursor adsorption, so no need to consider the inhibitor-GA interactions. Full coverage means the available geometric space on the surface has been filled to the maximum extent permitted by steric constraints. BDEAS is a silicon-based precursor containing two diethylamine (DEA) ligands and two hydrogen atoms bonded to the silicon center. Upon exposure to hydroxyl-terminated SiO_2 surfaces, BDEAS undergoes a ligand-exchange reaction in which one DEA group is displaced, forming a covalent Si–O bond with the surface and releasing a DEA molecule into the vapor phase. DEA is a volatile byproduct and remains stable under the thermal ALD conditions employed.

Following this initial attachment, the adsorbed BDEAS species may seek out an adjacent hy-

droxyl site to undergo a second ligand-exchange reaction, releasing the remaining DEA group and forming a silicon bridge structure across the two sites. As described in [3], the geometric layout of the β-SiO_2 surface constrains bridge formation to occur only between specific neighboring columns. Due to the lateral spacing between surface sites, bridges can only form between pairs of vertical site columns, leading to incomplete surface coverage—a phenomenon also supported by experimental observations [111]. Although the trilate DEA–silane configuration represents a necessary intermediate prior to silane bridge formation, such intermediates do not accumulate on the GA surface due to the distinct surface environment and reaction pathway. On the GA (SiO_2), the surface is free of inhibitor molecules and presents a high density of accessible hydroxyl sites. Under these conditions, once a BDEAS-derived trilate intermediate is formed, an adjacent hydroxyl site is almost always available, enabling rapid conversion into a bridged silane structure. This conversion step has a very low activation barrier [3] and is effectively instantaneous on the timescale of the Monte Carlo simulation, preventing trilate intermediates from persisting. In contrast, on the NGA, inhibitor molecules introduce significant steric hindrance that frequently blocks access to neighboring hydroxyl sites even when they are chemically vacant, kinetically trapping the trilate configuration. On the GA, the only scenario in which a trilate could transiently exist is when both neighboring bridge-forming sites are already occupied. However, in this configuration, the surrounding bridged structures themselves impose sufficient steric exclusion to suppress additional precursor adsorption in the vicinity, preventing sustained trilate populations. Consequently, persistent trilate intermediates are not observed on the GA, and the resulting bridge-only growth mode naturally reproduces the experimentally reported SiO_2 surface coverage of approximately 86–94%.

On the NGA (Al_2O_3), the situation differs. Experimental studies have shown that Hacac

chelate structures are chemically robust and effectively block BDEAS substitution [103]. However, the bonding between the monodentate inhibitor and the surface is weaker and can be displaced by BDEAS. In the simulation logic, precursor adsorption begins by randomly selecting a candidate site. If the site is occupied by a chelate inhibitor, the trial is aborted, as chelates are considered non-substitutable. Otherwise, those surrounding monodentates are temporarily removed, and the algorithm attempts to place a trilate structure, which is a geometric abstraction of the single DEA silane structure after first chemisorption of BDEAS molecule.

The trilate is modeled using three circles: a central circle (diameter 5.0 Å) representing the silane group, and two side circles (diameter 4.0 Å) representing the middle CH_2 of ethyl groups. The terminal CH_3 groups are ignored as they are located on the upper horizon. The two wings are placed at a 45° angle from the central axis, with each wing offset 3.44 Å from the center. This structure is visualized in Figure 6.6. If none of the tested orientations fit without overlapping neighboring adsorbates, the trial is discarded and the monodentates are restored. If the trilate is successfully placed, any previously removed monodentates that still geometrically fit are reintroduced around the trilate to preserve partial inhibitor coverage.

After successful trilate placement, the simulation checks whether an adjacent site is available for bridge formation. The bridge structure is a simplified remnant of the trilate, retaining only the central circle, as both DEA groups are eliminated, and extending across to an adjacent surface site, provided no steric collisions occur. If such a connection is feasible, the bridging structure is formed, representing the completion of Si–O–Si bond formation between two hydroxyl sites on both surfaces.

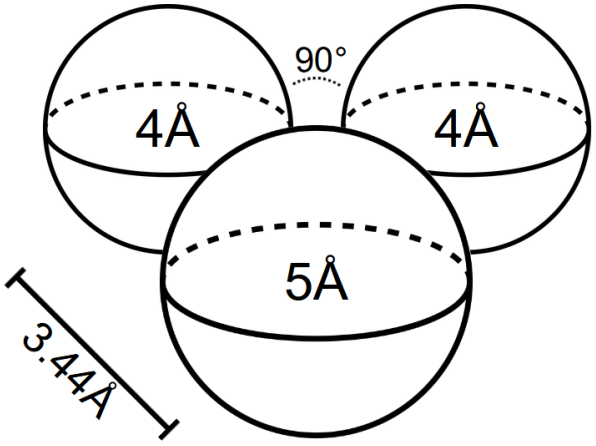


Figure 6.6: Diagram of Trilate DEA-Silane Structure. Center 5Å diameter circle denotes the Si-H_2 silane group, and each 4Å diameter circle denotes the ethyl group.

6.2.3 Step C: Ozone Oxidation

Following the precursor adsorption stage in Step B, the process proceeds to Step C, in which the surface is exposed to ozone to initiate oxidation and cleaning. The ozone exposure performs two primary functions: (1) it oxidizes the terminal -H groups on surface-bound silicon atoms into -OH groups, thereby completing the SiO_2 deposition process by creating the hydroxyl-terminated SiO_2 surface and has volatile O_2 as byproduct, and (2) it removes remaining surface-bound species, including monodentate and chelate inhibitors on the NGA and unreacted trilate precursor structures on both surfaces. It is important to distinguish between the chemical fate of fully incorporated silane bridge species and that of unreacted precursor-derived intermediates during the ozone step. Bridged silane structures are doubly anchored to adjacent surface hydroxyl sites and are therefore structurally integrated into the surface network. Experimental studies of SiO_2 ALD consistently show that such bridged species undergo oxidation of residual Si–H bonds during ozone exposure [103], yielding stable, hydroxyl-terminated SiO_2 surface species. In contrast, unreacted trilate BDEAS interme-

diates are only singly bound to the surface and retain a largely molecular character. Under ozone conditions, these weakly incorporated species do not follow the film-forming oxidation pathway, instead, they undergo ligand combustion and oxidative fragmentation, producing volatile byproducts that desorb from the surface rather than contributing to oxide growth.

In the simulation, ozone reactions are executed by randomly sampling surface sites. If a selected site is empty, the algorithm proceeds to the next trial. If the selected site corresponds to an endpoint of a silane bridge structure (i.e., a site occupied by a BDEAS-derived Si-H species), the simulation changes the state of the site to represent a newly formed Si-OH group. It is important to note a modeling simplification adopted here: although the two -H groups on the silicon atom are physically located between the two bridged surface sites (perpendicular to the bonding axis), the model assigns the oxidation product to the original endpoint sites. That is, ozone oxidation updates the state of the selected endpoint site to reflect SiO_2 formation, even though the actual -OH group lies spatially between the sites. This abstraction enables a clean and consistent update of grid-based surface state variables while avoiding the need for explicit sub-grid spatial resolution.

Furthermore, once both ends of a bridge are oxidized, the corresponding pair of sites is fixed and preserved as an intact deposited SiO_2 segment. In subsequent ALD cycles, when BDEAS adsorbs onto one of these previously deposited SiO_2 sites, the simulation enforces an immediate bridge formation with the adjacent paired site from the prior batch. This assumption ensures that deposited SiO_2 layers remain laterally uniform in height (i.e., all bridged sites belong to the same thickness layer), which simplifies thickness tracking and avoids complex three-dimensional structure resolution or adjacency-based bridging logic.

Lastly, if ozone selects a site occupied by residual adsorbates such as monodentate or chelate

inhibitors, or unreacted trilate BDEAS structures, the associated molecular geometries and corresponding entries in the simulation grid are removed. This cleanup operation resets those sites to an unoccupied state, making them available for inhibitor adsorption or precursor attachment in the next cycle.

6.2.4 Step D: TMA Exposure for AlF_3 Removal and Conversion Modification

Following the main deposition steps, Step D introduces a trimethylaluminum (TMA) exposure that implements the reactive half-cycle of a thermal TMA/HF ALE scheme. In the mechanism, Step D performs two distinct but physically linked functions that depend on the chemical state of the local surface. First, TMA removes AlF_3-terminated segments that were created during the HF step of the previous batch by converting them into volatile Al-containing products, thereby producing true material removal in the model. Second, on hydroxyl-terminated SiO_2 segments, TMA drives a conversion reaction that forms Al-O-Si / Al_2O_3-like modified surface species. This conversion does not immediately reduce the geometric film thickness; rather, it prepares the affected segments for fluorination during Step E, after which they become AlF_3 and are removed during the subsequent batch's Step D. This state-dependent, cycle-decoupled treatment reflects the well-established thermal ALE principle that modification (conversion/fluorination) and removal can occur in different half-cycles [127, 128].

From a simulation perspective, TMA molecules are modeled as stochastically sampling lattice sites on the surface. If the selected site is empty, the trial is skipped. If the selected site belongs to a deposited bridge segment, the model applies a state-dependent rule as follows:

(i) **Removal of AlF_3 (previous-batch product).** If the sampled bridge is in an AlF_3-terminated state (generated during Step E of the previous batch), the segment is eligible for removal during Step D. In this case, the entire bridge (both endpoints) is removed as a single correlated event, representing volatilization of the fluorinated layer during TMA exposure. If the removed segment resides on the first deposited layer (layer index = 1), the corresponding geometry is deleted and the sites return to the underlying substrate state. If the segment resides on a higher layer (layer index > 1), its layer index is decremented by one to represent net vertical removal while retaining the underlying stack.

(ii) **Conversion of SiO_2 (current-batch modification).** If the sampled bridge is in a hydroxylated SiO_2 state, TMA does not directly remove material in the same batch. Instead, the bridge is converted into an Al-containing modified state (Al-O-Si / Al_2O_3-like) while preserving its geometric representation and layer index. This chemical-state update captures conversion modification that enables subsequent fluorination during Step E. As in the deposition steps, if TMA samples one endpoint of a bridge, both connected sites are updated simultaneously to ensure consistent segment-level chemistry.

To more accurately represent the reactivity contrast between NGA and GA, Step D includes an exposure-based accessibility model. Since not all surface sites are equally accessible to incoming TMA molecules due to geometric crowding and steric shielding from surrounding features, a local exposure score is computed for each site before TMA attachment is accepted. Exposure is modeled as the fraction of a hemisphere above the surface site that remains unblocked by nearby deposited bridges.

This is computed in two stages:

1. **Horizontal (2D) shielding:** As shown in Figure 6.7, for each adjacent bridge within a two-step hexagonal neighborhood, the tangent lines to its circular footprint define the angular sector it obstructs. For a single nearby bridge, this horizontal angular block is computed as

$$\theta_{\text{block}}^{(j)} = 2 \cdot \arcsin\left(\frac{r}{d_s^{(j)}}\right) \tag{6.7}$$

where r is the effective blocking radius (typically the midpoint of the bridge), and $d_s^{(j)}$ is the center-to-center surface distance between the current site and adjacent bridge j.

2. **Vertical (elevation) shielding:** As shown in Figure 6.8, the elevation angle blocked by each neighbor is calculated as

$$\phi_{\text{block}}^{(j)} = \arctan\left(\frac{(L_j - L_i) \cdot H_{\text{layer}}}{d_s^{(j)}}\right) \tag{6.8}$$

where L_j and L_i are the discrete layer indices (heights) of the adjacent and current sites, respectively, H_{layer} is the height per monolayer in Å, and $d_s^{(j)}$ is the lateral distance between these two sites.

Assuming hemispherical access, the total blocked surface area fraction on the hemisphere is estimated from the combined solid angle of horizontal and vertical obstructions. However, since TMA is assumed to attack the root of a surface bridge from one side only, the molecule can only access half the hemisphere as the other half is geometrically obstructed by the bridge itself.

The hemisphere coverage blocked by a single adjacent bridge j is approximated as

$$f_{\text{block}}^{(j)} = \frac{\theta_{\text{block}}^{(j)} \cdot \sin\left(\phi_{\text{block}}^{(j)}\right)}{2\pi} \tag{6.9}$$

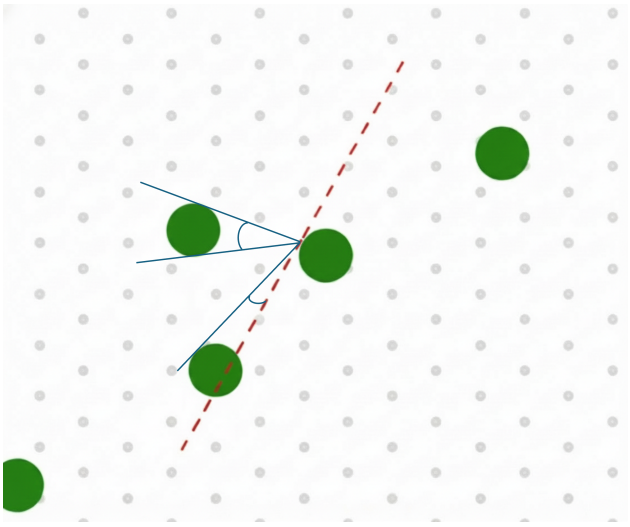


Figure 6.7: The diagram for 2D exposure mechanism. The selected bridge would block half of the hemisphere, and the adjacent bridges would block by angles between tangent lines.

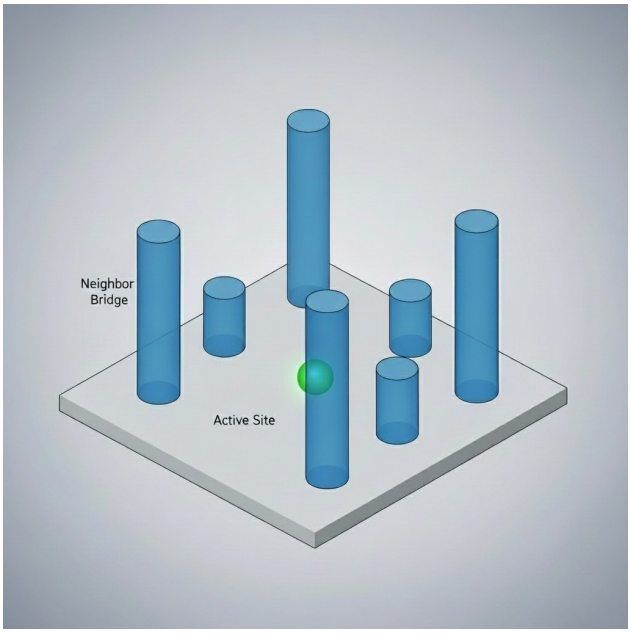


Figure 6.8: The diagram for 3D exposure mechanism. The height difference between the selected site and surrounding deposited sites can results in different obstructions.

and the total unblocked exposure at a site is computed as

$$\text{Exposure} = 0.5 - \sum_{j} f_{\text{block}}^{(j)} \tag{6.10}$$

For computational efficiency, only bridges whose midpoint lies within a hexagonal neighborhood of radius 2 (i.e., two hex steps from the central site) are considered in the blocking calculation. The hex-distance of 1 is defined by the six closest neighbors, and hex-distance of 2 includes the twelve next-closest sites forming a second shell.

Finally, the NGA exhibits higher effective exposure in early batches due to its lower nucleation density. This causes the initially sparse islands on the NGA to experience greater etch reactivity, while the denser GA remains more shielded. Similar phenomena have also been shown in the higher reaction rates of the protruding area in other surface reactions [131], and plasma etching can smooth the rough surface [132]. This exposure-driven etch contrast serves as the foundation for selective ALE that can suppress unwanted nucleation on the NGA while preserving most of the growth on the GA, thereby maintaining long-term process selectivity. In this chapter, exposure-based accessibility is explicitly considered only for the TMA modification step (Step D). While exposure limitations and steric hindrance are present for all surface reactions, Steps A–C are treated as self-limiting ALD processes and are simulated with sufficiently long exposure times such that detailed exposure effects have a limited impact on the final surface state. In contrast, Step D is intentionally exposure-limited and directly controls the subsequent etching behavior; allowing this step to proceed to completion would eliminate net deposition. This simplifying assumption reduces computational cost while focusing on the dominant selectivity-controlling mechanism. Extending exposure-based modeling

to all reaction steps will be considered in future work.

It is important to note that the thermal TMA/HF conversion–etch chemistry employed in this chapter is not intrinsically perfectly selective to SiO_2 over Al_2O_3. Under sufficiently long or repeated exposures, both oxide materials can undergo fluorination and subsequent removal through Al-containing intermediate formation [128]. Accordingly, the ALE step described here does not imply absolute chemical preservation of the Al_2O_3 non-growth area (NGA). The focus of this study is instead on exposure-limited, kinetically driven ALE, in which effective selectivity emerges from differences in local exposure, nucleation density, and roughness evolution between the growth area (GA) and the NGA. In practical integration, prolonged ALE would gradually consume the underlying NGA oxide; this limitation can be mitigated by employing a sufficiently thick blocking layer prior to ASALD so that gradual consumption during corrective ALE does not compromise device functionality over the targeted number of cycles. Future extensions of this chapter could explicitly incorporate NGA thickness consumption into the multiscale framework and investigate run-to-run control strategies that jointly optimize ALE exposure time and initial blocking-layer thickness to ensure long-term selectivity and process robustness.

Step E: HF Fluorination of the TMA-Modified Surface

After the TMA exposure in Step D, Step E applies a thermal hydrogen fluoride (HF) exposure that fluorinates the Al-containing modified surface segments formed during the current batch. In this step, Al-O-Si / Al_2O_3-like modified segments are converted into an AlF_3-terminated state. Consistent with the thermal HF-based ALE mechanism, this fluorination step is treated as a self-limiting chemical transformation that prepares the modified layer for removal during the next batch's

TMA exposure, rather than producing immediate thickness loss within the same batch [127, 128].

In the microscopic simulation, HF exposure updates the chemical state of any TMA-modified bridge segment to an AlF_3 state while preserving its geometric representation and layer index. HF is not allowed to remove material directly in the model; instead, removal is executed in Step D of the subsequent batch when TMA encounters an AlF_3-terminated segment and volatilizes it. This explicit decoupling ensures that the model captures the cycle-dependent nature of thermal ALE, in which fluorination and removal occur in different half-cycles.

6.2.5 Definition of Film Thickness and Selectivity

In the present work, film thickness is computed from the microscopic Monte Carlo simulation by tracking the average number of deposited surface layers on the growth area (GA) and non-growth area (NGA). Each completed silane bridge formed during Step B and oxidized during Step C is treated as one deposited SiO_2 layer element in the geometric stack. The normalized average layer number on each surface, denoted as L_{GA} and L_{NGA}, is obtained by averaging the discrete layer indices over all surface sites. The physical film thickness is then calculated as

$$T_{\mathrm{GA}} = L_{\mathrm{GA}} \cdot h_{\mathrm{SiO_2}}, \qquad T_{\mathrm{NGA}} = L_{\mathrm{NGA}} \cdot h_{\mathrm{SiO_2}}, \tag{6.11}$$

where $h_{\mathrm{SiO_2}}$ is the effective monolayer thickness of SiO_2 per ALD cycle. In this study, $h_{\mathrm{SiO_2}}$ is taken as constant and chosen to be consistent with experimentally reported growth-per-cycle values for SiO_2 ALD using BDEAS and ozone (typically 0.9–1.1 Å per cycle).

A key feature of the revised thermal TMA/HF ALE mechanism is that chemical conversion

and physical removal are temporally decoupled across batches: Step E (HF) converts Al-containing modified segments into AlF_3, while physical removal of AlF_3 occurs during the subsequent batch's Step D (TMA). Therefore, the instantaneous end-of-batch surface state after Step E may contain AlF_3-terminated segments that are chemically prepared for removal but not yet physically removed. To enable consistent batch-wise reporting of net thickness change and selectivity, we define the effective post-batch thickness as the thickness obtained after removing all AlF_3-terminated segments in a bookkeeping operation that mirrors the removal action that will occur at the start of the next batch's TMA exposure. This bookkeeping operation is used only for metric evaluation and does not represent an additional physical process step in multi-batch simulation.

Selectivity is defined as a normalized thickness contrast between the growth area and non-growth area:

$$S = \frac{T_{\mathrm{GA}} - T_{\mathrm{NGA}}}{T_{\mathrm{GA}} + T_{\mathrm{NGA}}}. \tag{6.12}$$

This definition yields $S = 1$ for ideal area-selective deposition with no net growth on the NGA and $S = 0$ when the GA and NGA exhibit identical thicknesses. This selectivity metric is used consistently throughout the manuscript to quantify the effectiveness of inhibition and the TMA/HF-based ALE correction across multiple deposition batches.

6.3 Macroscopic Computational Fluid Dynamics Model

To accelerate the commercialization and optimization of area-selective atomic layer deposition (ASALD) technology, it is essential to connect the surface-level phenomena of ASALD with the conditions encountered in full-scale industrial reactor systems. While experimental approaches re-

main invaluable for understanding process behavior, they are often costly, time-consuming, and limited in their ability to provide spatially and temporally resolved in situ data. In contrast, multi-scale in-silico modeling offers a cost-effective solution to these limitations by simultaneously analyzing surface kinetics and fluid transport phenomena within the reactor environment.

In this chapter, we expand upon the previously developed microscopic Monte Carlo surface model by integrating it with a computational fluid dynamics (CFD) framework, thereby enabling simulation of real-time ASALD behavior under dynamic reactor conditions. The objective is to develop a fully functional digital twin that captures both local reaction kinetics and global transport characteristics, with particular emphasis on reproducing the spatial pressure and temperature profiles on the wafer surface. Such a model facilitates accurate prediction of film growth behavior under realistic operating conditions and provides a tool for process optimization and control.

To ensure industrial relevance and physical accuracy, we build upon an experimentally established ALD reactor developed by the National Institute of Standards and Technology (NIST). This reactor system includes a well-characterized precursor delivery mechanism, reaction chamber, and outlet piping. Importantly, it is supported by comprehensive experimental measurements, which allow us to verify the accuracy of our CFD modeling approach. The reactor geometry and flow architecture are representative of industrial-scale ALD systems, enabling direct comparison between simulated and experimental results and thereby enhancing the credibility of the macroscopic simulation framework.

The following section section 6.3.1 will describe the reactor design, computational domain, and mesh generation for the CFD simulation. We then present detailed comparisons between simulation results and experimental measurements to validate the CFD setup, boundary conditions, and

model parameters in section 6.3.2.

6.3.1 Reactor Design and CFD Model Development

The simulations were performed using a reactor geometry based on the optically accessible perpendicular-flow single-wafer ALD chamber developed at NIST [133]. The chamber consists of an expansion cone leading into a cylindrical reactor body with three optical access ports and a centrally located heated wafer chuck. Precursor and carrier gases are introduced through four stainless-steel delivery lines (~4.8 mm ID) positioned symmetrically at the top of the chamber.

Computational domain and mesh: The computational domain included the full chamber volume, inlet manifolds, and exhaust lines. The mesh was an unstructured tetrahedral grid with prism layers along solid surfaces to resolve near-wall gradients. Local refinement was applied in three critical regions: (i) the inlet jet zone, (ii) the optical imaging plane, and (iii) the wafer surface. Mesh quality was maintained with a minimum orthogonal quality of ≈0.3 and an aspect ratio less than 4. A mesh-independence study confirmed that both peak $MoCl_5$ arrival time and integrated wafer-plane concentration varied by less than 3%, and the intermediate mesh was adopted for all cases.

Governing equations and species model: Argon served as the carrier gas, and $MoCl_5$ was treated as a non-reactive tracer to isolate transport effects (reaction kinetics will be introduced in subsequent multiscale coupling). A multicomponent diffusion model was used with a user-defined binary diffusivity D_{MoCl_5-Ar} in the range of 2×10^{-4} to 1×10^{-3} m^2 s^{-1}. Flow symmetry and plume structure were insensitive to variations within this range, indicating convective dominance under the examined flow conditions. The reactor temperature was fixed at 393 K to match the experiments.

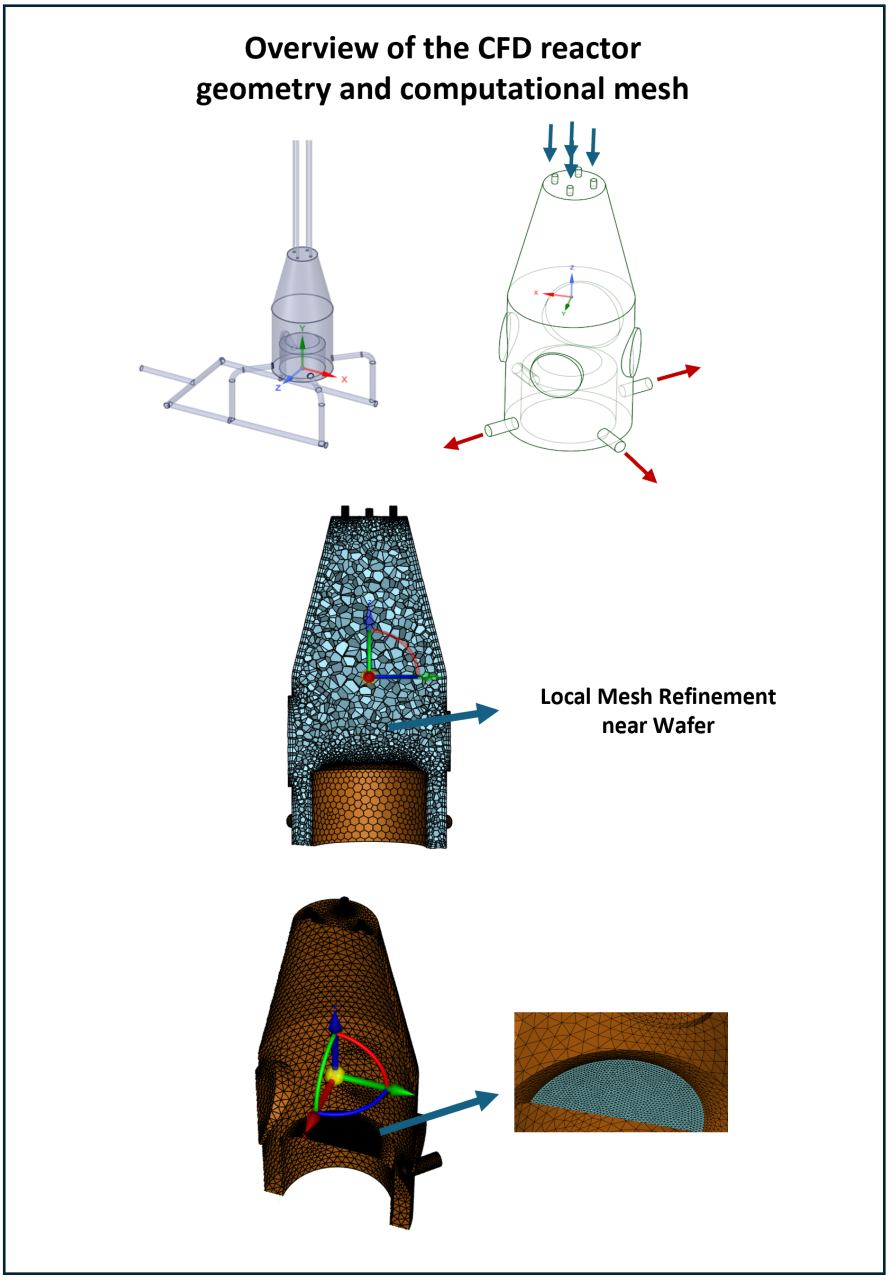


Figure 6.9: Overview of the CFD reactor geometry and computational mesh. (Top) Optically accessible ALD reactor with inlet and outlet configurations. (Middle) Unstructured tetrahedral volume mesh showing refined regions near the optical window and wafer. (Bottom) Detailed view of the prism boundary layers and local refinement applied at the wafer surface.

Turbulence modeling: At relatively low flow rates—such as the 0.1 L min^{-1} (100 SCCM) case with approximately 3 mol% $MoCl_5$ injection—the flow remained in a low-Reynolds-number transitional regime. Under these conditions, a laminar model successfully reproduced the experimentally observed symmetric plume. At higher flow (0.5 L min^{-1}, $\approx$ 2.4 kPa), however, transitional vortices developed near the wafer surface. Standard RANS models (k–ω SST, Transition SST) excessively damped this unsteady vortex motion, resulting in artificial asymmetry in the $MoCl_5$ distribution. To resolve large-scale unsteadiness while maintaining RANS stability near walls, the Scale-Adaptive Simulation (SAS) model was employed. The SAS approach restored the experimentally observed symmetric flow structure and accurately captured the natural vortex dynamics under the higher-flow regime.

6.3.2 Verification of CFD Model

(a) Feed-line and outlet flow validation.

To ensure accurate boundary conditions, separate inlet- and outlet-line simulations were conducted prior to full reactor modeling. For the inlet line, velocity and pressure were benchmarked against experimental data [133] to confirm realistic flow delivery. The simulated upstream inlet pressure (1044 Pa) agreed with the measured 998 Pa within 4.6%, and the outlet velocity, i.e., inlet reactor velocity (40.9 m s^{-1}), matched the measured 41 m s^{-1}. A separate outlet simulation verified the expected pressure drop through the exhaust network, yielding 308 Pa inlet pressure and 58.5 m s^{-1} outlet velocity, consistent with the full reactor' s $\approx$310 Pa prediction. These results confirm that both inlet and outlet configurations accurately reproduce the measured pressure–velocity relationships, providing reliable boundary inputs for subsequent transient $MoCl_5$ transport

simulations.

(b) Reactor flow structure and quantitative validation.

For validation, the 100 SCCM case was used to assess the CFD model's ability to reproduce key flow and transport parameters observed experimentally. The simulated chamber pressure (318 Pa) closely matched the NIST-measured value of 317.6 Pa at the chamber location (P2) in Figure 6.10, demonstrating excellent agreement. In addition, the inlet pressure was consistent with the experimental range of 300–340 Pa after accounting for the expected pressure drop through the inlet delivery lines.

The transient $MoCl_5$ concentration buildup and decay also reproduced the measured absorbance profiles at multiple points (inlet, mid-chamber, and outlet), confirming correct temporal evolution and flow symmetry as shown in Figure 6.11.

The velocity distribution at the wafer surface was predominantly between 0.10 and 0.12 m s^{-1} (blue region), with only localized areas reaching up to 0.33 m s^{-1}, aligning with the experimentally derived value of 0.1 m s^{-1} shown in Figure 6.12.

The residence time, obtained from the simulated $MoCl_5$ mass-fraction evolution as shown in Figure 6.13, was approximately 0.8 s (from 0.7 to 1.5 s), matching the experimentally observed gas residence time under the same flow and pressure regime.

(c) Image-based validation via circular-masked planes.

To directly compare CFD predictions with the optical measurements from the NIST setup, simulated $MoCl_5$ concentration fields were post-processed into circular-masked planes corresponding to the reactor's viewing window, through which the telecentric lens and CMOS camera captured time-resolved absorbance images [134]. The resulting sequence of simulated images reproduced

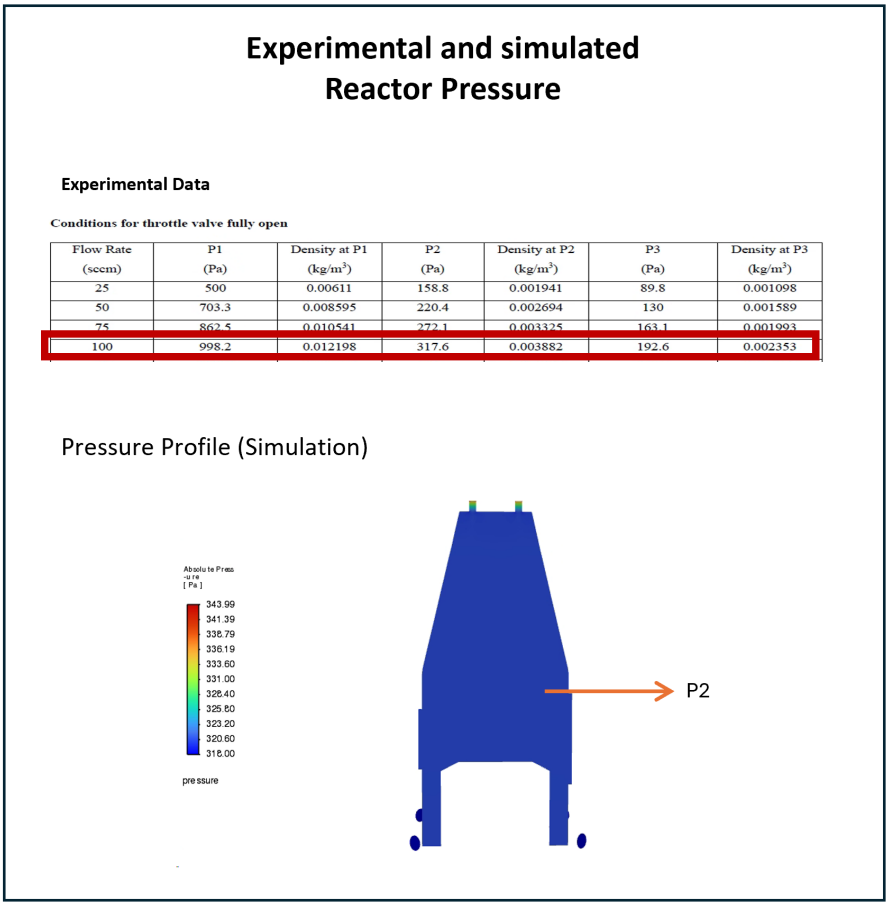


Figure 6.10: (Top) Experimental data from NIST showing pressure at multiple locations. (Bottom) Simulated absolute pressure contour with highlighted chamber pressure region (P2), showing excellent agreement (317 Pa experimental vs. 318 Pa simulated).

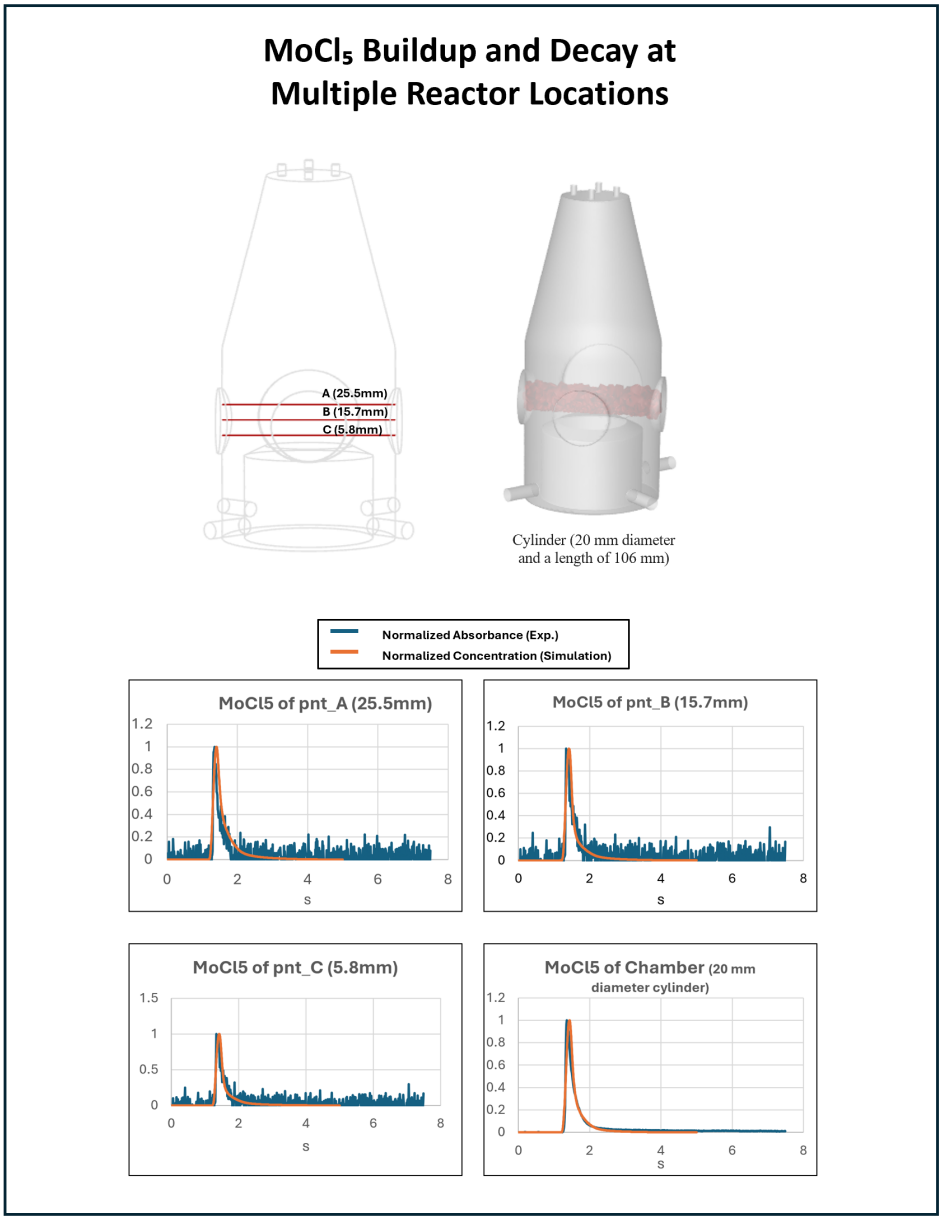


Figure 6.11: Comparison of simulated $MoCl_5$ concentration and experimentally measured absorbance at multiple chamber locations (Maslar and Kalanyan, 2025). Top: Reactor geometry showing monitoring points A (25.5 mm), B (15.7 mm), C (5.8 mm), and the wafer-plane cylinder (20 mm diameter, 106 mm length). Bottom: Normalized time-series data illustrating $MoCl_5$ buildup and decay, showing close agreement in transient behavior and peak timing between simulation and experiment.

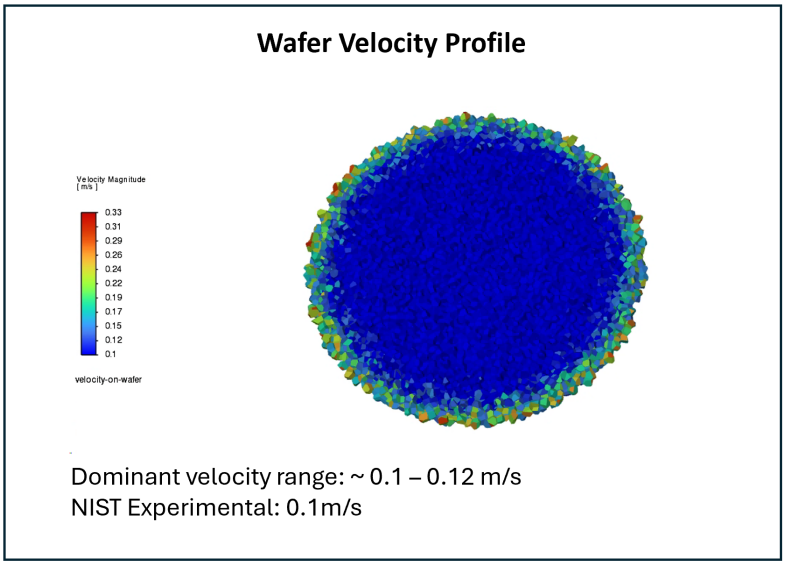


Figure 6.12: Wafer surface velocity distribution showing dominant velocity range of 0.10–0.12 m s^{-1} (blue region), consistent with the NIST experimental measurement of 0.1 m s^{-1} .

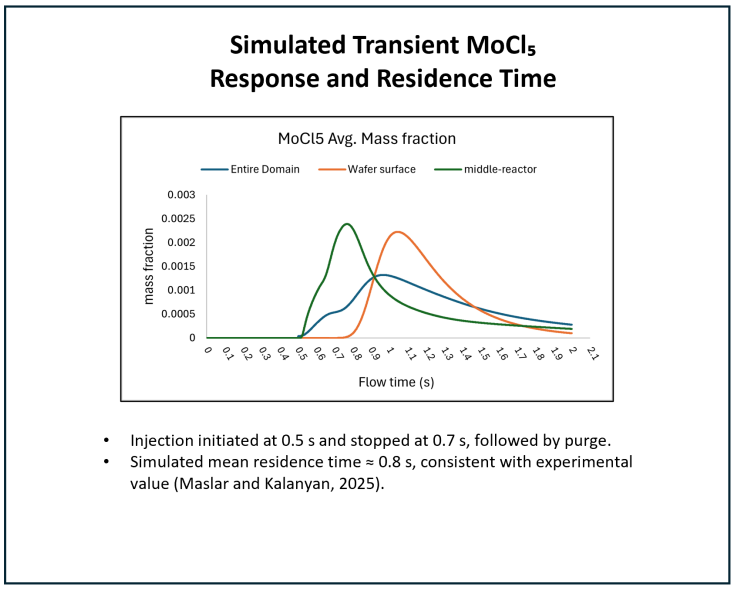


Figure 6.13: Transient $MoCl_5$ concentration response showing buildup during precursor pulse (0.5–0.7 s) and subsequent decay during purge. The average residence time of ≈0.8 s closely matches the experimentally reported value (Maslar and Kalanyan, 2025).

the buildup and decay of $MoCl_5$ during pulsed injection with high spatial and temporal fidelity as shown in Figure 6.14. Both the symmetry of the spreading front and the decay dynamics matched the experimental observations, with consistent transition timing between frame intervals (~1.12–1.45 s).

This image-based validation confirms that the CFD model accurately captures the spatial uniformity, transient $MoCl_5$ transport, and gas residence-time characteristics observed experimentally. Combined with the quantitative agreement in chamber pressure, wafer-surface velocity, residence time, and $MoCl_5$ buildup/decay profiles, these results validate the CFD model and establish its predictive reliability for subsequent reaction-coupled ALD simulations.

Although the CFD model uses Ar and $MoCl_5$ as a heavy, non-reactive tracer system for validation, this choice does not affect the fidelity of the multiscale integration because the CFD framework is designed to provide reactor-scale pressure and transport dynamics that are independent of specific molecular chemistry. The purpose of the validation step is to ensure that the turbulence model, boundary conditions, residence time, and transient precursor delivery are accurately captured. Once these macroscopic flow-field characteristics are verified, Fluent applies its built-in multicomponent diffusion model to simulate the transport of the ALD precursors such as BDEAS, O_3, TMA, and HF, using their individually defined diffusivities and material properties, making the model fully transferable and scalable to different species. Moreover, species transport in this reactor operates in a convection dominated regime, so differences in molecular diffusivity have only a minor influence on macroscopic flow and delivery patterns. Additionally, ALD is an atomic-layer process in which only a single molecular layer is deposited per cycle; the associated reactant consumption and byproduct formation occur within an extremely thin region above the wafer and do not significantly

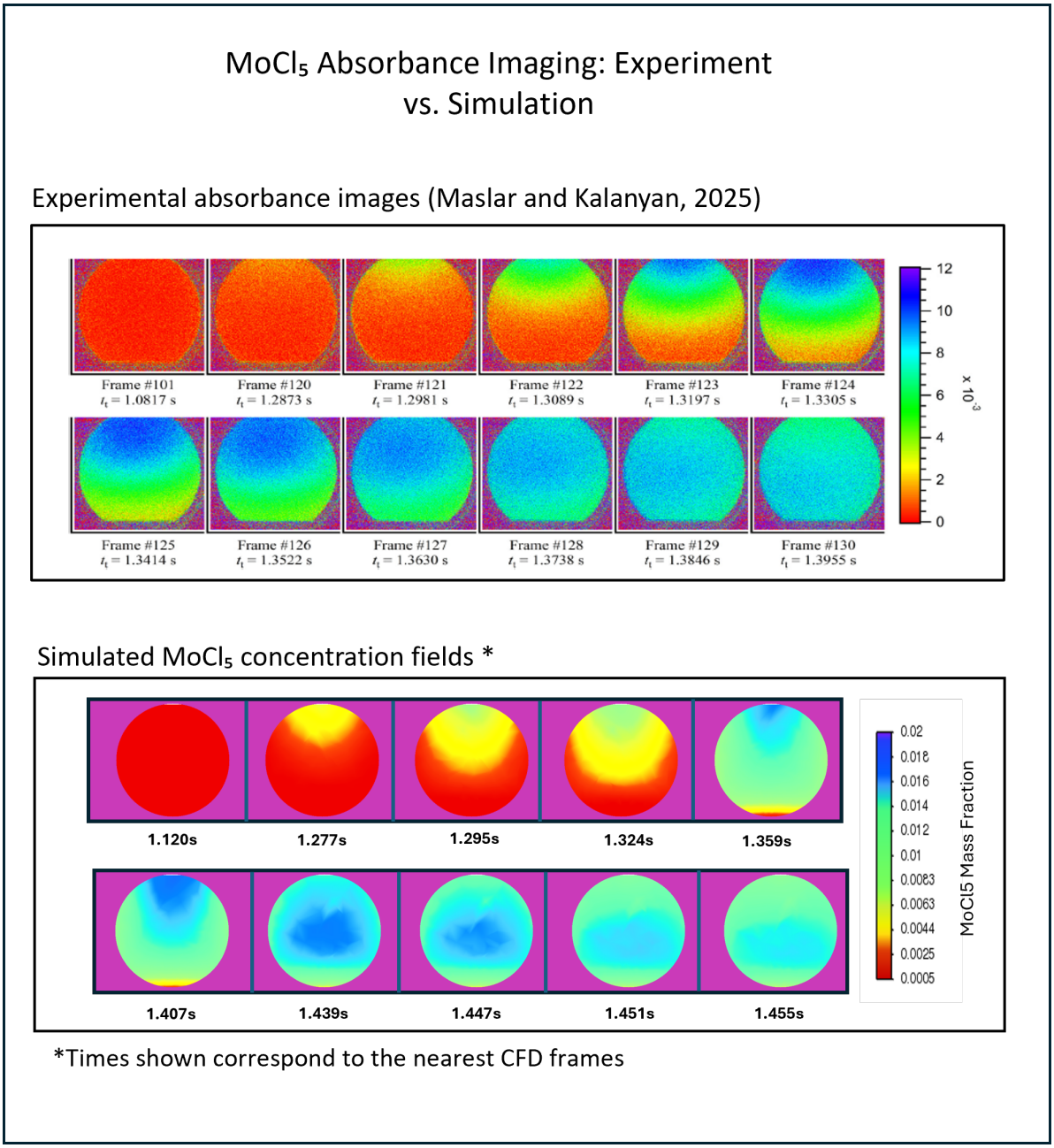


Figure 6.14: $MoCl_5$ image comparison between experiment and simulation during a pulsed injection sequence in the NIST optically accessible ALD reactor. **Top:** Experimental absorbance images acquired through the optical viewing window using a telecentric lens and CMOS camera (Maslar and Kalanyan, 2025). **Bottom:** CFD-predicted $MoCl_5$ concentration fields extracted from circular-masked mid-plane slices, aligned to match the camera field of view.

perturb chamber-scale hydrodynamics. Consequently, validating the CFD flow field with a heavy, non-reactive tracer reliably ensures accurate prediction of precursor transport and near-surface delivery for the ALD species used in this study.

The coupled multiscale simulation assumes the absence of gas-phase side reactions among precursors and co-reactants. This assumption is consistent with the fundamental operating principle of atomic layer deposition and atomic layer etching processes, in which reactive species are introduced sequentially rather than simultaneously, with inert gas purge steps separating each exposure. As a result, only a single reactive species is present in the reactor at any given time, and gas-phase reactions between precursors are intentionally suppressed by process design. Under these exposure-limited conditions, film growth and etching are governed by surface-mediated reactions, while gas-phase chemistry plays a negligible role. Potential gas-phase decomposition of individual precursors is minimal under the operating temperatures and residence times considered and does not contribute appreciably to material deposition or removal. Neglecting gas-phase side reactions therefore does not compromise the accuracy of the model for the surface-controlled ALD and ALE regimes investigated in this chapter and is consistent with established experimental interpretation and modeling practice in the ALD/ALE literature.

6.4 Multiscale Real-Time Simulation

Multiscale Integration of Microscopic and Macroscopic Models

After independently developing the microscopic Monte Carlo model and the macroscopic computational fluid dynamics (CFD) model, as well as verifying the accuracy of the CFD-based fluid

flow predictions, both components are combined into a real-time, coupled multiscale simulation framework. This integration forms the basis for a digital twin capable of resolving both surface kinetics and reactor-scale transport phenomena simultaneously. The motivation for such multiscale coupling lies in two key aspects: (1) the need to use real-time local pressure data to drive the microscopic surface reactions, and (2) the necessity of accounting for the effect of surface reactions as dynamic source terms within the CFD domain.

First, it is well established that the precursor partial pressure at the wafer surface does not instantaneously reach its target value upon dosing. Instead, there exists a finite transport delay governed by the reactor geometry, inlet configuration, and flow resistance. Many industrial ALD and ALE systems use showerhead or diffusive inlet structures to promote lateral gas uniformity [1, 135], but these structures inherently slow down the rate of precursor buildup. As a result, the local pressure on the wafer surface evolves dynamically, and assuming a constant or pre-defined pressure, as is commonly done in purely microscopic simulations, can lead to unrealistic kinetics. By feeding real-time pressure data from the CFD model into the Monte Carlo simulation, the multiscale model captures the spatial and temporal evolution of the precursor environment more accurately.

Second, the influence of surface reactions on fluid dynamics must be considered. Prior works have demonstrated coupled frameworks where surface kinetics are modeled using microscopic techniques and reactor flow is resolved using CFD [1]. However, in many such studies, the coupling is one-way: the CFD model is first solved independently, and the resulting pressure and temperature profiles are used as inputs to the microscopic simulation via interpolation. This neglects the feedback of surface consumption on precursor concentration near the wafer, which becomes especially important when reactions are spatially heterogeneous or kinetically fast. In practice, surface reac-

tions locally deplete precursor species, generate volatile products, and can alter the boundary layer flow, meaning that their effect should be reflected back in the CFD simulation via a surface-based source term.

The full multiscale simulation loop is illustrated schematically in Figure 6.15. Each cycle begins with a steady-state CFD simulation under N_2 purge flow, which is saved and used to simulate post-purge initial conditions. The simulation proceeds as follows:

1. Read the instantaneous facet-average pressure across the wafer surface from the CFD field.

2. Use the local partial pressure of reactants as input to the microscopic simulation to compute real-time reaction rates.

3. Run the Monte Carlo surface simulation over a short time interval (0.1 s in this chapter) to calculate the normalized coverage increment for each surface region.

4. Convert the coverage increment into a surface reaction source term using:

$$S = \frac{\Delta\theta \cdot Y_{\max}}{\Delta t} \tag{6.13}$$

   where $\Delta\theta$ is the change in normalized coverage, $Y_{\max}$ is the maximum molecular yield per unit area, and $\Delta t = 0.1$ s.

5. Inject the species source term back into the CFD model via a user-defined function (UDF), assigning it to mesh cells directly above the wafer using a user-defined memory (UDM) scheme to identify the corresponding surface locations. The thermal source from surface reactions is ignored, and the consistent isothermal operating condition is assumed.

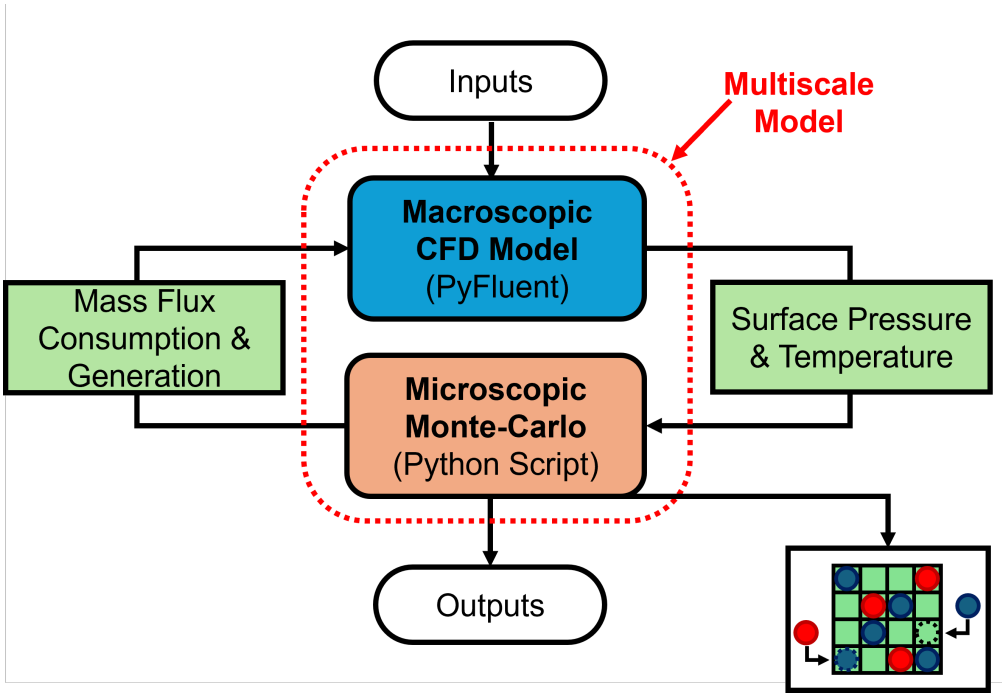


Figure 6.15: The schematic diagram of the multiscale simulation. The CFD program is started and controlled in PyFluent environment, sending temperature and pressure to microscopic python scripts, and receive the outputs of microscopic model as sources of species.

This cycle is repeated iteratively to produce a time-resolved coupling between surface chemistry and reactor-scale fluid dynamics. The choice of 0.1 s for the coupling interval reflects a balance between numerical stability and physical fidelity. The selection of a 0.1 s coupling interval represents a practical compromise between temporal resolution and the intrinsic stochastic resolution of the microscopic Monte Carlo surface model. Although this interval is smaller than the characteristic reactor residence time (approximately 0.8 s), further reduction of the coupling step does not proportionally improve accuracy in the present framework. The microscopic model operates on a discrete surface lattice and advances time through stochastic reaction events, with typical individual reaction time increments already on the order of several milliseconds. Under these conditions, the dominant source of temporal uncertainty arises from the finite spatial resolution and statistical variance of the Monte Carlo simulation rather than from the CFD–MC coupling frequency. Over-refinement of the coupling interval would therefore primarily increase computational cost and stochastic noise without yielding commensurate gains in physical fidelity. The chosen interval is sufficient to resolve the gradual precursor delivery transient at the reactor scale while remaining consistent with the effective temporal resolution of the surface kinetics model. Achieving stricter time-step independence would require substantially increasing the microscopic model resolution (e.g., larger surface grids or ensemble averaging), which is computationally prohibitive at present. Given that the NGA region contains approximately 2500 reactive sites and the GA contains around 3840 CFD cells, individual reaction time increments are typically on the order of 10^{-3} s, so the 0.1 s interval allows for sufficient reaction resolution while maintaining acceptable statistical variance inherent to the stochastic Monte Carlo simulation.

In the present multiscale framework, heat generation from surface reactions is neglected and

the reactor is treated as isothermal. This assumption is justified by both the process chemistry and the operating conditions considered in this chapter. Following the revision of the etching mechanism, the ALE step is formulated as a fully thermal, HF-based process without plasma excitation. Under these conditions, only a single molecular layer reacts per cycle, and the total amount of reacted material is extremely small relative to the thermal mass of the wafer, chuck, and reactor hardware. In industrial ALD and thermal ALE systems, the wafer temperature is actively controlled and maintained under near-isothermal conditions, such that any heat released by surface reactions is rapidly dissipated and does not result in measurable local temperature excursions. As a result, reaction heat does not significantly perturb surface kinetics, precursor transport, or selectivity in the present process window. Neglecting reaction heat is therefore a reasonable and widely adopted approximation in ALD and thermal ALE modeling, and does not affect the qualitative or quantitative conclusions of this study.

Through this multiscale framework, we realize a fully coupled, time-resolved digital twin of the ASALD process. The model faithfully captures both stochastic reaction dynamics and reactor-scale transport effects, enabling predictive analysis and optimization of self-aligned deposition and etching cycles under realistic industrial conditions.

6.5 Results and Analysis

This section presents the simulation results obtained from both the microscopic and multiscale ASALD models, focusing on the evolution of surface coverage, film thickness, and selectivity across multiple deposition-etching cycles. The results demonstrate the capability of the predictive model

to capture key physical trends and process sensitivities observed in ASALD systems.

The analysis begins with single-batch microscopic simulation results in section section 6.5.1, including both the first and second batches with sufficient reaction time for saturated reaction. While the first batch illustrates the initial surface adsorption and reaction dynamics, the second batch is critical for revealing the onset of nucleation accumulation on NGA. This comparison highlights how partially formed nuclei from earlier batches serve as reactive sites for continued growth, and clarifies the mechanism by which selectivity is gradually lost. Including the second batch also serves to establish how the multi-batch Monte Carlo scheme captures surface history and accumulation effects over time. Following this, multi-batch microscopic simulation results without any etching steps are provided to show how selectivity degrades as the number of cycles increases. The model predicts a characteristic nucleation delay period on the NGA, followed by accelerated growth that ultimately compromises selective deposition. Subsequent simulations introduce etching into the cycle and analyze the effect of different etch times on long-term selectivity. The results would reveal the dependence of batch-wise selectivity on etch duration.

The final part of the results examines the behavior of the multiscale model in section section 6.5.2, which integrates real-time precursor pressure evolution with microscopic surface kinetics. Comparisons between multiscale and pure microscopic simulations at fixed etch durations show that the multiscale model predicts a different effective etch rate under otherwise identical conditions, attributed to the finite pressure delivery and development time within the reactor.

6.5.1 Multi-Batch Microscopic Simulation

The NGA after 3.0s of inhibitor adsorption at $P_{Hacac} = 300$ Pa is shown in Figure 6.16 The orange-

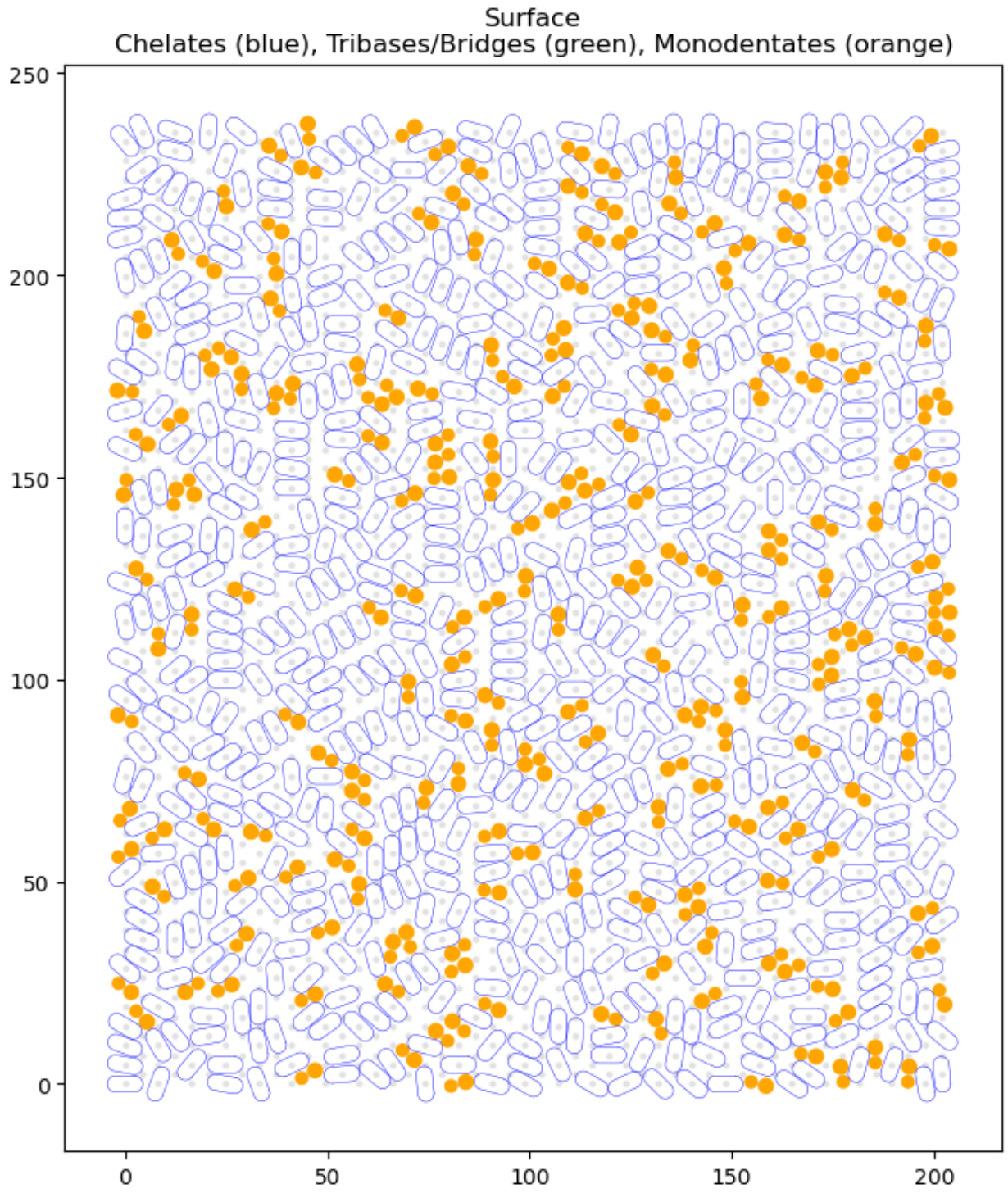


Figure 6.16: Inhibitor adsorption on NGA on bare grid. Orange molecules are monodentates, blue molecules are chelates.

filled geometries represent monodentate inhibitors, while the blue-outlined geometries correspond to chelate structures. The simulation shows that monodentates comprise approximately $22 \pm 2\%$ of the total adsorbed molecules, which is in good agreement with the experimentally reported fraction of $20 \pm 1.5\%$ [103]. The overall inhibitor density is found to be 1.92 molecules per square nanometer, which is slightly higher than the experimentally observed value of 1.7 molecules per square nanometer [103]. This deviation can be attributed to minor differences in site spacing used during surface grid generation. Nonetheless, both the fractional and total coverage values fall within a reasonable range of the experimental data, supporting the physical validity of the surface adsorption model.

The precursor adsorbed on the NGA after 3.0s reaction at $P_{BDEAS} = 300$ Pa is shown below in Figure 6.17 The green circle is the final silane bridge structure, and the trilate structure (three circles) is the intermediate silane-DEA structure where only one DEA molecule is eliminated and cannot form the bridge structure either because the adjacent sites are not available or the formed bridge would collide with other molecules. To calculate the normalized coverage, a simulation of direct BDEAS adsorption on NGA surface without inhibitor is conducted and shown in Figure 6.18. The average normalized coverage obtained from multiple simulation runs with different random seeds is approximately $6.7 \pm 1.2\%$, whether considering the total normalized coverage or only the coverage at bridge endpoints. This value is in good agreement with the experimentally reported normalized coverage of 8% [103], further validating the accuracy of the microscopic model in capturing unwanted precursor nucleation on NGA. For the BDEAS adsorption on GA, the result is demonstrated in Figure 6.19 As illustrated in Figure 6.19, the GA surface exhibits a geometric limitation due to its two-column arrangement. Specifically, if a given site is between an

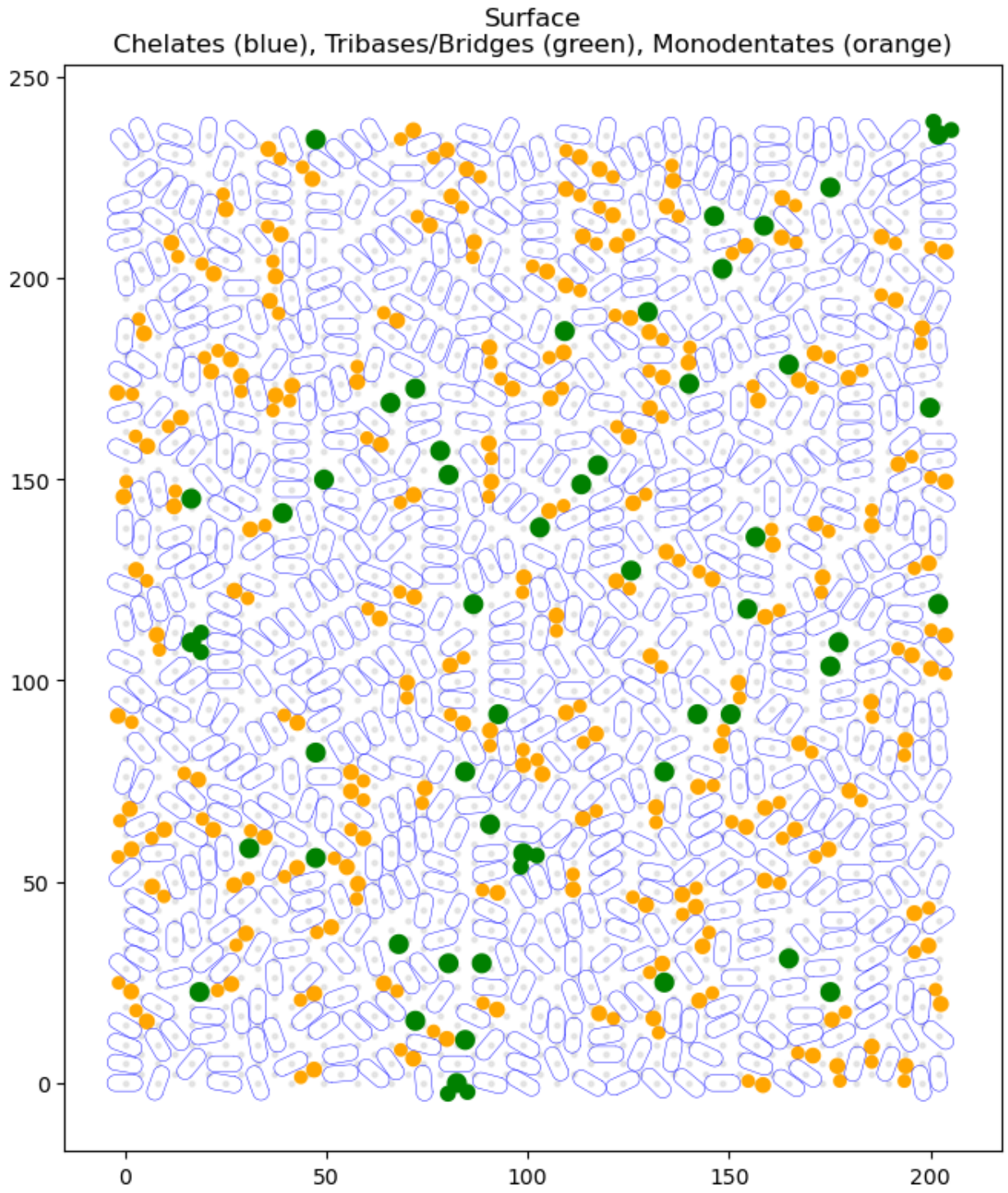


Figure 6.17: Precursor substitution on inhibitor-adsorbed NGA surface. The green circle is the final Silane bridge structure, the green trilate is the intermediate Silane-DEA structure.

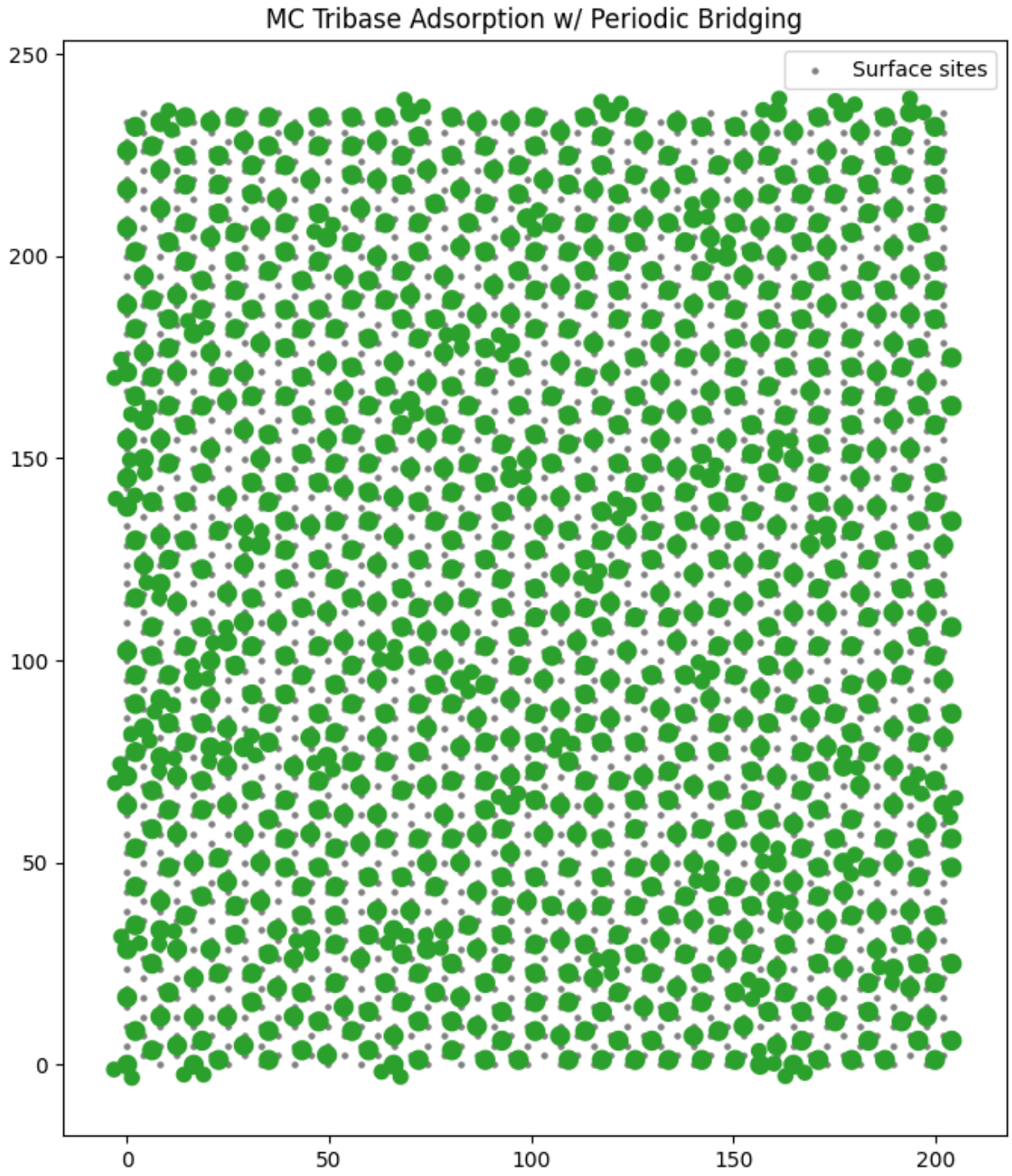


Figure 6.18: Normalized benchmark of BDEAS direct adsorption on NGA surface, with 898 total sites adsorbed and 838 of them are end points of bridge structure.

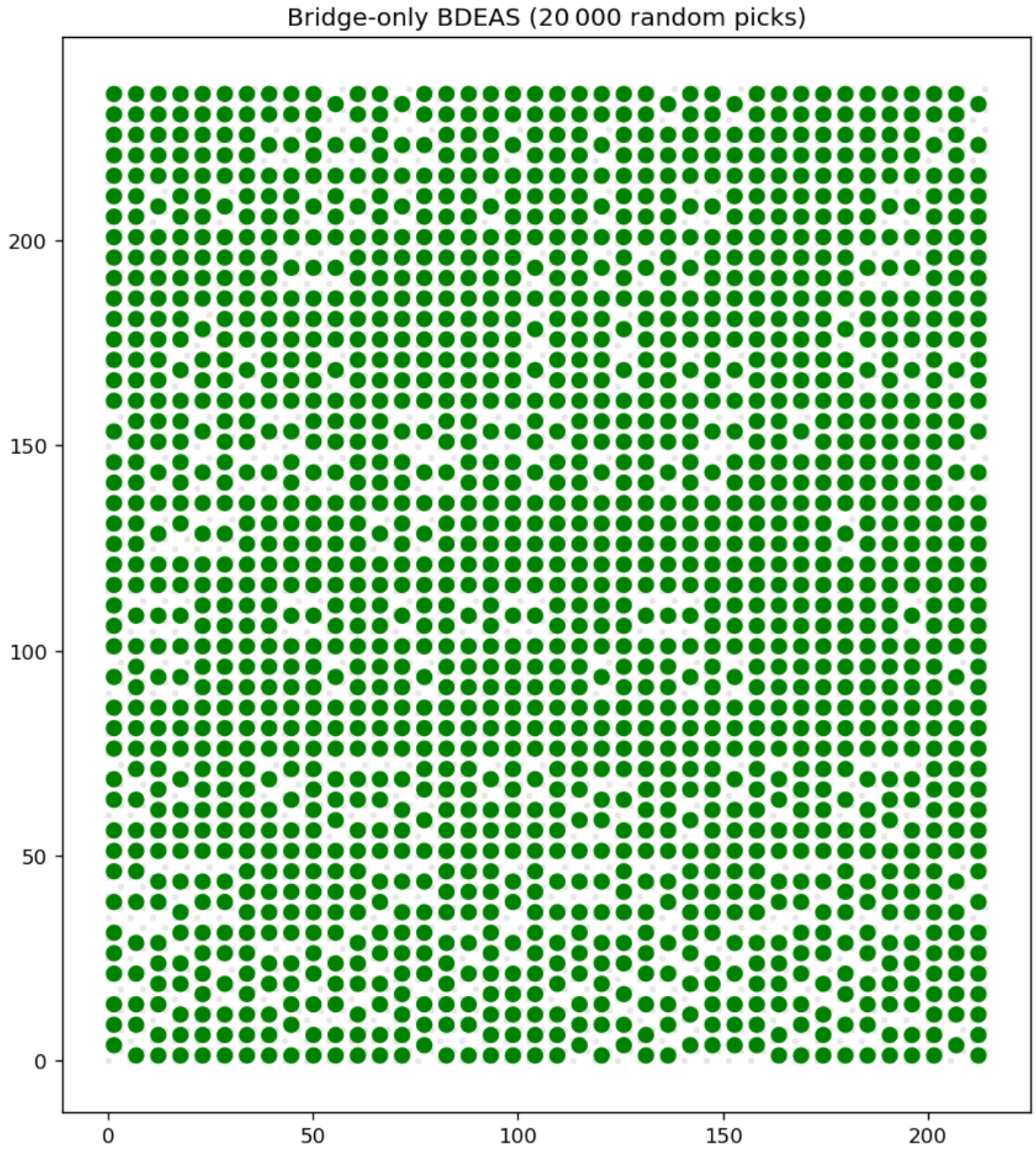


Figure 6.19: BDEAS precursor adsorption on GA with the microscopic mechanism described in [3]. The surface grid are separated into 2-column groups and the bridge can only formed between the two columns in each group.

existing bridge structure formed by the two sites directly above and another formed by the two sites directly below, it becomes isolated and incapable of forming a bridge with any neighboring site. This constraint results in so-called dead points where no further precursor attachment can occur. Experimental studies have reported that the actual surface coverage on the GA ranges from 86% to 94%. The simulation presented in Figure 6.19 yields a final coverage of 90%, which lies well within the experimentally measured range, further supporting the accuracy and physical relevance of the microscopic model.

The grid after Step C, the ozone oxidation for 1.0s at $P_{O_3} = 60$ Pa is shown in Figure 6.20 The ozone oxidation step does not alter the overall deposited layer structure. Its primary function is to oxidize the surface-bound silane species, converting them into hydroxyl-terminated SiO_2 structures. Simultaneously, the ozone removes residual species such as inhibitors on the NGA and unreacted trilate precursors, thereby preparing the surface for the subsequent ALD batch.

If no etching step is applied, the process proceeds directly to a second inhibitor adsorption cycle. To illustrate the multi-batch simulation mechanism under this condition, the second round of Step A is shown in Figure 6.21. The second inhibitor adsorption cycle illustrates how inhibition proceeds in the presence of pre-existing SiO_2 nucleation sites on the NGA surface. Due to the reduced available surface area caused by these existing deposits, the total number of adsorbed inhibitor molecules decreases from 936 in the first cycle to 901 in the second cycle. Moreover, as monodentate molecules occupy less surface area than chelate structures, their relative proportion increases under increasingly constrained spatial conditions. This shift in molecular distribution leads to a sustained rate of contamination and nucleation accumulation on the NGA, as the absolute number of monodentate molecules remains relatively stable during the early batches. Although the

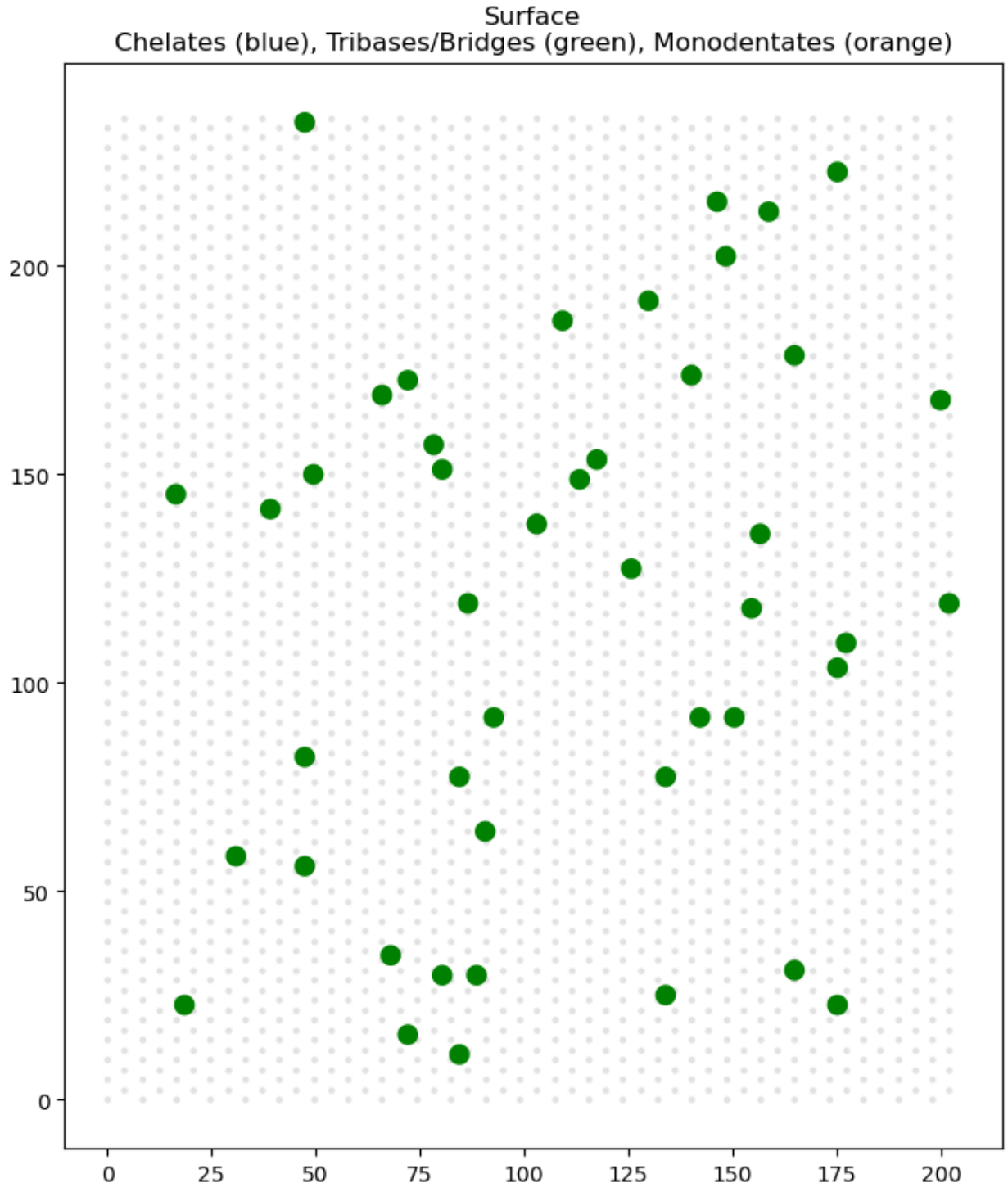


Figure 6.20: The ozone would complete the deposition of the deposited silane structure by converting the -H to -OH structure, at the same time strip off all other adsorbed inhibitor molecules and trilate molecules to prepare the surface for next batch.

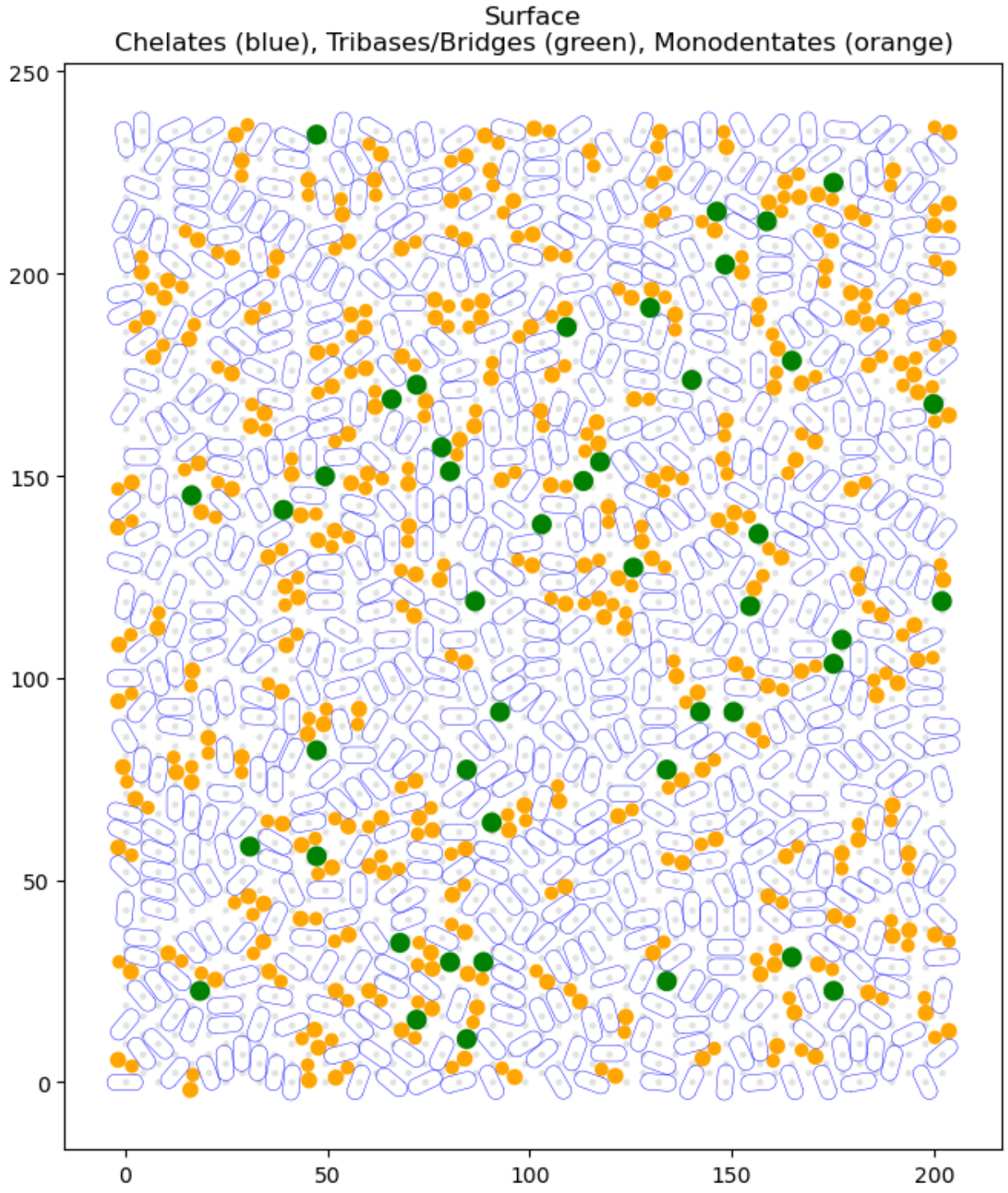


Figure 6.21: Second round of inhibitor adsorption on NGA surface. The pre-existing deposited SiO_2 takes some space that reduces the total number of inhibitors on surface and increases the proportion of monodentate inhibitor as the monodentate takes less spaces on the surface.

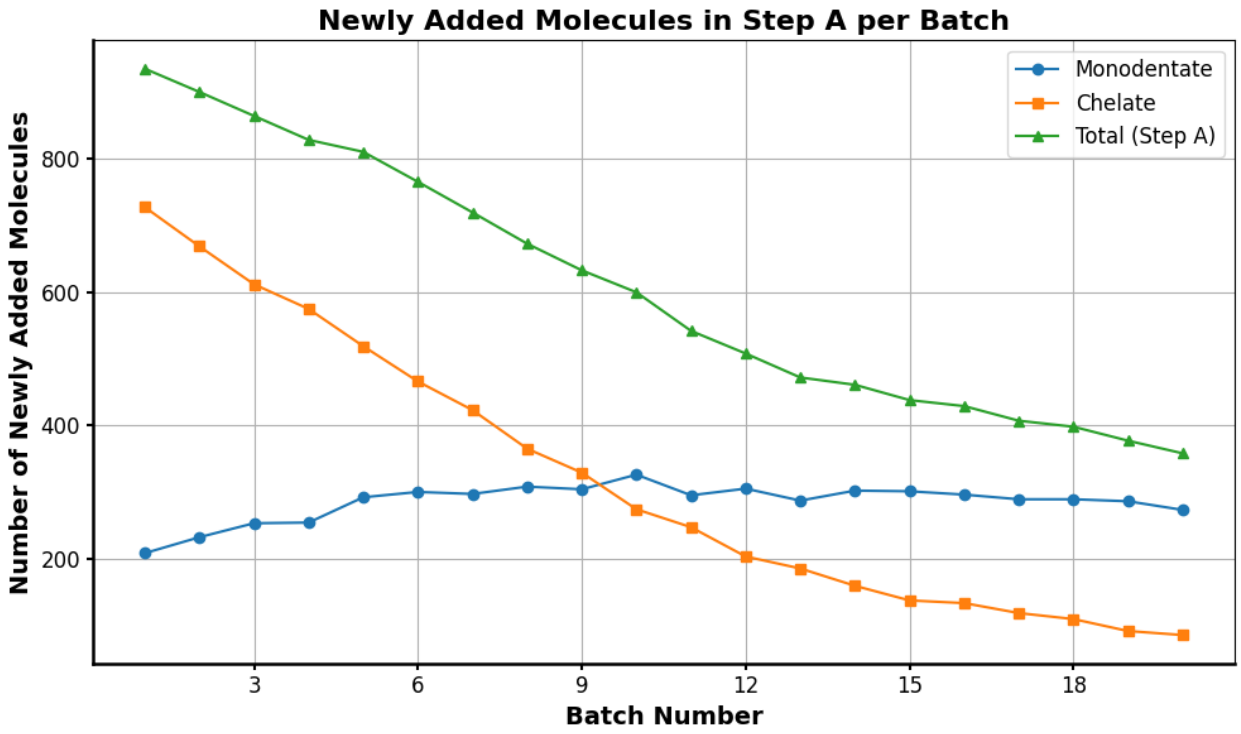


Figure 6.22: Number of monodentates and Chelates added in first 20 batches. Monodentate number doesn't change too much because of reduction in spaces. Chelates and total adsorbed molecule number decrease quickly with batches.

total number of inhibitors declines with each cycle, the rising fraction of monodentates compensates by maintaining available nucleation sites. The evolution of monodentate and chelate counts over the first 20 batches is shown in Figure 6.22. As a reference, the surface grid status after the second round of Step B and Step C is shown in Figure 6.23 and Figure 6.24.

Under the precursor accumulation without etching, the NGA thickness, GA thickness and selectivity for the first 40 batches are shown below in Figure 6.25.

The multi-batch simulation results demonstrate that, in the absence of an etching step, the selectivity cannot reach the threshold of 0.99 typically required for semiconductor manufacturing. Selectivity declines rapidly over successive batches; by the end of 40 batches, it approaches zero.

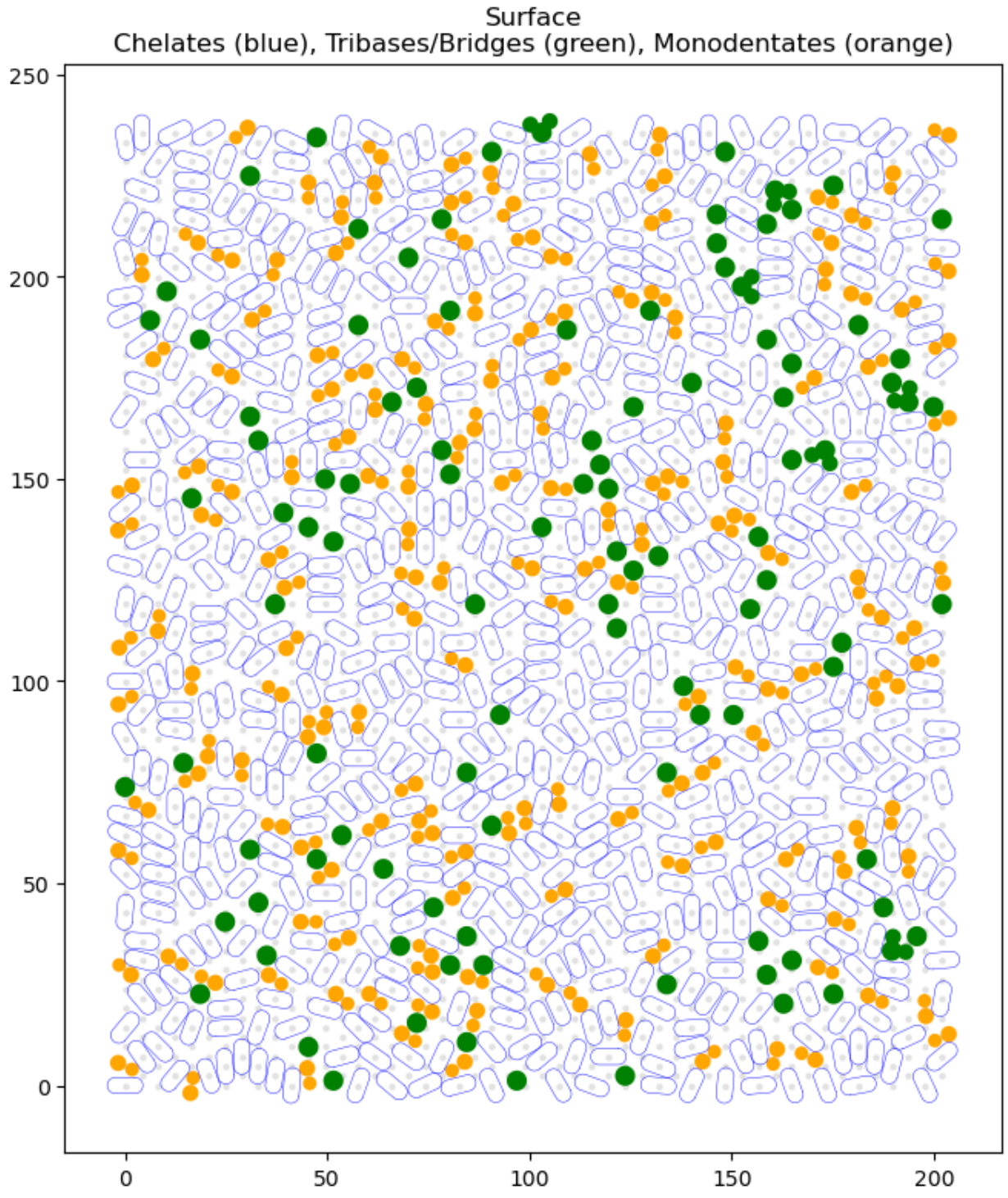


Figure 6.23: Second round of Step B, more contamination is formed on NGA surface.

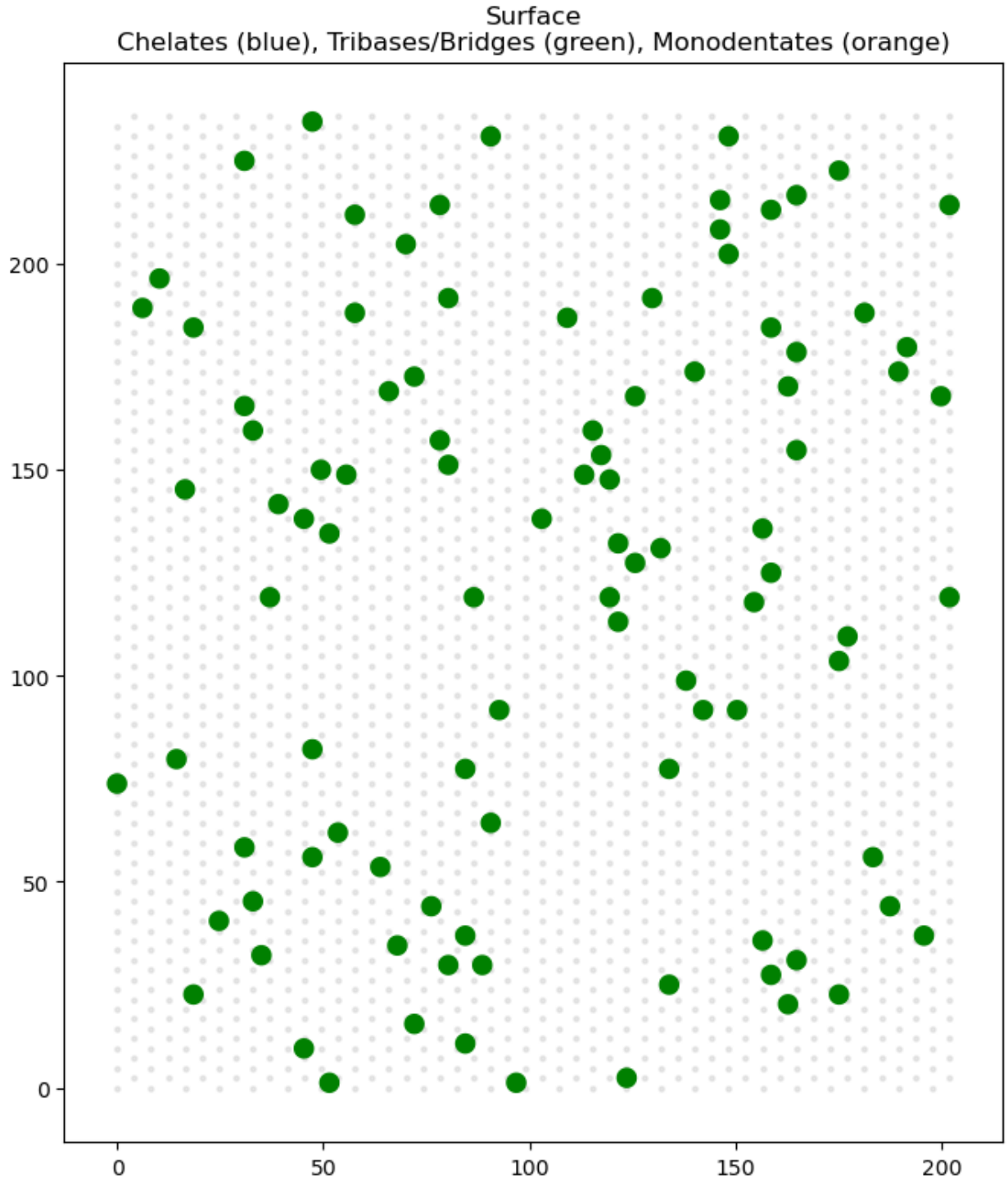


Figure 6.24: Second round of Step C, more unwanted nucleation is completed on NGA surface.

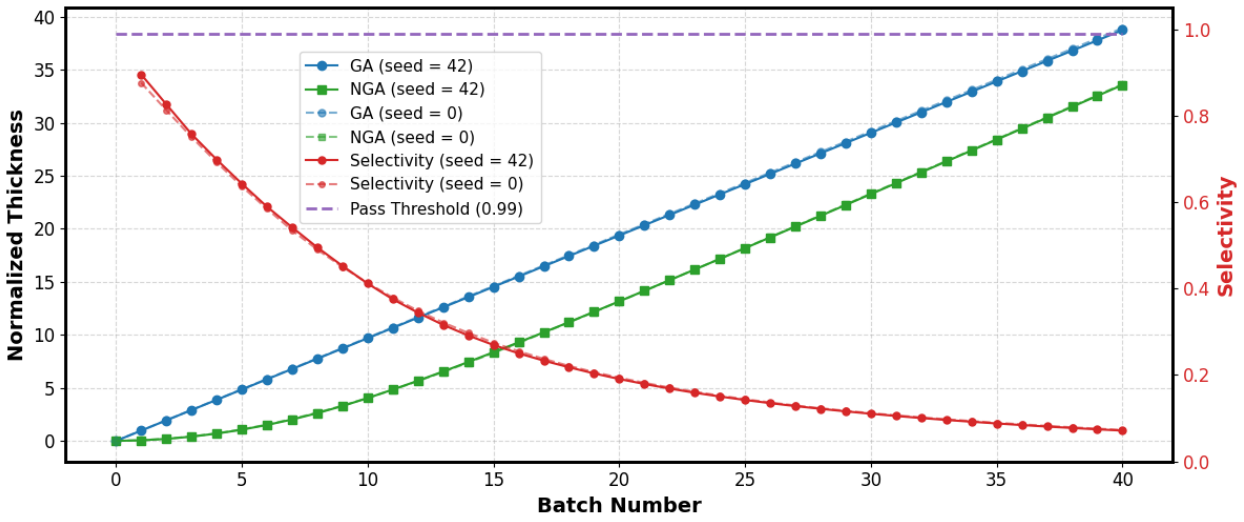


Figure 6.25: Pure microscopic simulation multi-batch result without etching. After a nucleation delay of about 10-15 batches, the NGA has identical deposition rate as the GA. The selectivity never achieves pass criterion (0.99) and decreases very quickly. The comparison between random seed of 42 and 0 shows the robustness and low variance of this simulation method under 50×50 scale.

This loss in selectivity is attributed to the progressive accumulation of unwanted nucleation on the NGA. After an initial nucleation delay of approximately 10–15 batches, the growth rate on the NGA becomes comparable to that on the GA, effectively resulting in the complete transformation of the Al_2O_3 NGA into SiO_2, indistinguishable from the GA. This convergence underscores the necessity of introducing an etching step after each batch to remove unwanted deposition from the NGA while preserving most of the material grown on the GA. The test of different random seed proves the low variance and robustness of the simulation method. The following simulations use random seed 42 as default value.

Etching results for durations of 0.6 s, 0.8 s, and 1.0 s under $P_{TMA} = 300$ Pa, $P_{HF} = 60$ Pa and $T = 523\ K$ are presented in Figures 6.26, 6.27, and 6.28, respectively. A comprehensive overview of the GA thickness, NGA thickness, and resulting selectivity as functions of batch number and

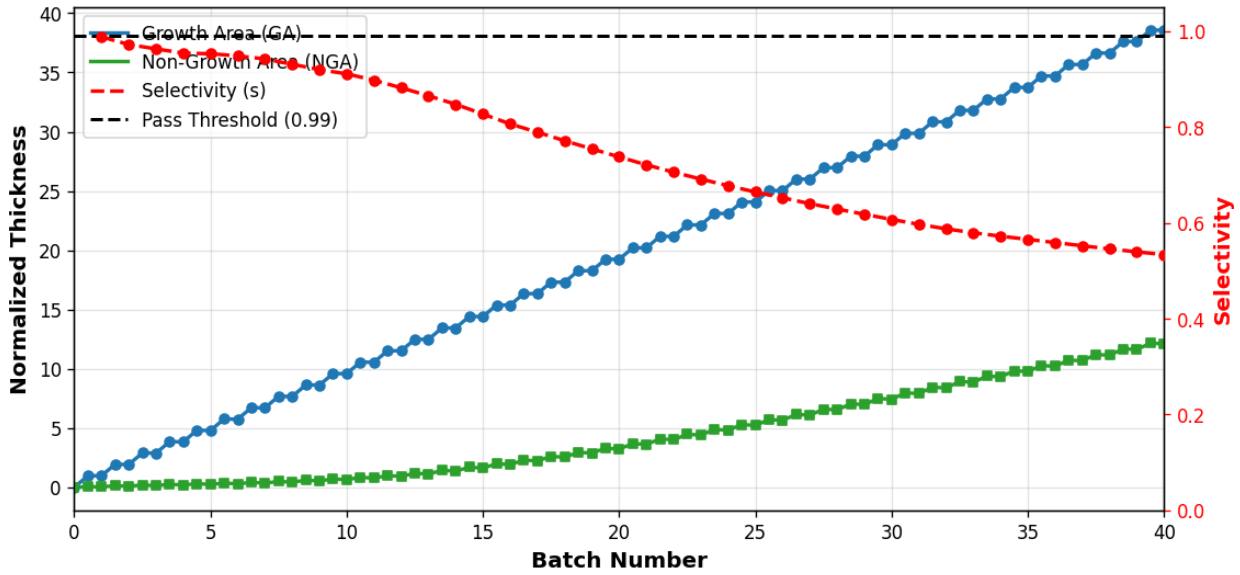


Figure 6.26: Under fixed 0.6s etching time, the selectivity cannot be maintained above pass criterion (0.99) at anytime, which cannot fulfil any thickness requirement.

etch time is shown in Figure 6.29. At an etch time of 0.6 s, the selectivity cannot maintain above the pass criterion at any time, which cannot fulfil the requirements of any application. At an etch time of 0.8 s, the selectivity remains above the 0.99 threshold for approximately 5 to 10 batches before declining rapidly, which can fulfil the requirements for very thin film on GA. In contrast, the 1.0 s etch time maintains selectivity above the pass criterion throughout all 40 simulated batches, indicating that this duration is saturated with respect to effective removal of unwanted NGA growth. Increasing the etch time beyond 1.0 s yields diminishing returns, as it further reduces deposition efficiency on the GA without improving selectivity. This saturation point represents a critical trade-off in the deposition-etching sequence. Etching on the growth area is reduced with increasing batch number, which is observed on all three cases, are attributed to the roughness development on surface that decreases the site exposure.

It should be noted that this critical etch time is not universal and may vary depending on the

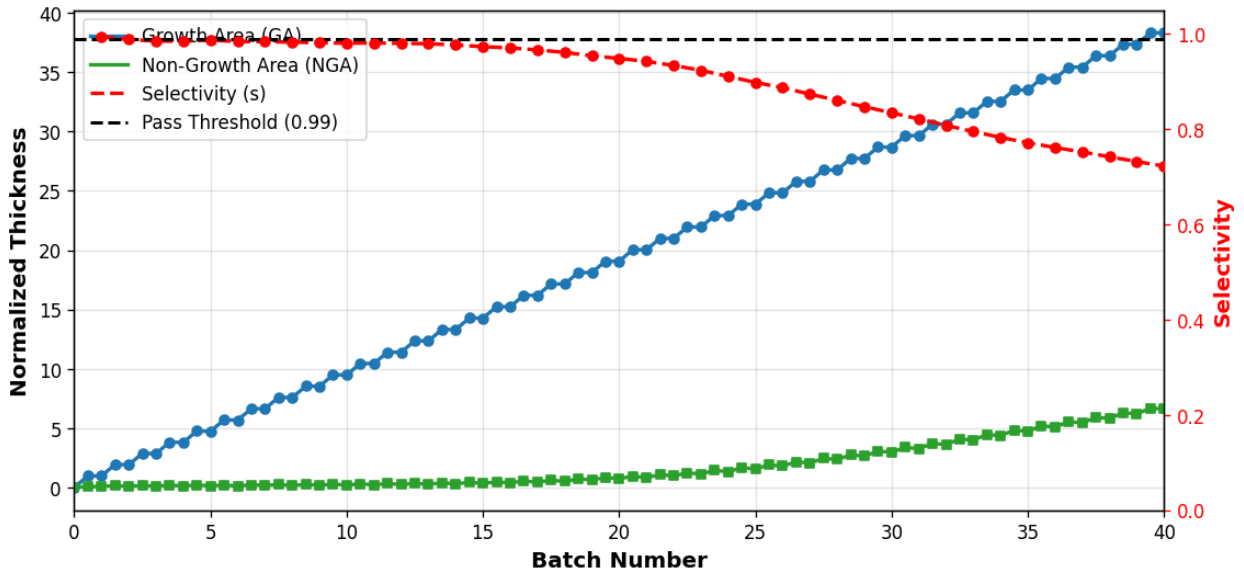


Figure 6.27: Under fixed 0.8s etching time, the selectivity can be maintained between 5-10 batches, which is suitable for very thin film requirement on GA, but for thicker film that requires over 10 layers the selectivity cannot fulfil the requirements.

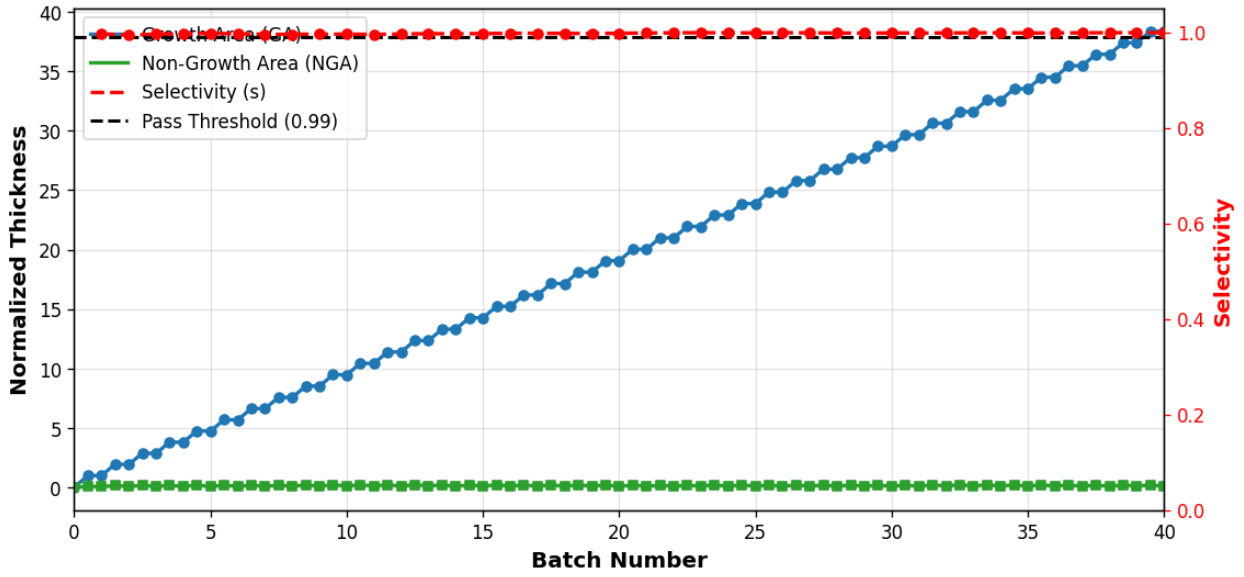


Figure 6.28: Under fixed 1.0s etching time, the selectivity is maintained for the whole 40 batches at perfect level, which indicates the etching on NGA is almost saturated.

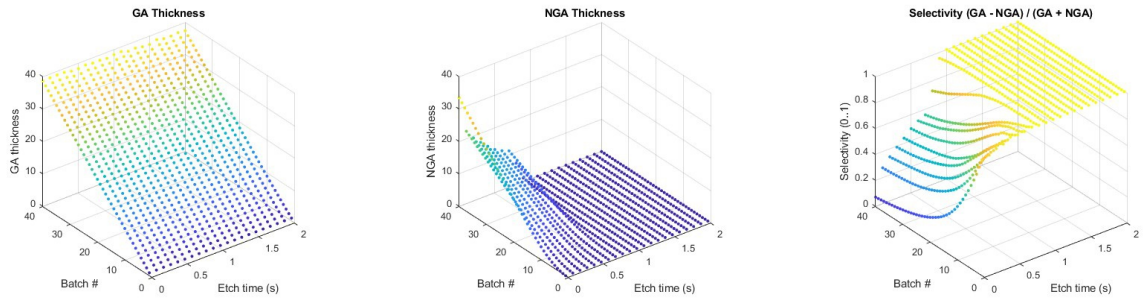


Figure 6.29: Holistic view of the GA/NGA thickness and selectivity at different batch numbers and etch times. The Selectivity improves very quickly with increased etch time and saturates at about 1.0s. The GA thickness decreases with higher etch time and shows nucleation delay-like behavior at long etch times because of surface roughness developed by etching.

specific reaction chemistry and operating conditions. The optimal etch duration must therefore be determined on a case-by-case basis. To emphasize the impact placed by exposure mechanisms introduced in this chapter, the results of 0.6s and 1.0s etching without exposure impact is shown below in Figure 6.30. The final average layer number on GA is about 6.4% lower than the case exposure is actively involved (38.37 v.s. 35.90), the final average layer number on NGA is about 17% lower (12.11 v.s. 10.03), which results in apparent increase in selectivity that have the risk of misguiding the implementation.

6.5.2 Multiscale Simulation

The pure microscopic simulation framework provides insight into surface dynamics under idealized conditions, assuming that the partial pressures of all precursor species at the wafer surface remain constant throughout the reaction cycle. This simplification neglects the transient nature of gas-phase transport within practical atomic layer deposition (ALD) systems, where precursor delivery from the inlet to the wafer surface occurs over finite time and is influenced by reactor geometry, flow resistance, and purge cycles. As a result, purely microscopic models cannot capture the dynamic

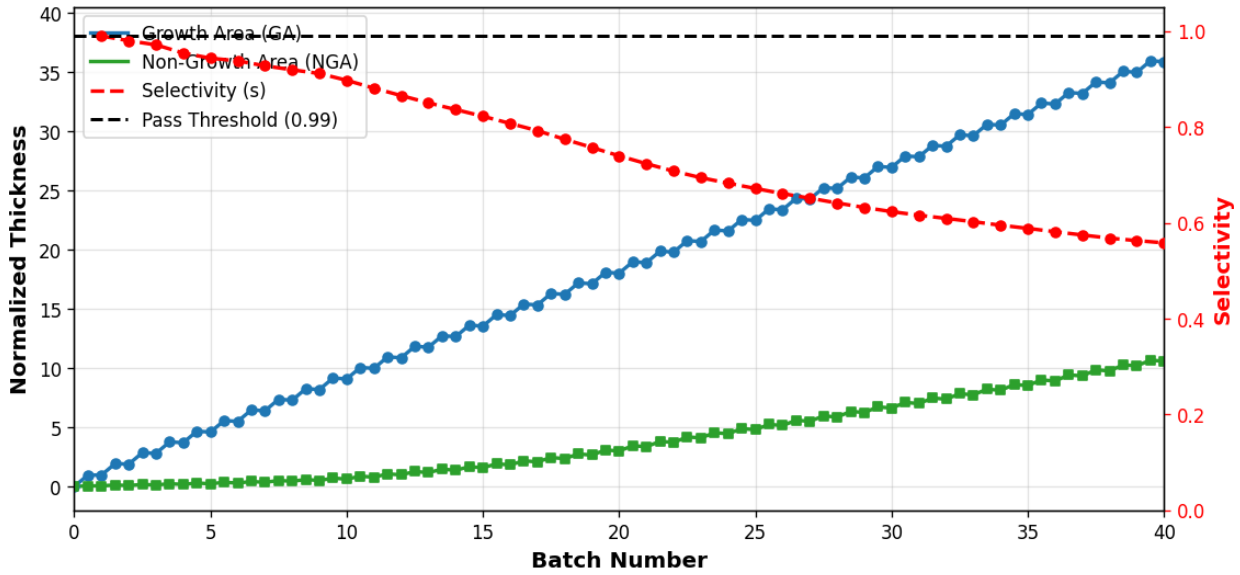


Figure 6.30: Without the involvement of exposure mechanism, the etching is more aggressive on both surfaces, which can be inferred by comparing with Figure 6.26 whose final average layers are higher on both GA and NGA than in this figure. Without exposure involvement, the selectivity is also apparently higher, which can misguide the implementation.

coupling between gas transport and surface reactions observed in real reactors.

Although atomic layer deposition and atomic layer etching are governed by self-limiting surface reactions, self-limitation does not imply instantaneous, spatially uniform, or cost-free reactant delivery. In practical reactor environments, the rate at which self-limited reactions approach saturation is determined by the local precursor partial pressure at the wafer surface, which evolves dynamically due to finite transport, mixing, and purge processes. Industrial ALD and ALE systems operate under strict constraints on throughput, precursor utilization efficiency, and cost, such that exposure times cannot be arbitrarily extended to guarantee saturation. Instead, the objective is to identify the minimum injection and dwell times required to achieve complete surface reaction across the entire wafer. Under these conditions, delays in precursor arrival, pressure non-uniformity, and transport-limited regions can significantly increase the effective saturation time, even for nom-

inally self-limited reactions. The CFD-based gas-flow simulations in this chapter explicitly resolve the transient and spatially resolved delivery of reactants to the wafer surface, enabling quantitative prediction of the time-dependent surface pressure that governs reaction kinetics. When coupled with the microscopic Monte Carlo model, this multiscale framework reveals how reactor geometry, flow configuration, and operating conditions translate into required exposure times for full coverage, rather than assuming idealized, instantaneous pressure equilibration. As demonstrated in the results that follow, neglecting gas transport leads to systematic underestimation of the required reaction time for time-sensitive steps such as ALE surface modification, whereas incorporating realistic gas delivery dynamics enables accurate correction of exposure times needed to maintain selectivity under industrially relevant conditions.

To address this limitation, the multiscale simulation integrates the real-time pressure and temperature profiles obtained from the macroscopic CFD model as inputs to the microscopic Monte Carlo surface simulation. In turn, the microscopic model provides time-dependent reaction source terms—such as precursor consumption or byproduct generation—that are fed back into the CFD domain. This bidirectional coupling enables the model to capture how surface reactions influence, and are influenced by, the evolving fluid flow field within the reactor. The following results highlight the differences between the idealized microscopic model and the fully coupled multiscale framework and demonstrate how pressure transients and reaction feedback alter the evolution of selectivity and film growth over multiple ALD cycles. The influence of multiscale simulation is not only on Step D, the major etching step, but on all steps. The multiscale simulation inlet mole

fraction is set to be:

$$f_{in} = \frac{P_{micro}}{P_{OP}} \tag{6.14}$$

Where f_{in} is the inlet precursor mole fraction, P_{micro} is the microscopic simulation pressure, and P_{OP} is the operating pressure set in CFD simulation. The P_{OP} applied in this chapter is 600 Pa, near vacuum, the typical working condition for ALD/ALE processes, and the operating temperature is 523 K. This setup can ensure a steady-state precursor partial pressure in multiscale simulation identical with the pressure applied in microscopic simulation, but with a development and delivery process. The comparison of coverage progress on the first batch between microscopic and multiscale is shown below in Figure 6.31 with $f_{in,Hacac} = f_{in,BDEAS} = f_{in,TMA} = 0.5$, $f_{in,O_3} = f_{in,HF} = 0.1$. Due to the time update algorithm defined in Equation 6.3, the reaction time increment is inversely proportional to the reaction rate. Since the reaction rate itself is a function of local precursor partial pressure in Equation 6.4, extremely low pressures at the start of the reaction can result in artificially large time increments, thereby underestimating reaction activity in the initial phase. To address this issue, a threshold pressure is introduced to prevent premature time advancement under negligible reaction conditions. In this chapter, the threshold is set to 20% of the final target partial pressure. This approach is commonly adopted in multiscale Monte Carlo simulations to suppress the effect of initial pressure near zero [1, 114]. The pressure threshold employed in the multiscale simulation is introduced as a numerical stabilization mechanism rather than a physically intrinsic parameter. During the initial precursor delivery transient, the local partial pressure at the wafer surface can be several orders of magnitude lower than its steady-state value. Directly applying the stochastic time-advancement scheme under such conditions can lead to unphysically large time increments that

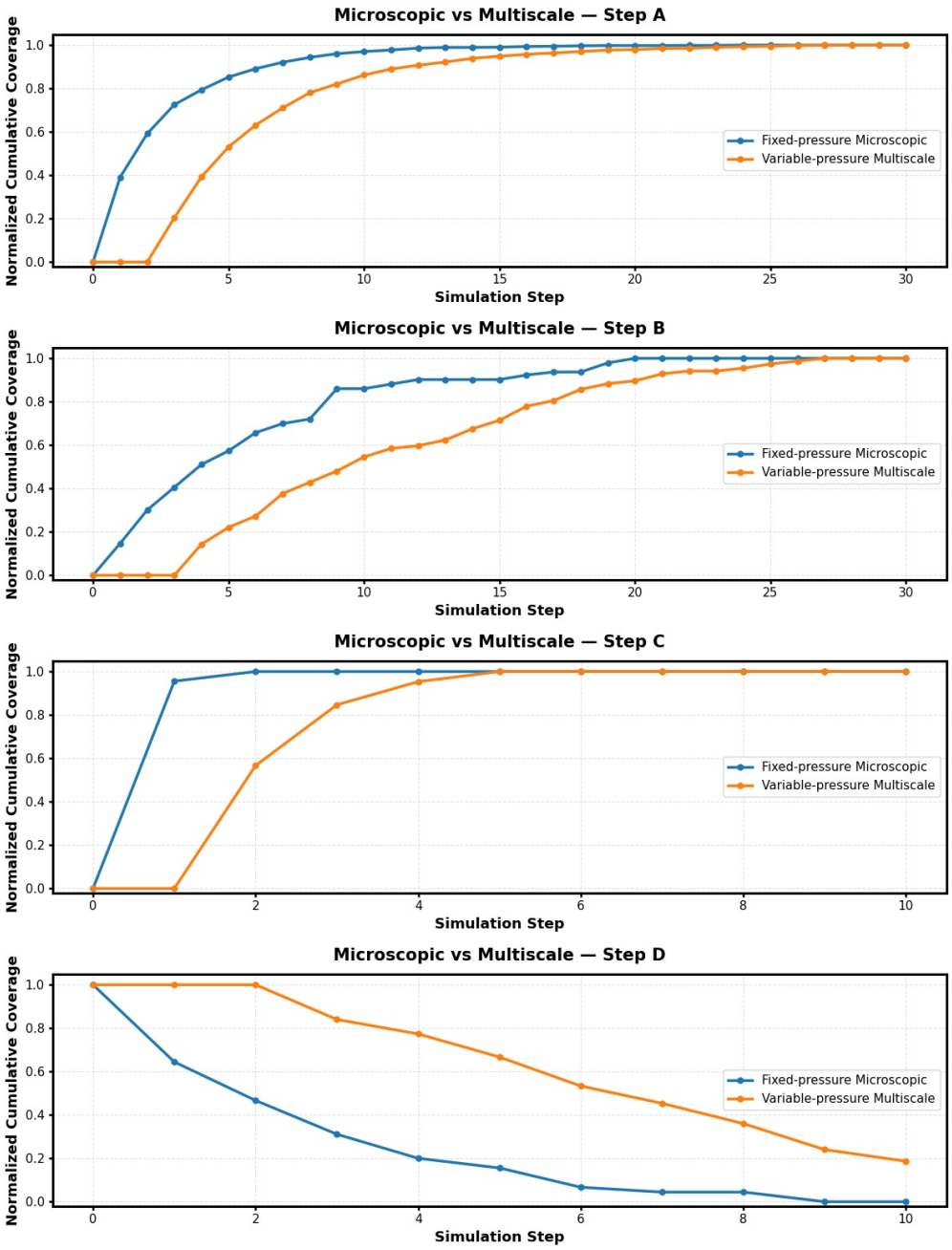


Figure 6.31: The comparison on first batch between microscopic and multiscale simulations. Because of the pressure threshold, the reaction starts with a lag and because of lower partial pressure of precursors, the initial slope of all steps is smaller for multiscale simulations.

do not correspond to meaningful surface reaction progress. To avoid this artifact, a lower-bound threshold on the local precursor partial pressure is imposed, below which surface reactions are temporarily suppressed. In this chapter, the threshold is set to 20% of the final target partial pressure, consistent with prior validated multiscale ALD [3] and ALE studies where this value yielded reaction completion times and saturation behavior in good agreement with experimental observations. While the precise numerical value of the threshold influences the absolute onset time of surface reactions, it does not alter the qualitative trends or conclusions regarding selectivity evolution or the relative correction required for the etching time. As is common in industrial process modeling, such calibrated numerical parameters are used to ensure numerical robustness and physical consistency when experimental reactor-scale transients are partially characterized or unavailable, and are subsequently refined as additional experimental validation becomes available.

As a consequence of this pressure-based thresholding, the multiscale simulation captures a delayed onset of surface reactions relative to the pure microscopic model, which assumes ideal and instantaneous pressure delivery. This effect manifests as a lag in reaction initiation and a smaller initial slope in surface coverage or thickness evolution plots for the multiscale case because of the lower pressure. To ensure complete conversion in Steps A, B, C, and E under this delayed regime, sufficiently long reaction durations are applied to allow full surface transformation.

However, Step D—the primary etching stage—requires precise control over reaction time to achieve selective removal without over-etching. For example, at an etch duration of 1.0 s, the microscopic simulation predicts complete removal of unwanted SiO_2 on the NGA after the first batch. In contrast, the multiscale simulation under the same etch time shows remnant material on the NGA surface due to the delayed reaction onset. These remnants can persist and accumulate in subse-

quent batches, ultimately degrading selectivity. The multiscale simulation result under the 1.0 s etch condition is illustrated in Figure 6.32, highlighting the need for time-adjusted etch strategies in multiscale modeling as when considering the precursor delivery in the industrial reactor, the selectivity can just be maintained for 18 batches then decreases significantly with increasing batch, compared to perfect selectivity of 1.0s etching in the microscopic simulation. To deal with this problem, it is necessary to properly extend the etching time in multiscale simulation. The result of a 1.6s etching multiscale simulation and the result of 1.8s etching multiscale simulation, also shown in Figure 6.32, which shows that extend etching time to 1.6s can mitigate the phenomenon and 1.8s to completely solve the problem by reach saturation. Please note that the 0.8s extension is specific to the reaction and reactor applied in this chapter; this extension time is highly dependent on the reaction, reactor geometry, and the flow conditions.

The etching-time delay observed between the microscopic and multiscale simulations does not arise from differences in the underlying surface reaction mechanisms, which are identical in both models, but rather from the inclusion of precursor pressure delivery and transport dynamics in the multiscale framework. Finite pressure buildup and decay reduce the effective etching reaction rate, resulting in a small but systematic decrease in etching efficiency per batch. While this reduction has a small impact during the initial batches, the deposition process is inherently history dependent that any residual growth remaining on NGA persists and serves as an active surface for subsequent cycles. As deposition proceeds, these early-stage residuals are progressively amplified, leading to an accelerating accumulation of NGA thickness across batches. Consequently, even a minor difference in etching effectiveness during the early stages that is introduced by transport-limited precursor delivery can translate into a substantial divergence in selectivity after many batches. This

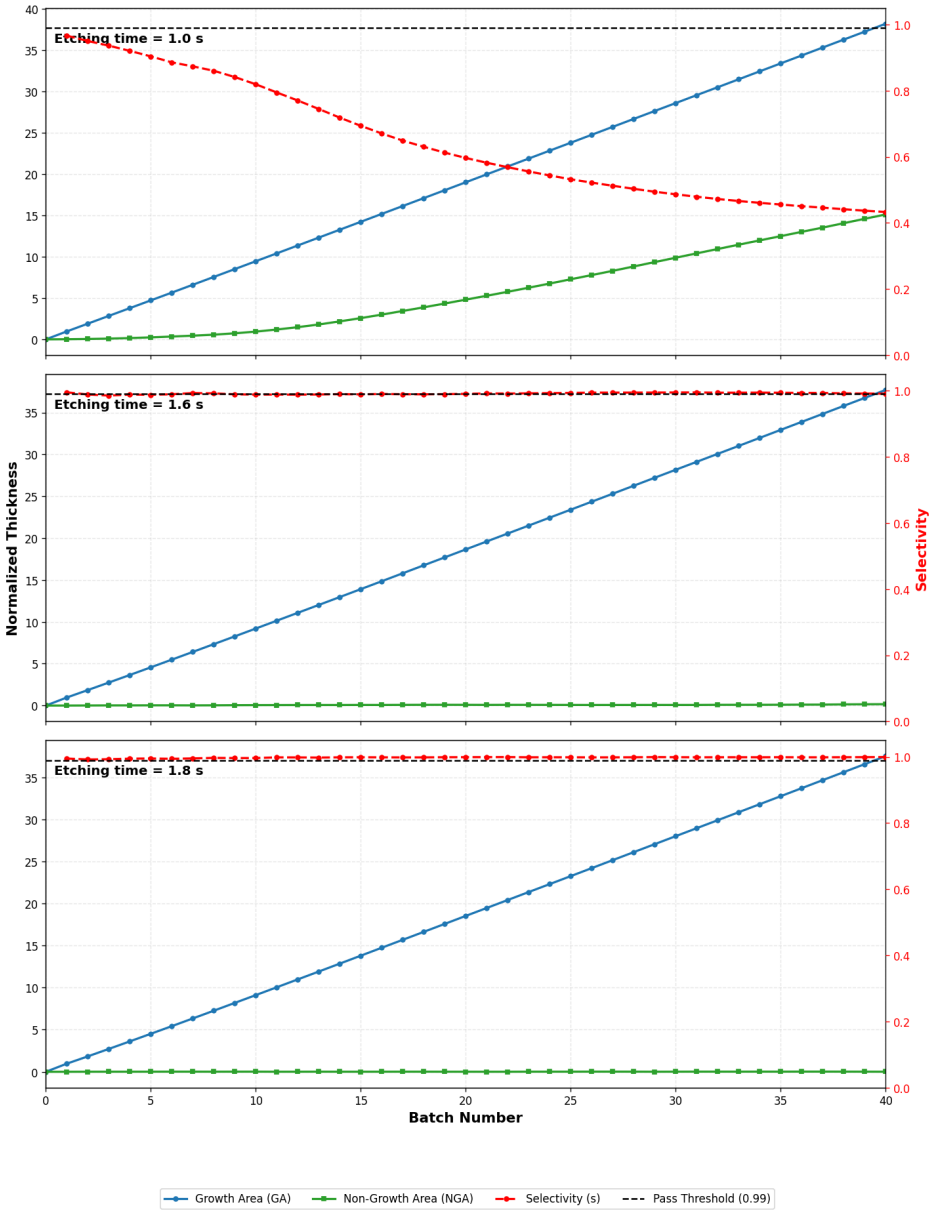


Figure 6.32: Selectivity evolution over 40 batches under different etching times predicted by multiscale simulation. At an etching time of 1.0 s, the microscopic-only model predicts near-perfect selectivity across all batches; however, incorporation of precursor transport and delivery effects in the full ALD reactor model leads to a pronounced degradation in selectivity to an unacceptable level. Increasing the etching time to 1.6 s marginally maintains selectivity above the pass criterion, but the response remains unsaturated and exhibits a gradual downward trend, suggesting limited process margin for thicker depositions. Further increasing the etching time to 1.8 s drives the system close to saturation in the multiscale simulation, indicating an effective 0.8 s delay relative to predictions from the purely microscopic model.

cumulative amplification explains why the multiscale simulation requires a longer etching time to achieve the same long-term selectivity predicted by the microscopic model.

The approximately 0.8 s delay observed between precursor injection and effective surface reaction onset in the multiscale simulations is primarily governed by reactor geometry and gas-distribution design rather than by surface reaction kinetics. In the NIST reactor employed in this study, precursor delivery follows an inlet-to-wafer pathway that includes an expansion cone structure designed to promote lateral diffusion and improve spatial uniformity of precursor partial pressure across the wafer surface. While this diffuser geometry enhances uniformity, it also increases the effective transport distance and mixing volume between the inlet and the wafer, leading to a finite pressure buildup time at the surface. As a result, the observed delay is dominated by geometric characteristics such as inlet–wafer spacing, diffuser volume, and flow expansion design. Although precursor flow rate and total injected volume influence the overall gas residence time, ALD and ALE processes are self-limiting and require only a small fraction of the delivered precursor to react; increasing flow rate to reduce delivery delay would therefore lead to inefficient precursor utilization and is generally undesirable in industrial practice. More compact or spatially efficient gas-distribution schemes, such as showerhead-type injectors, can reduce delivery delay while maintaining uniformity, as demonstrated in prior reactor design studies. Consequently, the magnitude of the observed delay is reactor-specific and should be interpreted as a geometry-dependent transport effect rather than a universal kinetic parameter.

6.6 Conclusion

this chapter addresses the critical challenge of alignment errors in advanced semiconductor fabrication by modeling area-selective atomic layer deposition (ASALD), a bottom-up, self-aligned method capable of eliminating edge placement error. To capture the imperfect nature of current ASALD chemistry, a microscopic Monte Carlo–based collision model was developed to simulate the adsorption of small-molecule inhibitors (SMIs), precursor nucleation, and surface oxidation reactions on SiO_2 and Al_2O_3 substrates. The model quantitatively reproduces known experimental behaviors, including ∼22% monodentate inhibitor coverage and ∼6.7% unwanted precursor nucleation on NGA, both closely matching reported values [103].

To recover selectivity in the presence of imperfect inhibition, the ASALD cycle was extended to include an exposure-dependent thermal atomic layer etching (ALE) step using TMA and HF. Parametric studies revealed that etch durations of 0.6 s and 0.8 s could preserve selectivity above 0.99 for 10 and 20 deposition cycles, respectively, while 1.0 s etching maintained perfect selectivity across 40 batches. Beyond this point, additional etching yielded diminishing returns due to increased material loss from the GA.

To simulate industrially relevant conditions, the microscopic model was coupled to a CFD-based reactor model developed and validated with NIST, forming a multiscale digital twin of the ASALD process. The multiscale framework revealed that delayed precursor delivery in the reactor leads to slower initial reaction rates, requiring adjusted etch durations. For example, a 1.0 s etch time in multiscale simulation has much worse performance and becomes unacceptable, whereas extending the etch to 1.6 s restored selectivity across all 40 cycles, while extending to 1.8 s reaches

saturation.

These findings demonstrate that integrating surface-level reaction dynamics with reactor-scale transport effects is essential for accurate prediction and optimization of ASALD processes. The modeling framework presented here can be readily adapted to other inhibitor chemistries, precursor systems, and reactor configurations, and serves as a foundation for future work in feedback control, surface defect engineering, and AI-guided reactor optimization.

Chapter 7

Industrial Data-Driven Machine Learning Soft Sensing for Optimal Operation of Etching Tools

7.1 Introduction

In the current decade, there has been a world-wide surge in demand for electronic products. This has in turn dramatically increased the manufacturing demand for related commodities such as micro-electronics, hard drives, and integrated circuits [136]. The innovations of modern-day electronics and their rising demand is in part owed to the rising density of transistors in semiconductor chips, which improves the computing performance of these chips [137]. This increased demand has resulted in recurring shortages of electronics and has hurt the global economy, which now depends on the manufacturing of electronic devices. Thus, there is a growing need to pursue innovation in the manufacturing sectors [138]. Smart manufacturing and Industry 4.0 concepts were proposed in the

mid-2000s for developing smart plants and factories that utilize network communications, Information Technology (IT), Internet of Things (IoT) and big data [139, 140]. Industry 4.0 aims to achieve resilient manufacturing processes that are characterized by high efficiency, high conformance, and high fidelity.

Smart Manufacturing, or Industry 4.0, necessitates holistic and continuous sensing, monitoring, and automation of processes that produce data in the form of quantifiable parameters. For instance, numerous advanced sensors are employed in manufacturing processes to monitor and understand the process parameters, and at the same time, they provide measured feedback to controllers for process automation. These sensors directly measure physical properties to obtain process information for monitoring and product quality assessment. For example, a quartz crystal microbalance is commonly used in the semiconductor industry to measure film thickness, providing coverage data for etching and deposition processes [141]. However, the operation of the measurement equipment is labor-intensive, time-intensive, and costly. In some scenarios, the capital and resource expenditures outweigh the value of the final product, one example of which is the treatment of wastewater [142]. However, the costs of these measurement steps can be mitigated through advanced process monitoring [143]. As industrial processes become increasingly complex, the direct measurement of key process parameters, which are often key performance indicators (KPIs), becomes more challenging. One process monitoring method is soft sensing, which detects critical process parameters by leveraging the wide range and scope of operational data that is generated in modern manufacturing processes [144]. Soft sensors use data from existing physical sensors and prior knowledge to develop data-driven algorithms that predict specific physical quantities and product quality, offering a more efficient, high-measuring frequency, and less labor-intensive alter-

native to traditional sensing approaches. Thus, they are particularly effective when there is a need to capture complex physical phenomena that are not easily measured or modeled or when there is a need for embedded fidelity that comes from years of operational experience. However, even when vast amounts of data are generated, the sensor performance will suffer if that data does not adequately cover the range, scope, and function of the operation relative to the sensor objectives.

In the context of modern complex manufacturing processes and the vast amounts of data they generate, there is a growing body of research that applies machine learning methods to develop soft sensors for detecting process and product properties. Recent literature reviews on deep learning methods in soft sensing [145] emphasize the significance of neural network approaches, including Convolutional Neural Networks (CNN) [146], Recurrent Neural Networks (RNN) [147], and a combination of RNN and Feedforward Neural Networks [148]. Seagate Technology also reported using the novel transformer network to predict the PASS/FAIL of an industrial etching process with time series data as the input [149]. Deep learning with neural networks have advantages in capturing complex nonlinear correlations between input and output parameters, and when large amounts of training data are available, they often outperform traditional machine learning methods [150].

While many works have investigated which models are best suited for which deep-learning tasks, the question of if and how data aggregation can be used to supplement modeling tasks with small data volumes remains unanswered. Thus, this work proposes a deep-learning-based soft sensor developed using industrial data for detecting both binary properties (PASS/FAIL) and numerical properties (oxide thickness) for several industrial etching tools from Seagate Technology with high-dimensional input parameters. To address the problem where there is not enough data to train a model with high performance on a single tool, this chapter proposes a novel data aggre-

gation method where datasets from other tools are combined with the dataset of a single tool to improve model performance on that specific tool. The data aggregation approach aims to improve the performance of the trained soft sensor model by properly (in a sense to be made clear below) combining datasets to increase the amount and variety of training data. Specifically, this chapter introduces a statistical method to optimize dataset selection during the aggregation process to improve aggregation efficiency.

this chapter is organized as follows: Section 7.2.1 provides an overview of the industry data used, Section 8.2 describes the preprocessing operations applied to the datasets, Section 7.2.3 and Section 7.2.4 describe the development of the soft sensor models, Section 7.3 demonstrates and evaluates the performance of the trained soft sensor models, and Section 8.4 summarizes the findings of this chapter.

7.2 Data Processing and Modeling

This section describes the collection, processing, and contextualization of data from five industrial plasma etching tools, which are used to train two models: a classification model and a regression model. Then, we cover a cross-process data aggregation procedure for improving the classification model and the various loss functions used in training the regression model.

In the semiconductor fabrication industry, all products begin as a raw silicon wafer substrate. These substrates follow a set of procedures called the process flow, which describes each process step that the wafer must undergo. Once the wafer has gone through the entire process flow, it is a completed product. As the fabrication process is very repetitive, each wafer will be processed

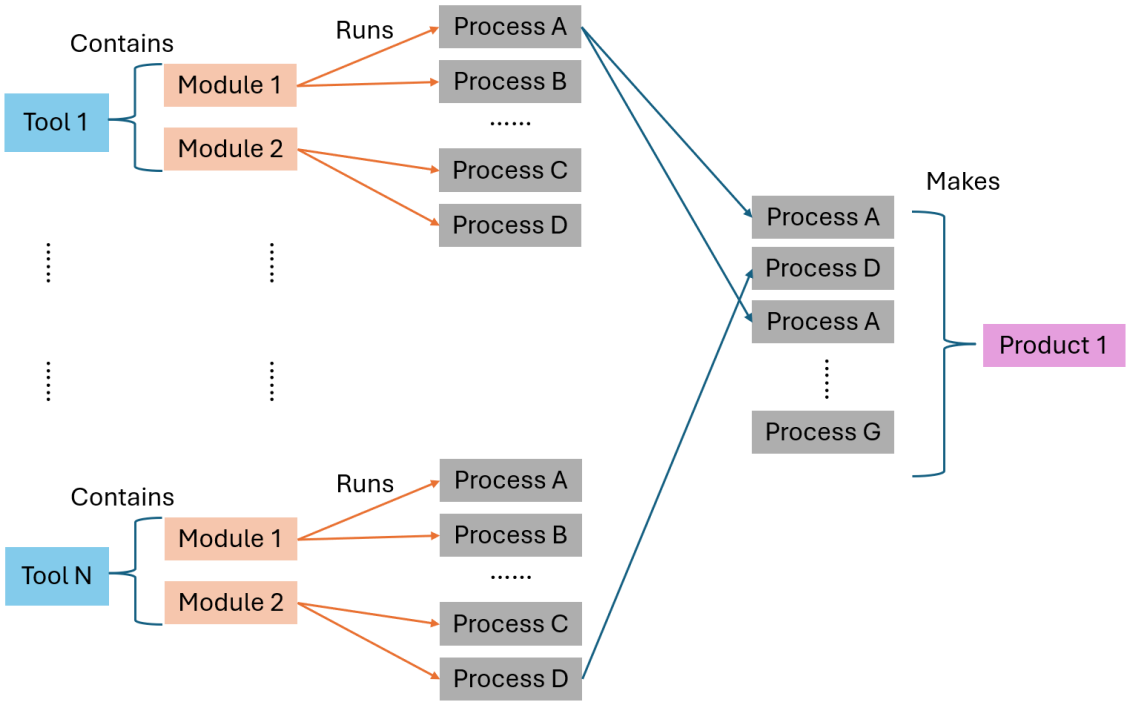


Figure 7.1: The overall manufacturing system for an industrial etching equipment. Each tool is an etching reactor that has multiple modules and can run various processes. Wafers start as pure silicon substrates, and after a series of production processes, they become a finished product.

on the same tool multiple times at different process steps. This chapter examines a toolset of five electrically-induced plasma etching tools. This toolset consists of five physically identical chemical etching reactors that possess up to two chambers; the reactor is referred to as the "tool," and the chamber is referred to as a "module." Each module can run a variety of process steps. The process data is gathered on a per-module basis, which means that each datum is for a specific tool-module combination. For example, an entry from T1-PM1 means that the wafer was processed in module PM1 of tool T1. A detailed diagram explaining the overall manufacturing process is shown in Figure 7.1.

7.2.1 Industrial Data Generation

The process data used in this chapter was collected from four tools: T2, T4, T5, and T7. Each tool has up to two modules: PM1 and PM2. Specifically, process data was collected from T7-PM1, T7-PM2, T2-PM2, T5-PM2, and T4-PM1. Each data entry comes from a single run, which is defined as a process step that starts at $t = 0$ and ends at a preset process time, t_{end}. During the process, physical sensors track 33 numerical features, and their average is recorded down alongside two process-specific discrete features and a time stamp. Thus, there is no time-series data. Once the wafer has finished processing, it is referred to as the product. The dataset used in this chapter spans from February 1st, 2018 to December 31st, 2022. The numerical features are categorized into three classes of variables which are pressure and gas flow, electromagnetic, and other equipment statuses and properties. The discrete features are the process name ID and substrate family ID, which are both alphanumeric text entries. These 35 pieces of information are inputs for both the classification and regression model.

The information that both models aim to predict is whether the process was successfully completed, and this information is gathered at a different tool. After the etch step, the product wafer is processed at a metrology tool that measures how much substrate was etched away. As all of the processes examined in this chapter are oxide etches, the metrology tool will measure the remaining oxide thickness. Depending on how much the measured oxide thickness deviates from the target oxide thickness, the run will be labeled as either a "PASS" or a "FAIL." The classification model uses the binary PASS/FAIL measurement as its output, and the regression model uses the target oxide thickness as an additional input and the measured oxide thickness as its output.

7.2.2 Data Preprocessing

Data preprocessing is a crucial step to effectively train any model. A properly preprocessed dataset allows the model to effectively learn the patterns of the data. For instance, [151] highlights the necessity and importance of data preprocessing in several deep learning-based applications. Preprocessing steps often include removing or filling in invalid and abnormal data as is appropriate, encoding discrete variables, and normalizing features to avoid skewing caused by the absolute value of variables. These steps ensure that the model receives consistent and relevant data, which facilitate better learning and prediction ability; specifically for interpolating unseen data points within the applied training range. The input data preprocessing procedures in this chapter are demonstrated in Figure 7.2, and the procedures for output data preprocessing are shown in Figure 7.3.

From a practical manufacturing standpoint, physical sensors do not function properly at all times, and not all parameters can be measured in all processes. Thus, there are almost always missing physical measurements in real industrial data. The data analyzed in this chapter is not exempt from this phenomena. Some particular tool-module combinations are missing entire features, and other features are only collected within certain time ranges. To address these issues, any feature that has no measured value (which is recorded as N/A) is filled in with a numerical value of 0 to maintain consistency with the other data points. On the other hand, any runs (one row of data) that are missing either of the outputs, which are the PASS/FAIL criterion and the measured oxide thickness, are removed from the dataset. Without the true results, the input features are meaningless. By applying these two methods, all the invalid and abnormal data is pared from the datasets.

As with most machine learning models, which includes neural networks, all inputs must be

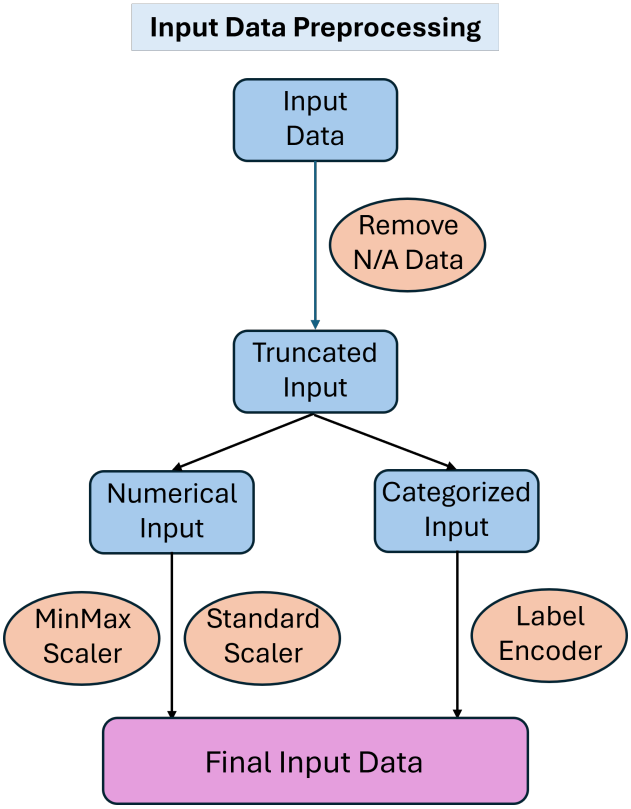


Figure 7.2: Input Data Preprocessing **Step 1**: Eliminate or pad missing data. **Step 2**: Scale the numerical features, and encode the discrete features. **Step 3**: Combine numerical and discrete features into one complete input dataset.

numerical. Thus, it is necessary to encode any categorical or nonnumerical features if they are included in the training process. For this chapter, the label encoder from the scikit-learn package [152] is used to encode both the substrate family ID and the process name ID from alphanumeric features into numerical features. To create a consistent and holistic encoder that is capable of handling all possible cases, the encoder is trained on data that comprise the concatenation of all datasets from all tool-modules. This step creates a complete map for the encoder, ensuring that each discrete feature is transformed into a unique number, which avoids conflicts during the training of the model. Additionally, for the binary PASS/FAIL output data, a 0/1 encoder is applied, which encodes PASS as 1 and FAIL as 0. These methods ensure that all the categorical features are transformed into

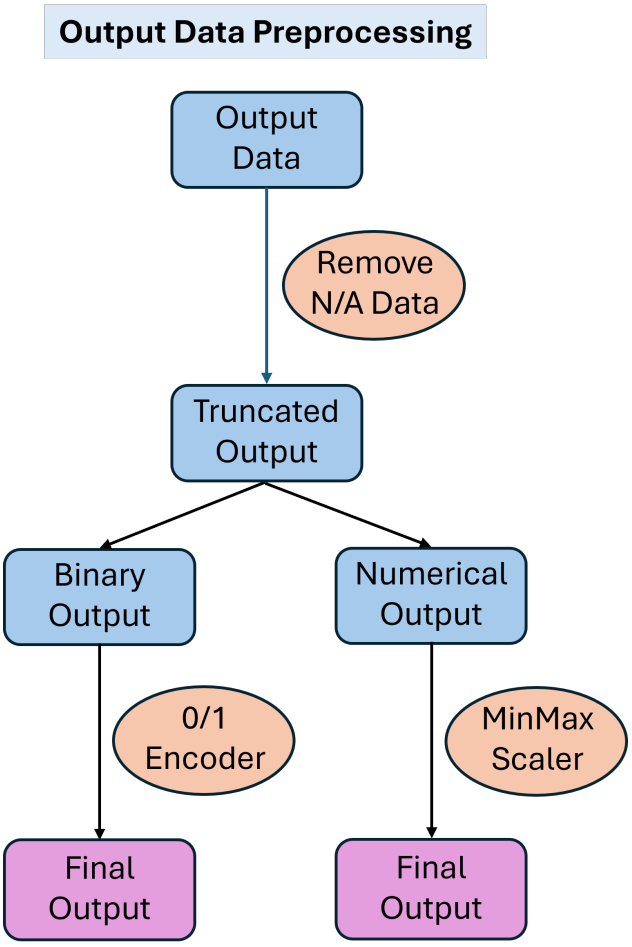


Figure 7.3: Output Data Preprocessing **Step 1**: Eliminate missing data. **Step 2**: Encode binary output to FAIL(0) and PASS(1). **Step 2-2**: Scale numerical output with MinMax Scaler.

numerical values so that the models can function effectively.

For neural networks, it is especially crucial to scale all numerical data to prevent the vanishing gradient and gradient explosion phenomena from occurring during the training process [153]. This is particularly important for input data features that vary significantly in absolute values. For instance, the dataset examined in this chapter has features in the range of both 10^{-4} and 10^{2}. A scaler normalizes the input features of the data, forcing each numerical feature to exist in a similar range and making the training process more stable and the optimizer task easier. For the classification

task with a binary PASS/FAIL output, the numerical input features are scaled with a standard scaler, and the output features are already encoded as 1/0. For the regression task with numerical output features, separate MinMax scalers are applied for both the input and output data. The standard scaler is described in Equation (8.1), and the MinMax scaler is described in Equation (7.2). After the encoding process described earlier and the scaling processes described here are completed, all the input features, including the scaled numerical features and encoded categorical features, are concatenated into a data vector that represents a single run.

$$Z = \frac{X - u}{s} \tag{7.1}$$

$$Z = \frac{X - Min(X)}{Max(X) - Min(X)} \tag{7.2}$$

where Z is the output vector of a scaled numerical feature, X is the input vector of the original feature, u is the average value of X, and s is the standard deviation of X. Note that the vectors used here consist of all the data for a particular feature (a data column). The standard scaler scales the original dataset around 0 for each feature in a similar range, while MinMax scaler scales the numerical data into a range of [0,1]. There is no fixed rule to determine the best scaling method. Rather, scaler selection is dependent on the optimal model performance, which will be more rigorously defined later on in Section 7.3. Specifically, both the standard scaler and the MinMax scaler are applied to the same dataset and used to train two independent models. Then, the scaler that yields the best performance is chosen for that task.

7.2.3 Classification Model

The goal of a classification model is to accurately determine whether future, unknown wafers from a specific tool-module combination will pass or fail the following metrology step. The preprocessed input data vectors are the input variables to the model, and the output variable is a binary PASS/FAIL value. The dataset is first separated into two datasets by time: the modeling set and the test set. The modeling set comprises all the runs from February 1st, 2018 to December 31st, 2021, and it will be used to train the model. The test set comprises all the runs from January 1st, 2022 to December 31st, 2022, and it is used to evaluate model performance. These datasets are separated by time because it is necessary for the model to be generalizable across all times. From a practical point of view, the soft sensor model can only be considered successful if it can be effectively applied on unseen data and conditions from future manufacturing processes. To prevent overfitting, the modeling set is further separated into two more sets: the training set and the validation set. 80% of the modeling dataset is randomly chosen for the training set, and the remaining 20% is allocated to the validation set. The model is only trained and optimized on the training set, and the performance of each model is examined on both the training and validation sets to tune and optimize the generalization ability of the model. Stratified sampling is also used to ensure that the PASS/FAIL distribution is nearly identical between the training and validation datasets. With these specifications, when the model is trained on the training set and tested on the validation set, the candidate model is the model with the best performance on the validation set.

Model Training

The Feedforward Neural Network (FNN) is applied to the classification model in this chapter. The general structure of the network is shown in Figure 7.4.

In an FNN, each neuron in the hidden layers takes a weighted sum of inputs from the input layer or previous hidden layers, applies a non-linear activation function, and then passes the result to the next layer. The final output layer produces the binary PASS/FAIL classification by using the sigmoid function to transfer the input value into the [0,1] range. The sigmoid function is also employed as the activation function in the hidden layers of classification models, due to its superior performance in classifier models [149]. The sigmoid function is described below in Equation (7.3):

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{7.3}$$

The FNN has several tunable hyperparameters that need to be optimized to determine the best network structure. These hyperparameters include the number of hidden layers, the number of neurons in each hidden layer, the learning rate, and the L2 regularization coefficient. A grid search method is applied in this chapter to find the proper combination of hyperparameters to optimize the model performance on the validation set. The complete list of tuned hyperparameters and their candidate values are shown in Table 7.1. The models are trained on powerful graphical processing units (GPU), such as the Nvidia RTX A4000, Nvidia RTX 3060, and Nvidia RTX 4090, to facilitate the complete grid search of hyperparameters by exploiting the high computational capabilities of these GPUs. The selected hyperparameters are bolded in Table 7.1.

The training goal of a neural network is to minimize the loss function. For most binary clas-

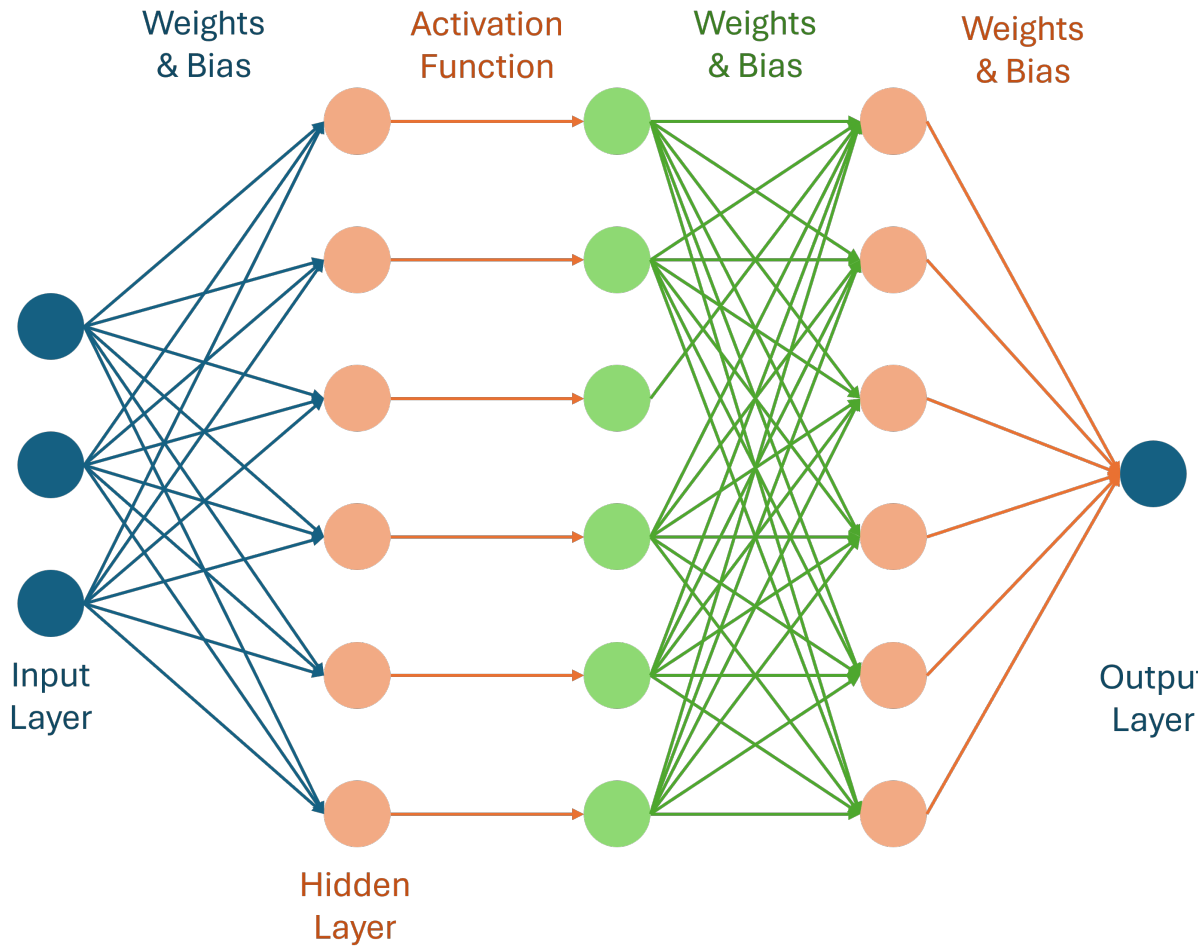


Figure 7.4: The general structure of the Feedforward Neural Network (FNN) is shown here. The hidden layer neurons take a weighted sum from the input layer or the previous hidden layer, which is then activated by nonlinear functions. The output layer takes the weighed sum from the last hidden layer to output the result.

Table 7.1: Classification FNN Hyperparameters and Tuning Range.

Hyperparameters	Candidate Values
Number of Layers	[1,**2**,3]
Number of Neurons	[32,**64**,128]
Learning Rate	[0.005,**0.001**,0.0005,0.0001]
Dropout Rate	[**0**,0.1,0.5]
L2 Regularizer	[**0**,0.0001,0.0005,0.001]

sification tasks whose output values are processed by a sigmoid function, a cross-entropy loss is typically used, as shown in Equation (7.4):

$$J = -\frac{1}{N}\sum_{i}[y_i \ln(y_i^p) + (1 - y_i) \ln(1 - y_i^p)] \tag{7.4}$$

where J is the loss value, y_i is the true value (0 or 1) of the data point i, y_i^p is the predicted value from the model within range [0,1], and N is the number of data points. However, it is important to note that the data is imbalanced because it comes from an industrial toolset that generally runs well with a low but still significant fail rate. All the fail rates for all five datasets are shown in Table 7.2, and most datasets exhibit a fail rate of approximately 2.5%. T5-PM2 is a notable exception with a significantly higher fail rate. This can be partially explained by its small dataset volume, which causes a few FAIL data points to have a large impact on the fail rate, but TM5-PM2's data clearly has more fails than the other tool-module combinations, marking it as different. Thus, by aggregating the TM5-PM2 dataset with other datasets and observing whether model performance increases or decreases, the effects of aggregating less related datasets together can be seen. For all other datasets, a model that always predicts "PASS" will not result in a high loss with the normal cross-entropy

loss function, and that behavior will be favored during the training process.

Table 7.2: Overall Size and Distribution of Datasets.

Dataset	Total Data Points	PASS Data Points	FAIL Data Points	FAIL Rate
T4-PM1	39717	38836	881	2.22%
T7-PM1	23387	23057	330	1.41%
T2-PM2	36335	35461	874	2.41%
T5-PM2	771	667	104	13.49%
T7-PM2	2722	2650	72	2.65%

However, such a model is not useful because it will have a 100% false positive rate. In other words, the model will not be able to detect any misprocessed wafers even though it has a very low training loss. To address this issue, a weighted cross entropy algorithm is proposed by multiplying the weight to data points by the output distribution [149]. The weighted loss function has the form:

$$J = -\frac{1}{N}\sum_{i} w_i[y_i \ln(y_i^p) + (1 - y_i) \ln(1 - y_i^p)] \tag{7.5}$$

where w_i is the weight and N is the total number of data points. The weights for binary classification are calculated as follows:

$$w_i = \begin{cases} \frac{N}{n_0}, & \text{if } y_i = 0 \\ \frac{N}{n_1}, & \text{if } y_i = 1 \end{cases} \tag{7.6}$$

where w_i is the i^{th} weight, n_0 is the number of FAIL data points, and n_1 is the number of PASS data points. The weights applied in the loss function emphasizes minority instances to favor models with more comprehensive and balanced performances, thereby mitigating the potential bias that can arise from imbalanced datasets.

The Adam optimizer is applied throughout the entire model training process due to its excellent performance in handling various deep learning tasks [154]. Adam combines the advantages of other optimizers such as AdaGrad and RMSProp to enhance convergence speed and model performance. Model training is conducted over 1000 epochs, with the validation set loss continuously monitored after each epoch. The model with the lowest validation loss up to that epoch is then saved as the best model until the end of the training process. This approach ensures that the model with best generalization ability is retained, effectively reducing the risk of overfitting and improving the model's generalization capabilities. After the training process, the model is tested on future data of the test set to evaluate its performance.

To further reduce variability in the training and testing process, a cross-validation method is applied. This approach can greatly reduce model variance and improve the reliability of the model performance by averaging the performance of multiple models that are trained on different sections of the original data [155]. First, the modeling dataset is randomly divided into five segments. Then, a segment is chosen as the validation set with the other four segments becoming the training set. By repeating this five times in total, once for each segment, five models are trained. Finally, each model is individually run on the test dataset, and the average of the five scores from the five models is used as the final test score for that dataset. This method enhances the robustness of the model evaluation by ensuring that the performance is consistent across different subsets of the data.

Data Aggregation

Deep learning models, with their complex architectures and numerous parameters, can effectively capture intricate patterns within large datasets, leading to superior performance in most tasks [150].

As a result, deep learning networks have a distinct advantage over traditional machine learning methods, especially when working with training data at the industrial scale. However, industrial datasets often vary significantly in size between different tool-module combinations. For example, in this chapter, as shown in Table 7.2, T7-PM2 and T5-PM2 have dramatically smaller datasets with less than 3000 points compared to T4-PM1, T7-PM1, and T2-PM2, which have more than 20000 points. This variation in dataset size implies that models trained using limited data from a particular tool-module combination for that specific combination will have considerably worse training results compared to tool-module combinations with larger datasets. Additionally, when the validation dataset is limited in size, it can lead to bias in the model, decreasing its ability to generalize because the validation set might no longer accurately represent the general data distribution. Moreover, even for the three tool-module combinations with large datasets, dataset aggregation will increase the variational and operational coverage of the training data and generally improve model performance. Consequently, maximizing high-quality training data volume is necessary to improve model performance. However, it is not enough to simply merge a large dataset and a small dataset, as that would merely result in a model that predicts the average behavior of the two aggregated datasets. Rather, multiple datasets from various tool-module combinations should be aggregated into a superset of varied data. When trained on this superset, the model performance improves as the larger volume of data better represents the overall process, increases the operational scope, and has more comprehensive information regarding process failures. A large amount of varied training data points that can only be obtained from aggregating data from multiple tool-module combinations refines the model, reduces overfitting, and enhances the robustness of the predictions.

To provide more data to train a model for each tool-module dataset, a data aggregation method

is developed and tested that combines multiple datasets, significantly increasing the amount and variety of training data and improving model performance. During this process, the candidate datasets for aggregation must be selected carefully, making the analysis and selection process a critical data processing step. If the chosen tool-module dataset is very different from the current dataset, then the model will likely be misdirected by the new data and fail to retain the distribution of the original dataset. One method to guarantee optimal data aggregation is to exhaustively test all possible dataset combinations, but that is only feasible when there are only a few datasets. The number of possible dataset combinations increases exponentially with the number of datasets; n datasets have $2^n - 1$ unique combinations, making it impractical to exhaustively test all possible combinations when there are many tools and datasets.

To address this issue, this chapter develops an indexing method to evaluate the similarities and differences between datasets. It was then used to select candidate datasets for aggregation, ones that are most similar to the dataset of interest. The indexing method is a point-biserial correlation analysis, which is a statistical method that measures the relationship between a continuous variable and a binary variable. In this chapter, the point-biserial correlation analysis is conducted on each numerical feature in the dataset with respect to the output binary variables, which provides information about the contribution of each feature to the binary outcome (PASS/FAIL). The analysis involves calculating the correlation coefficient as shown in Equation (7.7).

$$r_{bis} = \frac{\bar{Y}_1 - \bar{Y}_0}{s_y} \sqrt{\frac{N_0 N_1}{N(N-1)}} \tag{7.7}$$

where r_{bis} is the correlation score between a specific feature and the output, $\bar{Y}_1$ is the average value

of the feature for all PASS data points, $\bar{Y}_0$ is the average value of the feature for all FAIL data points, s_y is the standard deviation of the feature for all data points, N_0 is the number of FAIL data points, N_1 is the number of PASS data points, and N is the total number of data points. After the correlation coefficients are calculated for each feature, they are assembled into a characteristic vector. Each dataset has its own characteristic vector, and the difference score between any two datasets is calculated by finding the mean absolute error (MAE) between their characteristic vectors as shown in eq. (7.8).

$$d = AVG(|\vec{c_1} - \vec{c_2}|) \tag{7.8}$$

where d is the difference score between the two datasets, AVG is an operation that takes the average value of all elements in a vector, $\vec{c_1}$ is the characteristic vector of first dataset, and $\vec{c_2}$ is the characteristic vector of second dataset. After the difference scores are calculated between the current analyzed dataset and all other datasets, the one with the smallest difference score is chosen as the candidate dataset for aggregation. The statistical analysis of the datasets themselves does not require any model training until the candidate datasets for aggregation are selected. This approach significantly reduces the number of models that need to be trained and tested, saving a substantial amount of time and computational resources.

7.2.4 Regression Model

this chapter also explores machine-learning models that quantitatively predict the measured oxide thickness of processed wafers. This model, also called a regression model, is another FNN, though one with a different structure better suited for non-binary outputs. The preprocessed input data for

the regression models are the same as that of the classification models, but with the addition of the target oxide thickness as an input feature. This feature is included as the target oxide thickness for each process must be known before the run starts. The model output is the measured oxide thickness, which spans a wide numerical range, from 0 to over 5000 Å, depending on the process and product.

Regression models generally require more data than classification models as the output of the latter is more complex. Thus, this chapter only focuses on training regression models with large datasets [150]. Specifically, T2-PM2, T4-PM1, and T7-PM1 are used to train regression models while T5-PM2 and T7-PM2 are not. The latter two datasets cannot support the training of a complex regression model because the limited data volume can induce problems such as severe overfitting and high variance results. Due to a lack of feasible datasets, data aggregation, which was explored for the classification task, is not conducted for the regression task because there are only three applicable datasets for regression. The available multi-dataset combinations are limited; specifically there are three two-set aggregations and one three-set aggregation. This small sample size means that any conclusions reached through the statistical analysis may be biased. Lastly, regression models do not require any data stratification like classification models because random selection and chronological selection can easily create artificial differences in fail rates between the training and validation sets due to the natural imbalance of the datasets. Because only datasets with substantial data volumes are selected, which effectively reduces the variance in the input features between different time segments, the training-validation split is organized solely by time. Specifically, the first 80% of the total training-validation dataset, sorted chronologically, is used for training, and the remaining 20% is reserved for validation. This approach aims to enhance model performance

by selecting for the regression model that best predicts future data points.

Regression Model Training

The Feedforward Neural Network (FNN) for the oxide thickness regression task applies the ReLU (Rectified Linear Unit) activation function to provide nonlinearity for the model, which is described by the following equation:

$$\mathrm{ReLU}(x) = \max(0, x) \tag{7.9}$$

The ReLU function is chosen because of its promising performance on various training tasks, ability to avoid gradient vanishing problem, and increase in the speed of the training process [156, 157]. In addition, due to how complex it is to train regression models and the need to save computational resources during the hyperparameter grid search, the neural network is designed such that each subsequent hidden layer has half the neurons of the previous layer. This approach helps manage model complexity, reduces the risk of overfitting, and ensures efficient feature extraction. Similar to the classification task case, the tunable hyperparameters and their range of interests are shown in Table 7.3, and the selected hyperparameters are bolded.

Table 7.3: Regression FNN Hyperparameters and Tuning Range.

Hyperparameters	Candidate Values
Number of Layers	[2,**3**]
First hidden layer Neurons	[32,**64**,128]
Learning Rate	[0.0001,**1E-5**,5E-6,2E-6]
Dropout Rate	[**0**,0.1,0.25]
L2 Regularizer	[**0**,0.0001,0.0005,0.001]

A small learning rate and many training epochs (100,000 in this chapter) are essential for training regression models due to the wide range of output values and the highly complex nonlinear correlations between the 37 input/output features. These conditions ensure that the training process reaches an optimum. Without the small learning rate, the model is sensitive to improper convergence, which causes the training process to jump around the optima or even diverge. By contrast, a small learning rate allows for more precise adjustments to the weights during backpropagation. The extensive number of training epochs ensures that the model has sufficient iterations to learn these complex relationships thoroughly, ultimately leading to better generalization and performance on future data.

Because the output has such a wide range of numerical values, two kinds of loss functions are tested in this chapter: the mean squared error (MSE) and the mean absolute percentage error (MAPE). The MSE loss function is given below:

$$J = \frac{1}{N}\sum_{i}(y_i^p - y_i)^2 \tag{7.10}$$

And the MAPE is described as follows:

$$J = \frac{1}{N}\sum_{i}\frac{|y_i^p - y_i|}{y_i + 1} \tag{7.11}$$

where J is the loss function value, N is the total number of data points, y_i^p is the predicted output value by the regression model, and y_i is the true output value. The plus one term in the MAPE equation prevents the equation from dividing by zero and also avoids ridiculously high loss values when the true output is close to zero. The loss functions are applied to the normalized data scaled by

the MinMax Scaler, which keeps small data values that are close to 0 still close to 0 and scales large data values close to 1. As the original data' s minimum value is retained, the data normalized by the MinMax scaler also retains the properties of the original dataset, which improves the ultimate performance of the model. When applied, these two loss functions intrinsically favor very different modeling patterns. MSE measures the average of the squares of the errors, treating all errors equally regardless of the magnitude of the true values. This results in data points with low true values having high percentage errors, as the model may focus more on minimizing absolute errors, which can disproportionately affect smaller values. Conversely, MSE can result in lower percentage errors for data points with high true values, as the absolute errors tend to be relatively smaller in proportion to the larger true values. This can be advantageous when precision for higher value predictions is critical. MAPE, on the other hand, measures the average absolute percentage error, ensuring that the model minimizes the percentage error across all data points. This results in a more uniform percentage error distribution, which is beneficial when the relative accuracy of predictions is more important. Each loss function has its benefits and drawbacks, which is why this chapter uses both to train the model. Using both MSE and MAPE allows for a holistic review of the data and the modeling process. This dual approach provides a more comprehensive understanding of the model's potential across the entire range of output values.

7.3 Results and Analysis

7.3.1 Classification Model Performance

The performance of the classification models proposed in section 7.2.3 are ideally evaluated within the context of a manufacturing environment. With the classifier model, there are four possible process outcomes: a pass is classified as a pass (true positive), a pass is classified as a fail (false negative), a fail is classified as a fail (true negative), and a fail is classified as a pass (false positive). Of these outcomes, the true positive and true negative outcomes are trivially good outcomes, as the classifier model is correct. The false negative, while not ideal, can be mitigated by manufacturing procedures. If all runs classified as fails are reevaluated at the metrology machine and manually measured to determine whether they truly failed, then the false negatives will be caught and correctly reclassified as passes. Thus, the main manufacturing concern is false positives, as there is no easy way to identify them; manually measuring all passes in addition to all fails would make the classifier model superfluous.

Generally speaking, most false positives will be eventually caught at future metrology steps or at the final metrology and reliability testing of the end product, which means that they will ultimately have little effect on product quality [158]. The main impact of misidentifying a misprocessed wafer is that, as the misprocessed wafer moves through the process flow, it wastes resources and time. Thus, a high-performing classification model will have a low false positive rate as that minimizes wasted manufacturing resources, and any metrics used to analyze these models must be an indication of their false positive rates.

However, each model does not have a singular, representative false positive rate. Specifically, each model can be tuned to be more aggressive in flagging misprocessed wafers, lowering the false positive rate and increasing the false negative rate, or more conservative, which does the opposite. While confusion matrices are often used to evaluate classifier model performances, they are insufficient to evaluate the overall model performance, as a confusion matrix reflects the results of a single tuning approach. It cannot capture the model' s performance across all possible tuning configurations. Other traditional criteria for evaluating classification model performance, such as overall accuracy, are also unsuitable because the binary outputs of all the datasets are highly imbalanced. As previously mentioned, a model that only predicts PASS may have a high accuracy but it will have a 100% false positive rate, rendering it meaningless in actual industrial applications. The Receiver Operating Characteristic (ROC) analysis offers a more robust evaluation method by examining the true positive rate (TPR) and false positive rate (FPR) of the model's predictions on test data at different thresholds. The output from the sigmoid function in the model's last layer is a continuous value in the range of [0,1]. Although a fixed threshold of 0.5 is traditionally used that classifies values higher than 0.5 as pass and those lower as fail, this threshold can be set to any value within the [0,1] range. With different thresholds, different TPR and FPR values are produced. For instance, setting the threshold to 0 (all pass) results in a TPR of 100% and an FPR of 100%. Conversely, setting the threshold to 1 (all fail) yields a TPR of 0% and an FPR of 0%. Thus, selecting an appropriate threshold involves balancing model sensitivity (TPR) and false acceptance rates (FPR).

In this context, the model's performance is evaluated using the Area Under the Curve (AUC) score. The AUC score is defined as the area under the ROC curve, which plots TPR on the y-axis and FPR on the x-axis across different thresholds. Ideally, a perfect model would achieve a TPR

of 100% (max sensitivity) and an FPR of 0% (zero false alarms), resulting in an AUC score of 1. Conversely, a model that makes random guesses would have a TPR of 50% and an FPR of 50%, resulting in an AUC score of 0.5. The ROC-AUC score provides a comprehensive measure of model performance across all possible thresholds, making it particularly useful and widely applied for evaluating models on imbalanced datasets [159].

Single Dataset Model Performances

The model is first trained on single tool-module datasets to evaluate if the training method is effective at making predictions that are significantly better than random guessing, which would have an AUC score of 0.5. For each dataset, five models are created via the cross-validation training process as described in section 7.2.3. The average AUC score is defined as the mean value of the test scores for each trained model. Note that all ROC graphs are of the model whose performance best matches that of the average of the five models; this is the representative model. The scores for each tool-module combination are shown in Table 7.4. The ROC-AUC plot of the representative model for all five datasets is shown in Figure 7.5 The single dataset training results in Figure 7.5 demonstrate

Table 7.4: Single Dataset Training AUC Score.

Dataset Name	Average AUC Score
T4-PM1	0.73
T7-PM1	0.73
T2-PM2	0.80
T5-PM2	0.48
T7-PM2	0.52

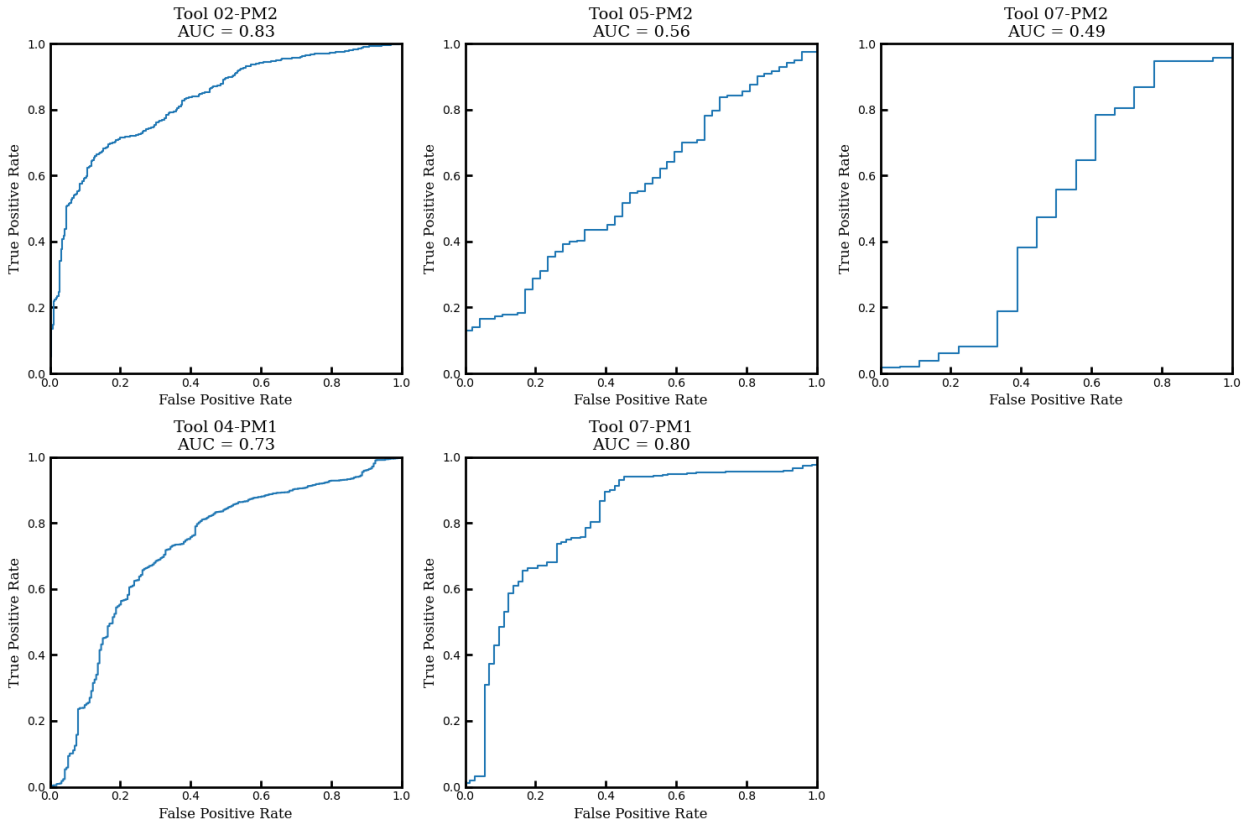


Figure 7.5: ROC plot for all five datasets. The performance of the models trained on datasets T5-PM2 and T7-PM2 is similar to random guesses, while the models based on the other three datasets have superior performances.

that the amount of training data is a key factor in model performance. The two smaller datasets, T5-PM2 and T7-PM2, show unacceptable performance close to near-random guesses (AUC scores of around 0.5). In contrast, the three larger datasets have AUC scores significantly above 0.5, indicating successful model training and effective classification of the PASS/FAIL status of the product. For these three tool-module combinations, the models can achieve a FPR rate of about 35% to 40% when the TPR is around 80%. This means that, if 10000 wafers are processed with 200 fails and 9800 passes (2% failrate), then 1960 wafers would be false negatives (20% of 9800 passes) and 70 wafers would be false positives (35% of 200 fails). This means that the overall rate of missed fails is less than 1% of the overall product throughput, which is considered high-performing. The single dataset cases suggest that data volume is crucial to model performance; with enough data, the models are able to effectively classify runs between PASS/FAIL. And with even more data, the AUC score can be further improved and the FPR reduced.

Multi Dataset Model Performances

To validate the efficacy of data aggregation in enhancing model performance, the process is first investigated within each module (PM1, PM2). Specifically, this involves forming each unique superset between T4-PM1 and T7-PM1 for PM1 and each superset between T2-PM2, T5-PM2, and T7-PM2 for PM2. This results in three two-set supersets for PM2 and one two-set superset for PM1. PM2 also has the option to aggregate all three datasets. The optimal AUC score for a given tool-module combination is defined as the best AUC score among all the possible supersets that contain that tool-module combination, and the optimal AUC score for each tool-module combination is illustrated in Figure 7.6. This analysis aims to determine whether combining datasets from

different tools within the same module can lead to improved predictive performance, as measured by the AUC score.

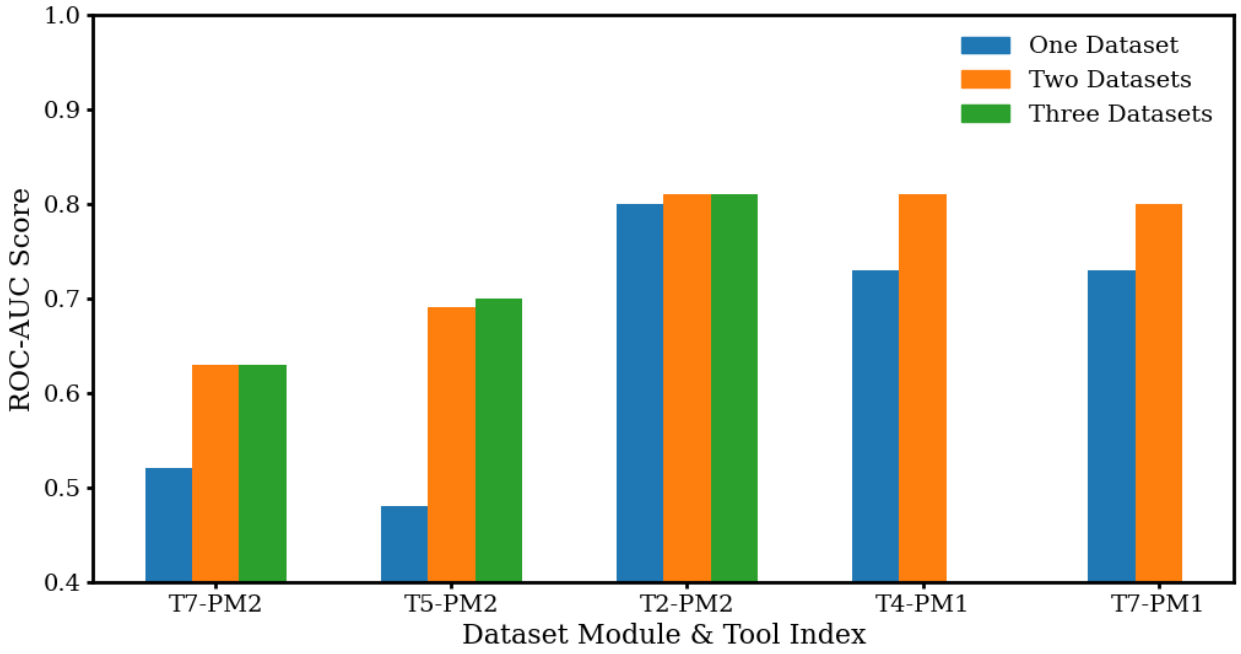


Figure 7.6: Best possible AUC score for all five tool-module combinations. Model performances improve substantially as data increases for all five cases.

From the same-module data aggregation results shown in Figure 7.6, the AUC score for the T7-PM2 model trained on only the T7-PM2 dataset is just above 0.5, which is an unacceptable performance. However, for the best-case two- or three-set supersets, the AUC score improves past 0.6. The results for T5-PM2 are even better. In the case of T2-PM2, it is clear that the base T2-PM2 dataset contributes the most to the model and that the model receives little benefit when it is aggregated with the other smaller datasets. However, the contribution that the T2-PM2 dataset makes in the orange and green bars of the T7-PM2 and T5-PM2 models demonstrates substantial performance improvement for Tools 05 and 07, which have smaller datasets. Generally, data aggregation substantially improves model performance for tool-module combinations with smaller

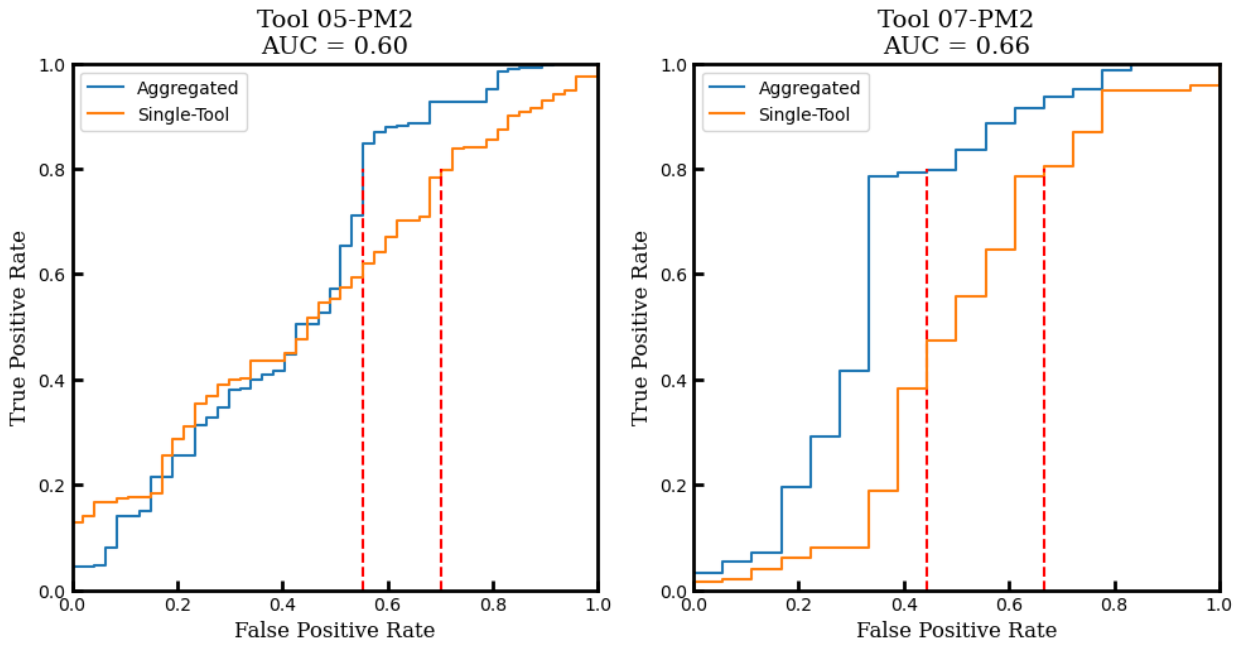


Figure 7.7: ROC plots of the representative models of T5-PM2 and T7-PM1, which were trained by aggregating all available datasets in the module. The performances are noticeably better than that of random guessing and single-set models. The FPR values for an 80% TPR value have also significantly improved.

datasets, such as T5-PM2. Furthermore, the three-set aggregation in PM2 allows the models to achieve a FPR of around 50% with a TPR of about 80%, as shown in the ROC plot in Figure 7.7. For the base model trained on only one dataset, an 80% TPR would correspond to an 80% FPR, which would result in an overall missed fail rate of 1.6% given a 2% fail rate. However, the model trained with multi-dataset aggregation would have a missed fail rate of 1%, a significant improvement. Additionally, even the tool-module combinations that have the largest datasets improve with data aggregation. Compared to their single-set results in Figure 7.6, the AUC scores of the aggregated models reach over 0.8 for both T2-PM2 and T7-PM1, as shown in Figure 7.6. These results underscore the effectiveness of data aggregation in bolstering model accuracy, especially for tool-module combinations with limited datasets.

Candidate Dataset Selection Method

While the previous section shows that data aggregation improves model performance, it does not explain how it should be conducted. Additionally, there is no easy way to evaluate a model's performance without actually developing and testing the model. Even in Figure 7.6, the displayed datasets had varying levels of improvement. Thus, to minimize the presence of false positives, it is necessary to examine how to optimally aggregate datasets together. To that end, five datasets (T4-PM1, T7-PM1, T2-PM2, T5-PM2, T7-PM2) are examined to assess the effectiveness of the indexing method explained in Section 7.2.3 that is used to select ideal datasets to aggregate with the base dataset, also called candidate datasets. The difference scores for each dataset pair as calculated with Equation (7.8) are shown in Table 7.5:

Table 7.5: Difference score between each pair of datasets.

	T4-PM1	T7-PM1	T2-PM2	T5-PM2	T7-PM2
T4-PM1	N/A	0.019	0.017	0.083	0.037
T7-PM1	0.019	N/A	0.011	0.083	0.032
T2-PM2	0.017	0.011	N/A	0.080	0.029
T5-PM2	0.083	0.083	0.080	N/A	0.101
T7-PM2	0.037	0.032	0.029	0.101	N/A

The candidate dataset selection criterion for a given dataset is to choose the paired dataset with the smallest difference score. For example, to improve the T4-PM1 model, the T4-PM1 dataset should be aggregated with the T2-PM2 dataset, as the T4-PM1/T2-PM2 pair has the smallest difference score (0.017) compared to the T4-PM1/T7-PM1 (0.019), T4-PM1/T5-PM2 (0.083), and T4-PM1/T7-PM2 (0.037) pairs. For the same reason, the candidate datasets for T5-PM2 and T7-

PM2 are both T2-PM2; the T5-PM2/T2-PM2 (0.080) pair is the smallest amongst all the T5-PM2 pairings, and the same holds for the T7-PM2/T2-PM2 pair. Note that the candidate dataset relationship is not necessarily true in reverse. While the candidate dataset for T5-PM2 may be T2-PM2, the candidate dataset for T2-PM2 is not necessarily T5-PM2. From Table 7.5, it can be seen that the candidate dataset for T2-PM2 is actually T7-PM1, with a difference score of 0.011. Additionally, to aggregate three datasets, or when choosing the second candidate dataset, the difference score has to be recalculated between the current two-set superset and all the other datasets. For example, to determine the candidate dataset for the T4-PM1/T2-PM2 superset (SS1), the difference scores for the SS1/T7-PM1, SS1/T5-PM2, and SS1/T7-PM2 pairs must be recalculated.

To further assess the effectiveness of this data aggregation metric, four datasets are examined in a case study: all three PM2 datasets and T4-PM1. It is still possible to exhaustively test the 15 possible unique supersets formed from these four datasets, which will allow us to evaluate both the idea that data aggregation generally improves model performance and the effectiveness of the candidate dataset selection method. First, the best possible AUC score for each tool-module combination is found by exhaustively creating a model for every possible superset and then selecting the superset that yields the highest AUC score for that tool-module. The best possible AUC score at each superset size is shown in Figure 7.8. Then, the best possible AUC score is compared to the AUC score obtained by aggregating candidate datasets, and this is shown in Figure 7.9 for two-set supersets and Figure 7.10 for three-set supersets.

Figure 7.8 validates the idea that data aggregation generally improves model performance because the best possible AUC scores for all four tool-module combinations increase as more datasets are aggregated. Additionally, in Figure 7.9 for two-set aggregation and Figure 7.10 for three-set

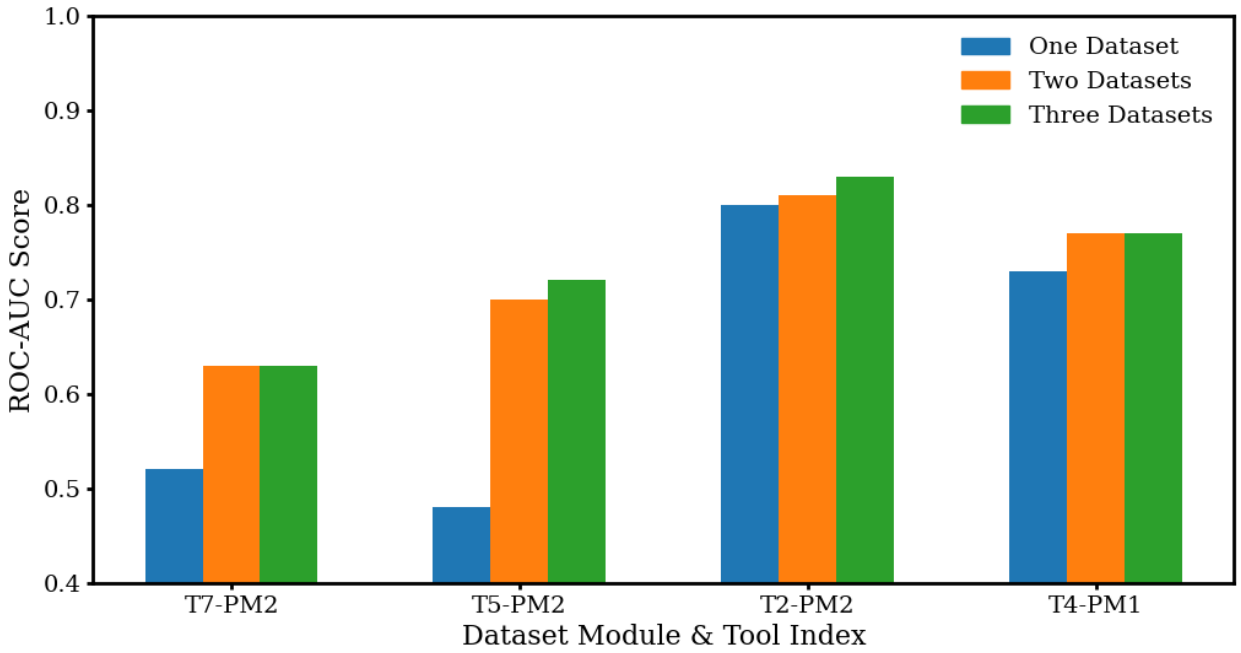


Figure 7.8: Best possible AUC score as a function of how many datasets are aggregated among the four datasets. Model performance improves significantly for all tool-modules as data aggregation increases.

aggregation, the AUC scores of the proposed data aggregation strategy based on statistical analysis methods aligns with the best possible score, with T7-PM2 being an exception in both cases. The exception of T7-PM2 can be explained by examining the difference score between its historical data and its future data. In Table 7.6, the difference score between the historical data used to train models and the future data used to test models is displayed for five tool-module combinations. Notably, the difference score for T7-PM2 is the largest of the examined datasets, which implies that there is significant variability between the modeling set and testing set that may impact data aggregation effectiveness as the data aggregation strategy is purely based on the historical dataset. This indicates that the proposed statistical analysis methods are effective for this case study and can be considered as a preliminary solution for selecting candidate datasets for aggregation.

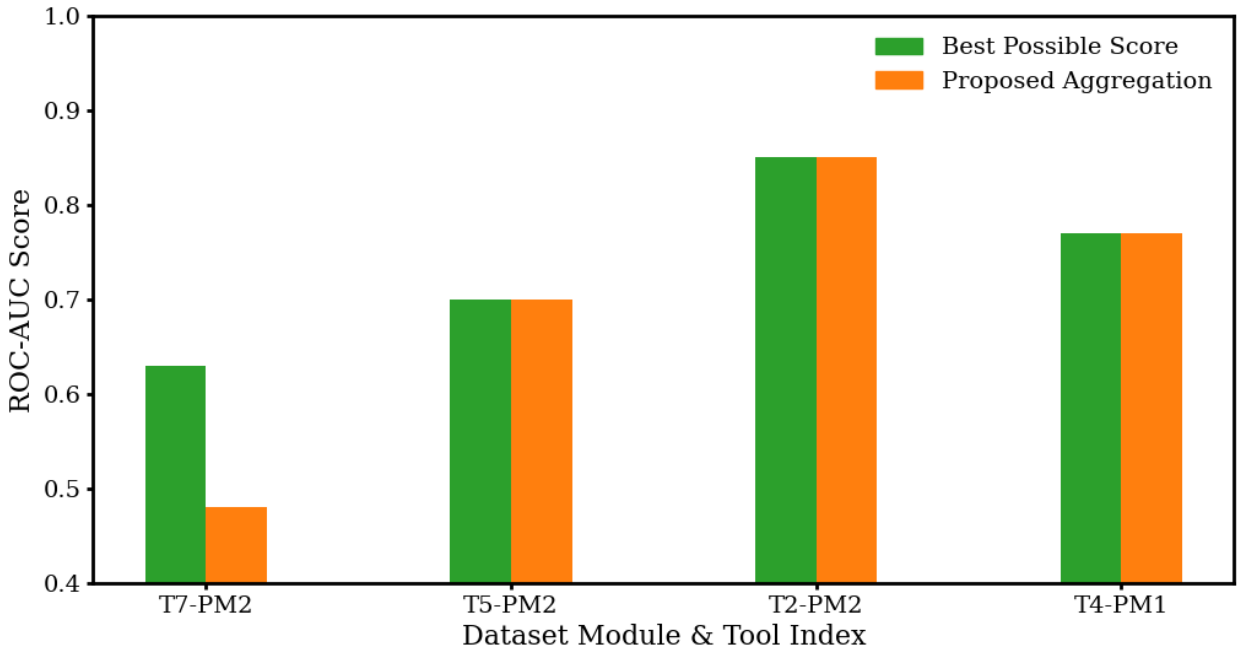


Figure 7.9: Comparison between the best AUC score for a **two-set** superset and the AUC score for the superset proposed by the candidate dataset selection method. Proposed AUC scores conform with the best-case scores for most tool-modules.

Table 7.6: Historical and Future Data Difference Score.

Dataset Name	Difference Score
T4-PM1	0.089
T7-PM1	0.052
T2-PM2	0.082
T5-PM2	0.090
T7-PM2	0.093

One of the most important applications of the candidate dataset selection method is to address scenarios where numerous tool-module combinations are involved. In these situations, it is impractical to exhaustively test all possible combinations of data aggregation to determine the ideal superset for each tool-module combination. To further test the capabilities of the candidate dataset

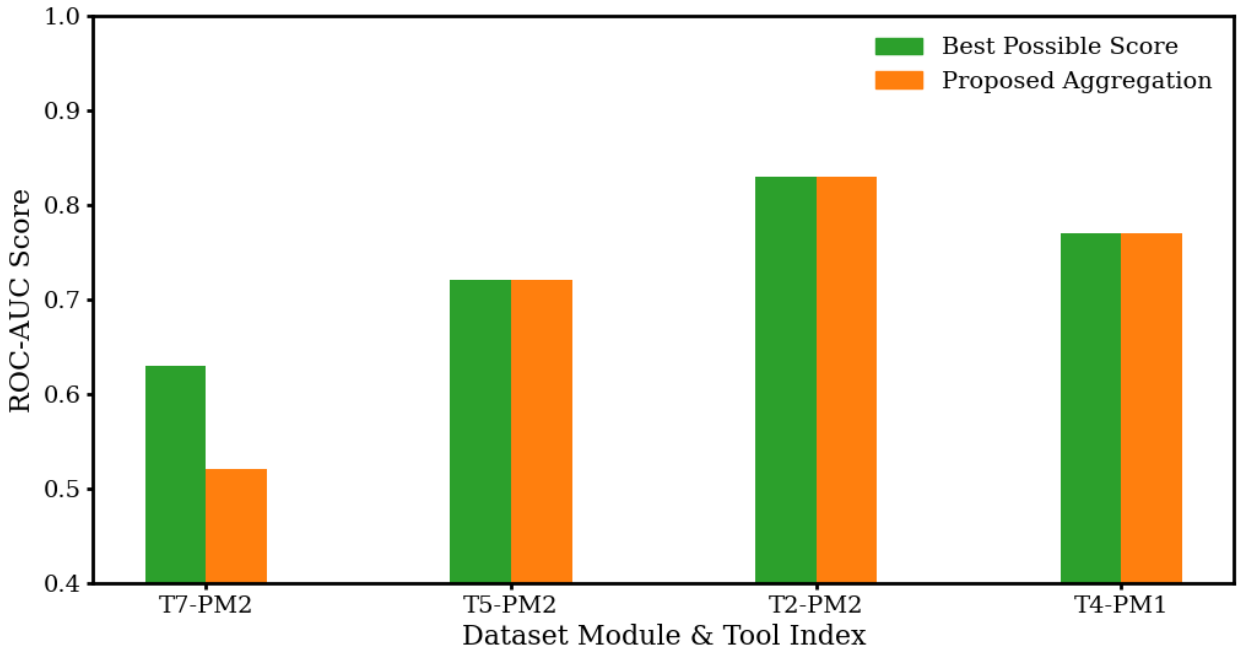


Figure 7.10: Comparison between the best AUC score for a **three-set** superset and the AUC score for the superset proposed by the candidate dataset selection method. Proposed AUC scores conform with the best-case scores for most tool-modules.

selection method, the five datasets in Table 7.5 are reexamined. For this analysis, not all possible superset combinations are tested. Instead, only the supersets proposed by the candidate dataset selection method are examined. The resulting AUC scores are then evaluated to validate the effectiveness of the proposed aggregation method that was previously demonstrated to work well for the four dataset case study. This approach aims to show that the aggregation strategy will continue to be feasible even as the number of datasets increases and prove that statistical analysis methods can identify the ideal candidate dataset without training numerous models. Among the five datasets, the candidate dataset for T2-PM2 is T7-PM1 as this pair has the lowest difference score (0.011), and the remaining datasets have the same candidate datasets as in the previous case study. The aggregation AUC scores are shown in Figure 7.11, and a specific example is shown in Figure 7.12.

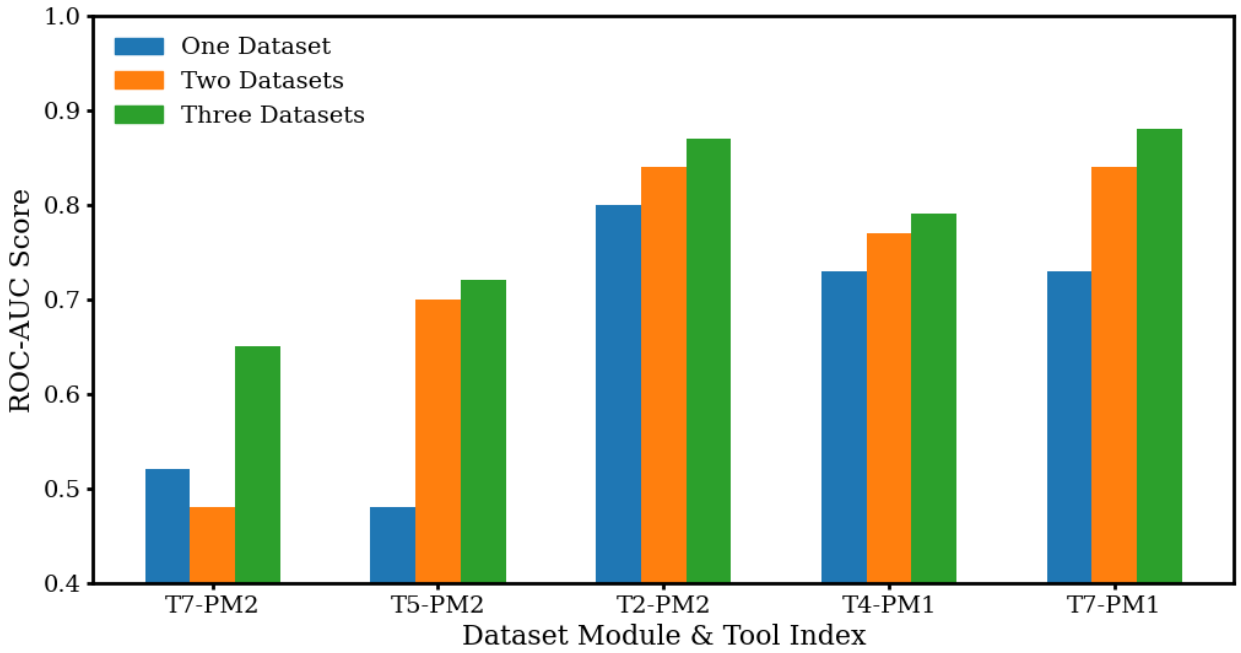


Figure 7.11: AUC scores for the models trained on the supersets proposed by the candidate dataset selection method. The AUC scores for all the tool-module combinations improve as more datasets are aggregated into the modeling set.

Figure 7.11 illustrates a noticeable improvement in the performance for all the tool-module models. Specifically, T2-PM2 and T7-PM1 achieve AUC scores of 0.87 and 0.88, respectively, which is considered high performing. As shown in Figure 7.12, the ROC plots for T2-PM2 and T7-PM1 indicate that aggregating data has lowered the FPR from between 35%-40% to 20% at 80% TPR, representing a significant improvement.

Figure 7.11 also demonstrates the importance of data volume and how data aggregation can supplement datasets with small volumes. This is especially in the case of T7-PM2, which previously underperformed in the four dataset case study. table 7.2 states that it has a small volume of data of 2722 compared to T4-PM1, T7-PM1, and T2-PM2, which have dataset sizes in the tens of thousands. For the cases where only one or two datasets are used, the T7-PM2 model performs

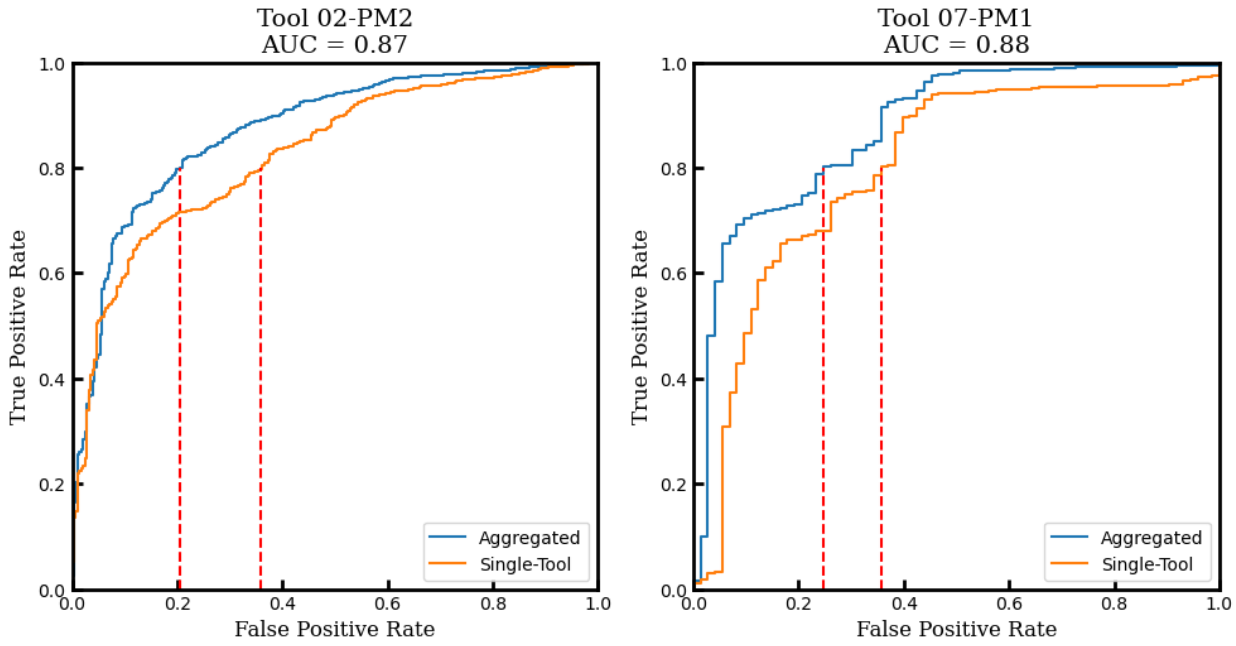


Figure 7.12: ROC plots for model performance on T2-PM2 and T7-PM1, which are trained by aggregating three datasets. The performances are noticeably better than the models trained on single datasets. The FPR values at 80% TPR are improved from good (around 40%) to perfect (around 20%).

poorly, with an average ROC-AUC score of 0.5, which is the same as randomly guessing. However, by aggregating three datasets, it achieved an AUC score close to 0.7 and an FPR of about 40% with a TPR of around 80%. This demonstrates the effectiveness of the data aggregation strategy based on statistical analysis methods at enhancing model performance, particularly in scenarios where there are so many datasets that it is impractical to exhaustively test all possible combinations.

While data aggregation is certainly powerful, it also has its own limitations. The two tool-module combinations with the smallest datasets, T5-PM2 and T7-PM2, cannot attain a FPR of 20% at a TPR of 80% like T2-PM2, a tool-module with a large dataset. This is because the original dataset is simply too limited. Although data aggregation can greatly increase the amount of training data by combining similar datasets, the newly added data will never perfectly follow the distribution of the original data, or in other words, data aggregation dilutes the "identity" of the original dataset. This forms a complex balance between data aggregation increasing the data volume, which improves model performance, and data aggregation diluting the identity of the original dataset, which decreases model performance. Additionally, the improvement effect from increased data volume experiences diminishing returns as seen in Figure 7.11; T7-PM1's performance experiences a sizable increase when moving from one to two datasets, but its performance only experiences a minor boost when moving from two to three datasets. Thus, the diminishing returns of increased data volume and the cost of identity dilution implies that there exists an optimum where further data aggregation would lead to decreased rather than increased model performance.

This optimum will also differ for each individual classification problem and dataset. For example, complex problems naturally require larger datasets in comparison to simpler problems, which reduces the diminishing returns effect of increased data volume. Additionally, the dilution extent

of the original dataset's identity is dependent on the potential candidate datasets; if the datasets to be aggregated are very similar, then the dilution effect will be weak whereas if the datasets to be aggregated are very different, the dilution effect will be strong. Nonetheless, the improved AUC scores and reduced FPRs highlight the efficacy of data aggregation at limiting the number of missed misprocessed wafers. A key part to data aggregation is the candidate dataset method and its ability to search for ideal datasets that optimize model accuracy and efficiency when dealing with extensive datasets as randomly selecting datasets will result in minor improvements if not decreases in performance. It should be mentioned that the point-biserial analysis and difference scores are not guaranteed to nominate the ideal candidate dataset. Nevertheless, the results of this chapter demonstrate that the proposed data aggregation method can still significantly improve the performance of industrial classification models, validating the practicality and effectiveness of the approach in real-world applications.

7.3.2 Regression Model Performance

The regression model is evaluated with the median percentage error metric. Percentage error is chosen as it is commonly used in industrial settings for numerical data that spans a wide range from 0 to over 5000 Å, and it is calculated with the following equation:

$$J_{percentage} = \frac{|y_{pred} - y_{true}|}{y_{true}} \tag{7.12}$$

where $J_{percentage}$ is the percentage error, y_{pred} is the predicted oxide thickness, and y_{true} is the measured oxide thickness. y_{true} spans from 0 to 5000 Å, and some values are close to 0 Å, e.g.

0.001 Å. The percentage error for these data points with small y_{true} values will be abnormally large despite their small absolute error. To minimize the influence of these outliers, the median, rather than the mean, is used to represent the overall percentage error as it is a more robust and representative measure of model performance. The R^2 criterion is not applied here because of the imbalanced data distribution; the data points are dense around a low target region (from 0 to 200) and sparse at a high target region (from 200 to over 5000). A model that mostly focuses on the dense, low target region can still receive a good R^2 score even if it performs badly on the sparse, high target region. Thus, the median percentage error is a better criterion for evaluating the overall model performance on all ranges of data. As previously stated, the regression task is more complex than the classification task, so only the T2-PM2, T4-PM1, and T7-PM1 datasets are analyzed. Furthermore, since the training, validation, and test datasets for the regression task are all sorted by time rather than randomly selected as was the case for the classification task, the median percentage error of all three datasets is an informative metric and will be shown.

Since the target oxide thickness is always known, a benchmark model can be constructed. Specifically, the benchmark model is defined as a model that always predicts the measured oxide thickness to be the same as the target oxide thickness; it always guesses that the product will pass metrology. Then, by calculating the median percentage error of this benchmark model on the validation set, it can act as a baseline for performance evaluation. The minimum requirement for an acceptable regression model is to outperform the benchmark model. By comparing the performance of various regression models against this benchmark, we can better assess the regression model's accuracy at predicting the measured oxide thickness, especially in an industrial context where reliability is crucial.

MAPE Training Results

The results for all the models trained with the Mean Absolute Percentage Error (MAPE) loss function for the three specified datasets split by the training, validation and test datasets are shown is shown in Figure 7.13. Figure 7.13 shows that, although the trained regression model outperforms

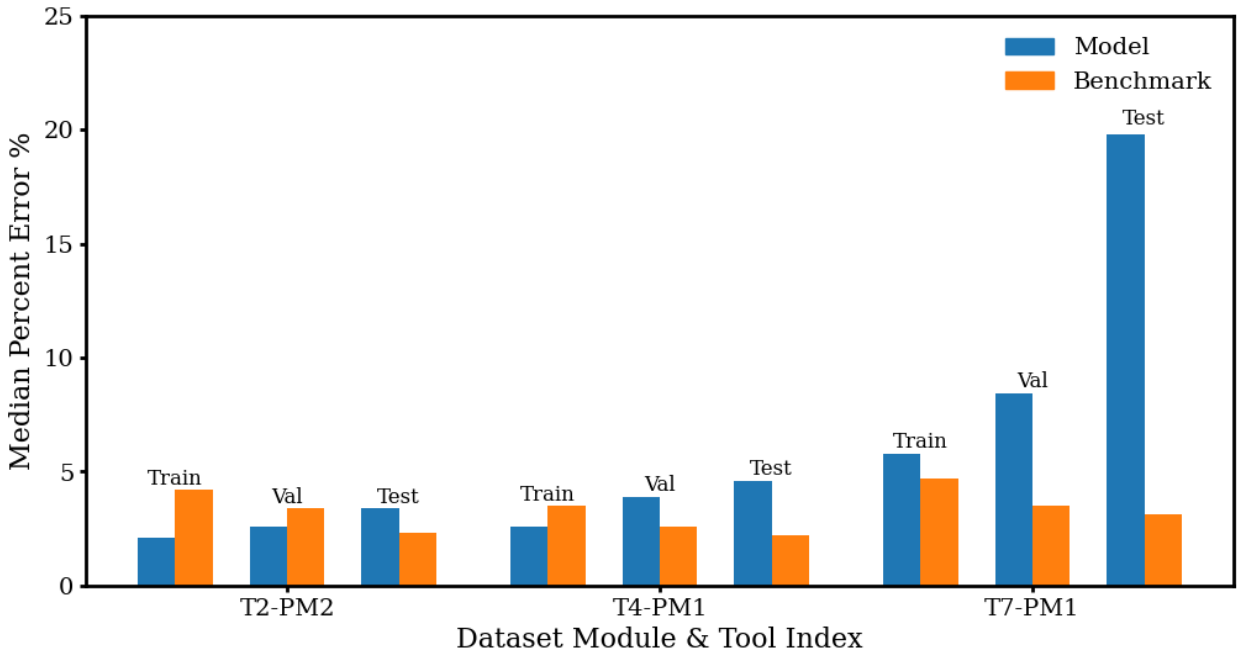


Figure 7.13: The median percentage error after training with the **MAPE** loss function, compared to the benchmark model. Each tool-module labeled on the x-axis has three groups of bars, which represent the training, validation and test sets, respectively. Within each groups of bars, the blue bar is the median percentage error of the trained regression model, and the orange bar is the median percentage error of the benchmark model. Figure 7.14 to Figure 7.18 have the same formatting.

the benchmark model on the training and validation datasets for T2-PM2, the regression model still underperforms in comparison to the benchmark model on the test set. For T4-PM1, the trained regression model even fails to outperform the benchmark model on the validation set. Moreover, for T7-PM1, the trained model does not outperform the benchmark model on any of the dataset. Overall, it is evident that any regression models trained with the MAPE loss function cannot surpass

the benchmark model when evaluated over the entire dataset.

Even when the data is divided into runs with a target oxide thickness value lower than 200 Å (low target) and runs with a target oxide thickness higher than 200 Å (high target) as shown in Figure 7.14 and Figure 7.15, the previous conclusion remains consistent. The regression model trained with the MAPE loss function does not outperform the benchmark model on either the low target or the high target test sets. The designation of "low" and "high" targets is arbitrarily determined by the real-world behavior of the tool-module combinations. Specifically, the runs with a target below 200 Å and those above 200 Å have very different behaviors. By splitting up the data and analyzing it in detail, it gives a more detailed and comprehensive sense of the overall model performance by examining the model performance in different applications.

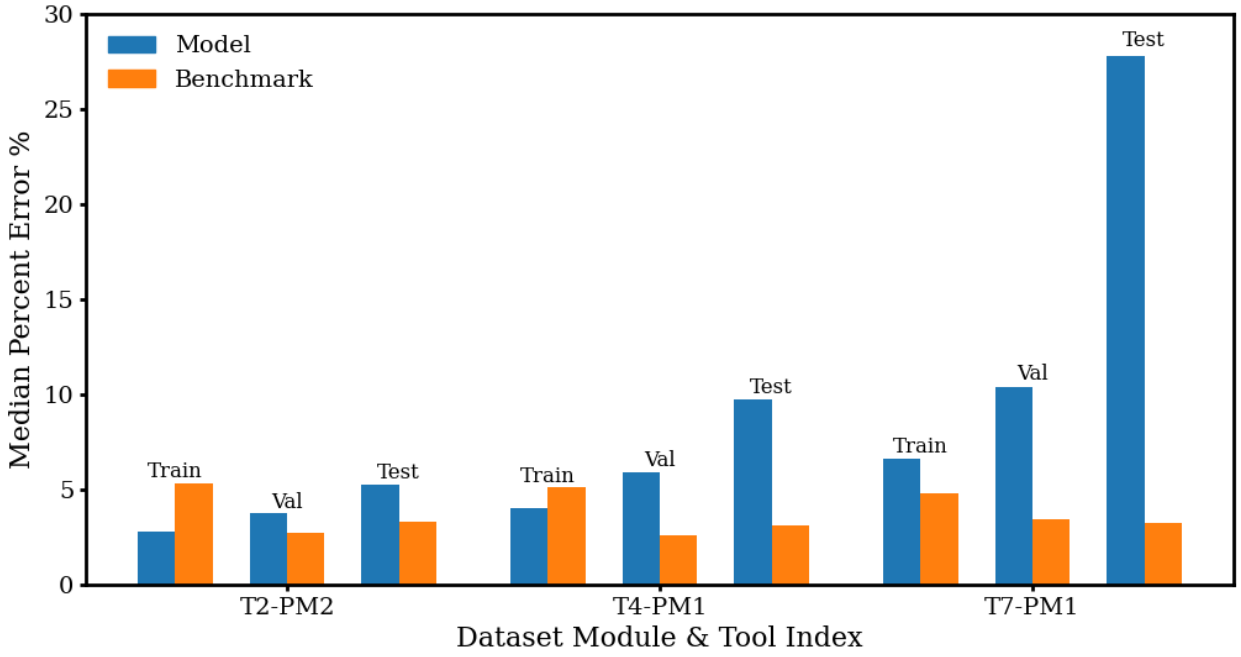


Figure 7.14: Comparison of the median percentage error between various regression models trained with the **MAPE** loss function for the training, validation and test sets and the benchmark model on runs with target oxide thicknesses of **less than** 200 Å.

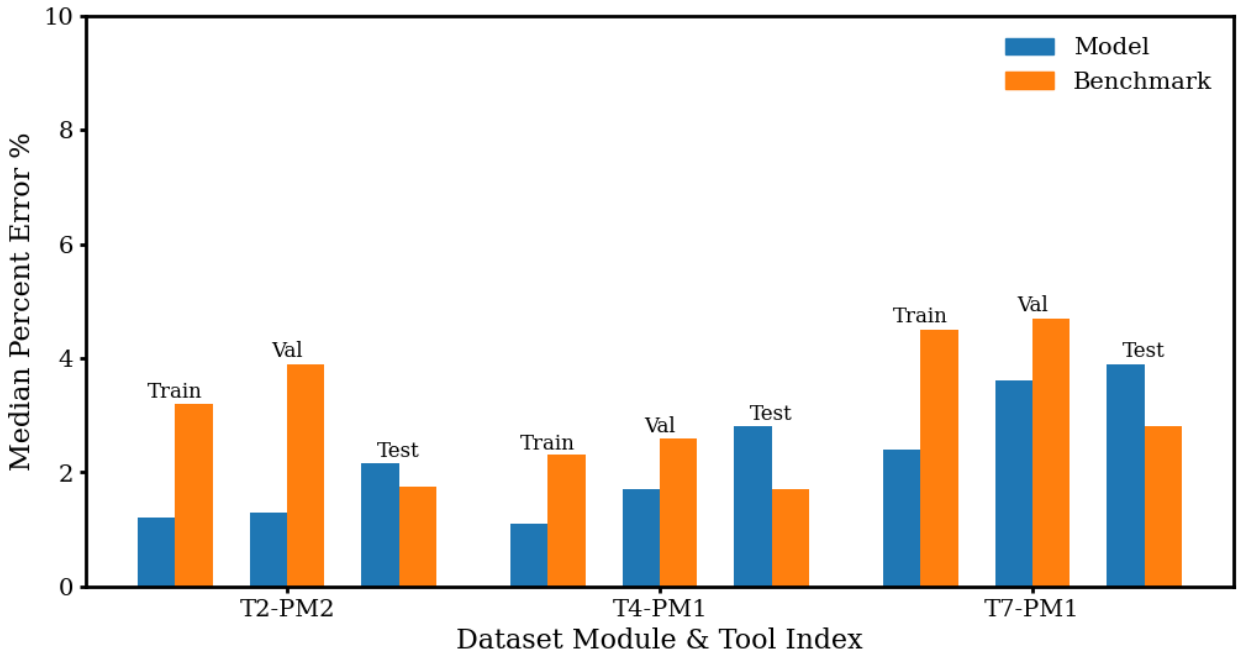


Figure 7.15: Comparison of the median percentage error between various regression models trained with the **MAPE** loss function for the training, validation and test sets and the benchmark model on runs with target oxide thickess of **over** 200 Å.

MSE Training Results

The results for all the models trained with the Mean Square Error (MSE) loss function for the three specified datasets split by the training, validation and test datasets are shown in Figure 7.16. Figure 7.16 shows that the median percentage error of the regression model trained with the MSE loss function is worse than that of the regression model trained with the MAPE loss function. This is expected as the MAPE loss better aligns with the evaluation criterion. The regression models trained with the MSE loss function perform worse than the benchmark model for all three tool-module combinations across the training, validation, and test sets. This trend is also evident in the model's performance on low target data points, as shown in Figure 7.17. Due to the intrinsic properties of the MSE loss function, which treats absolute errors on the high and low target data

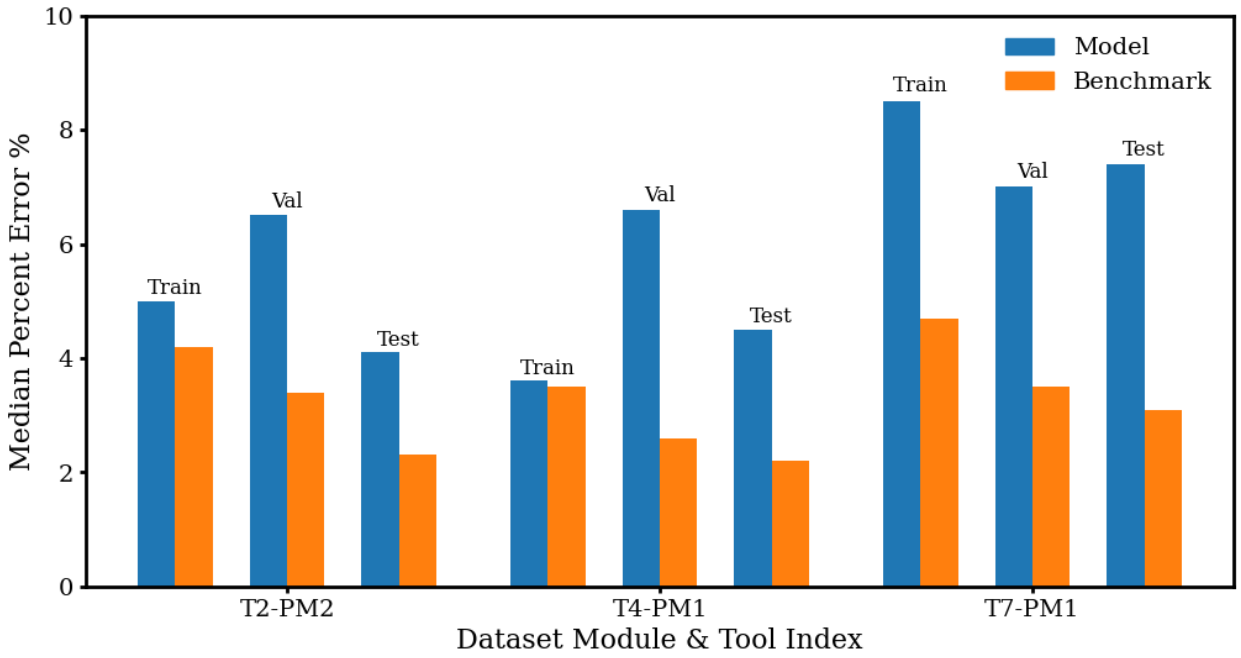


Figure 7.16: The median percentage error after training with the **MSE** loss function, compared to the benchmark model. For all tool-module combinations and all datasets, the trained regression model performance is worse than that of the benchmark model.

points the same, the median percentage error on runs with low targets is even higher than the median percentage error on all runs. Conversely, this means that the model performs exceptionally well on runs with high targets, as shown in Figure 7.18. When only examining the runs with high targets, the median percentage error of the trained regression models is significantly lower than those of the benchmark model across the training, validation, and test datasets.

The results shown in Figure 7.16 and 7.17 and 7.18 prove that the MSE loss function can achieve exceptional performance in terms of median percentage error for runs with high targets. Nevertheless, if the runs with high targets only constitute a negligible portion of the overall dataset, then the good performance of the regression model in this category is not particularly noteworthy. In this chapter, however, the high target data points contribute significantly to the overall dataset, as

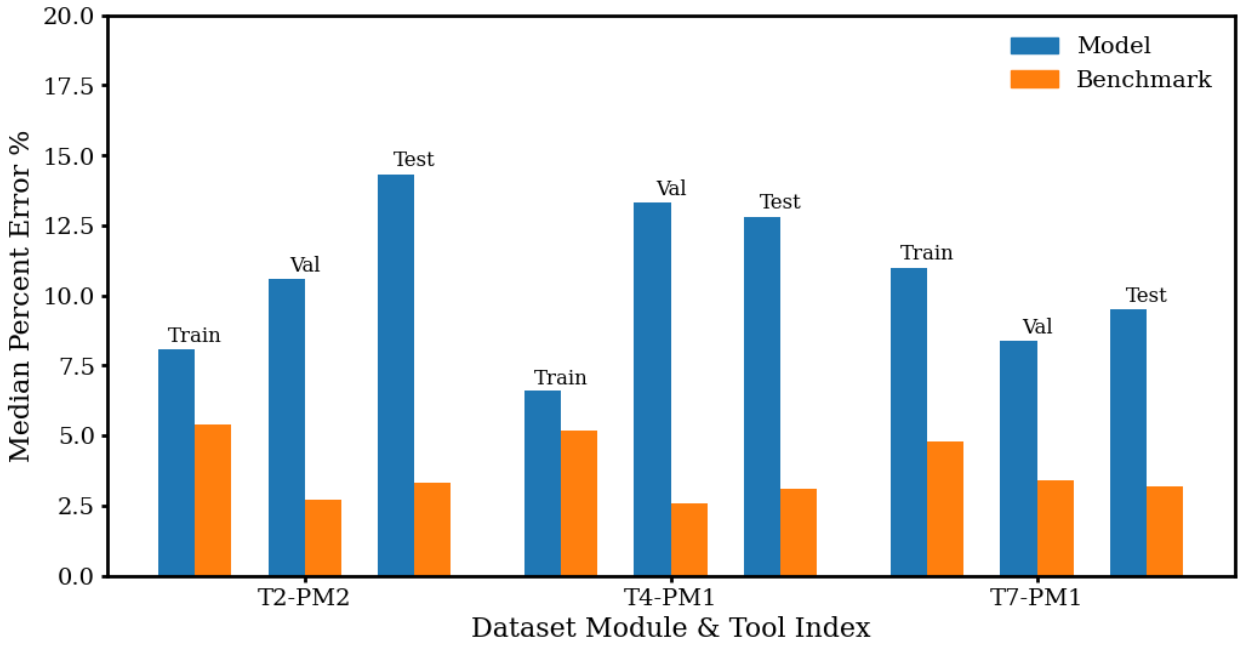


Figure 7.17: Comparison of the median percentage error between various regression models trained with the **MSE** loss function for the training, validation and test sets and the benchmark model on runs with **low** target oxide thicknesses. The performance of the regression model here is even worse than in Figure 7.16.

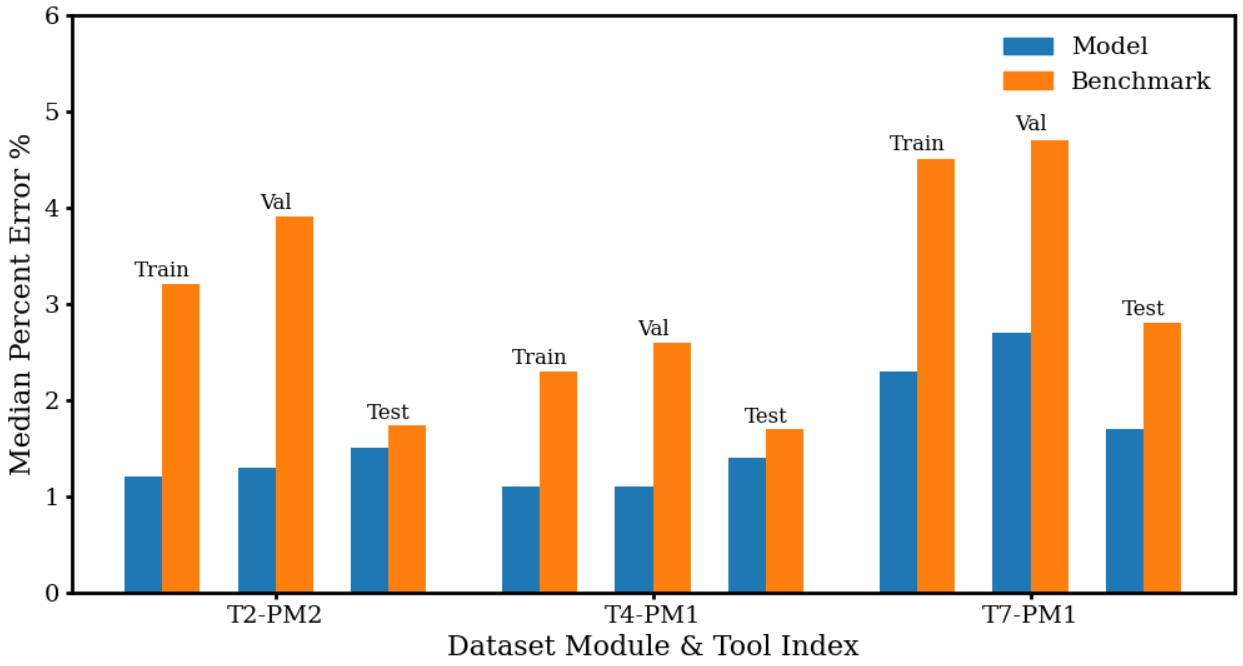


Figure 7.18: Comparison of the median percentage error between various regression models trained with the **MSE** loss function for the training, validation and test sets and the benchmark model on runs with **high** target oxide thicknesses. The performance of the regression model is better than that of the benchmark model on all examined runs for all three training, validation and test datasets.

shown in Table 7.7. For instance, over one-third of the data points in the test sets of T2-PM2 and

Table 7.7: Percentage of runs with High Targets in each dataset.

Tool-Module	Training	Validation	Testing
T2-PM2	20%	17%	38%
T4-PM1	21%	39%	38%
T7-PM1	8%	18%	11%

T4-PM1 have high target oxide thicknesses.

Given this substantial representation, although the model trained with the MSE loss function cannot be confidently used in all cases, it holds a distinct advantage in predicting the measured oxide thickness when the target oxide thickness is high. Since the target value is always known before processing, this characteristic can be strategically utilized to employ the model in areas where it shows strong performances.

The poor performance of the regression models for both the MSE and MAPE loss functions, compared to that of the benchmark model, can primarily be attributed to the exceptionally high pass rate, which reaches 98% in some datasets. In industry, a common pass criterion is for the result to be within a specified threshold of the target value. This high pass rate indicates that most runs are successfully etched and that their measured oxide thickness is very close to the target thickness, resulting in the benchmark model having a very low median percentage error in most cases. However, this means that it is very difficult to train a regression model that outperforms the benchmark model across all target values. This stands in stark contrast to the results of the classification model, where the favorable ROC-AUC curves in Figure 7.12 show that the trained models can and do outperform a benchmark model that randomly guesses the outcome.

The reason for the different results stems from the difference in complexity between the tasks.

The available process data contains enough information to train a high-performing classification model that simply allocates the results into two classes. However, the regression model must predict the exact oxide thickness value from a wide range of possible values. As the regression task is many times more complex than the classification task, a high-performing regression model requires process data that contains deep insight into the process itself. Thus, the reason why the regression models do not outperform the benchmark model is because the process data does not contain enough information regarding the process. To improve the performance of the regression models, more complex process data, such as time-series data, must be incorporated into the training process. In conclusion, machine learning-based soft sensors are powerful at predicting the physical properties of the process, but only if the model is trained on sufficiently insightful process data collected from physical sensors.

7.4 Conclusion

This chapter describes machine learning-based soft sensors trained on process data from five industrial etching reactors for two tasks: PASS/FAIL classification and oxide thickness regression. A data aggregation method is proposed to improve the model performance for the predictive classification task. In addition, for the regression task of predicting measured oxide thickness, both MSE and MAPE losses are tested for model training. The results presented in this chapter validate the hypothesis that data aggregation and statistical analyses can significantly enhance the binary PASS/FAIL prediction accuracy and robustness of feed forward neural network models on industrial etching reactors. This is done by aggregating datasets using point-biserial correlations and

difference scores as metrics to choose candidate datasets for aggregation, particularly for tools with initially small data volumes. The two datasets with the smaller number of data points improved from a score barely above random guessing ($AUC \approx 0.5$) when trained on a single dataset to a significantly better performance ($AUC \approx 0.65$) with a 40% false positive rate (FPR) at a 80% true positive rate (TPR) when trained on an aggregation of three datasets out of five possible datasets. The other datasets with relatively larger datasets still benefited from data aggregation; their model performance improved from $AUC \approx 0.8$ to $AUC \approx 0.9$, and the FPR improved from around 35% at 80% TPR to to 20%. The statistical analyses accurately chose the best candidate dataset for aggregation, aligning with the results shown by exhaustively testing every possible dataset combination. These results confidently show the effectiveness of data aggregation and using statistical methods as an aggregation strategy. Additionally, for regression models that predict oxide thicknesses, while the MAPE loss function was more effective for overall percentage error minimization, the trained model could not outperform the benchmark model that always predicted the measured oxide thickness to be the target thickness. Although the model trained with the MSE loss function did not yield satisfactory predictions across all data points, it exhibited exceptional performance for runs with high target oxide thickness with target values of over 200 Å. This finding is useful for specific processes with high target values, as these target values are typically known in industrial settings. Consequently, the strategic use of MSE for processes with high target values can enhance regression model performance in these specific scenarios.

Chapter 8

Industrial Multi-Machine Data Aggregation, AI-Ready Data Preparation, and Machine Learning for Virtual Metrology in Semiconductor Wafer and Slider Production

8.1 Introduction

As the transition to the digital era accelerates and the Internet of Things (IoT) grows, the demand for microelectronic devices is skyrocketing. The Semiconductor Industry Association (SIA) announced that global semiconductor industry sales totaled $574.1 billion in 2022, the highest ever annual total and an increase of 3.3% compared to the 2021 total of $555.9 billion. The industry shipped a record 1.15 trillion semiconductor units in 2021, as chip companies ramped up production to address high demand amid the global chip shortage [160]. These are devices equipped with

integrated circuits such as central processing units (CPUs), graphics processing units (GPUs), hard drives, and solid state drives [136]. Demand for these devices has long driven the need for higher transistor densities and narrower gate widths to improve computing performance and reduce power consumption rates [4]. Today, the increase in demand has led to recurring shortages [161], negatively affecting global economies in various sectors, for example, the automotive industry [162]. There is a substantial incentive in semiconductor manufacturing to increase production volume, increase product quality and precision, and increase productivity without only increasing operational equipment and personnel. Digital transformation and leveraging Industry 4.0 for Smart Manufacturing are pivotal with how to take advantage of operational data, AI/ML, IoT, information technology, and interconnectedness at greater scales to open new avenues of increased productivity, performance and precision [163]. Smart manufacturing is, by definition, about using real-time data and modeling at scale. It is the intersection of plant-wide agility and optimization, sustainable production and resilient demand-driven supply chains made interoperable through interconnectedness with trust and meaningfully shared data and tools [164].

Smart Manufacturing at scale depends on operating sensor data collected, contextualized, and aggregated from sensors on factory floor operations. Specifically, sensors are installed on each machine for each process operation, and they process and/or collect data in real time. These data are used to manage, control, and optimize the performance of each individual machine tool and process operation. They are useful for multiple management objectives such as monitoring, diagnosis, operational health, preventive maintenance, faster changeovers and quality assurance [165]. These data are critical to automation and autonomous operations. When used across line and factory operations, the data are used to address interoperability, agility, and higher-level key performance

indicators in combination. Line operation and factory interoperability extend into supply chain interoperability. Cross company data that provide visibility into supply chain material and product flows and product demands and capacities are needed for resilience. In addition to the benefits of applying data and models to improve physical operations at all levels, these same data are also valuable assets that can be aggregated to build richer data sets for model building [16]. The data can be categorized, discovered and used to validate models and when synchronized with models, the resulting digital twin systems can be used for real-time preventive, proactive, predictive control, management and optimization [166].

In this chapter we focus on data as valuable assets and demonstrate how data from multiple similar machines can be aggregated into richer datasets for more robust machine learning (ML) model building for all machines and/or how multiple machines can be optimized together. We focus on method and demonstration of virtual metrology (a.k.a., soft sensing) as one important application in semiconductor manufacturing. Unlike direct measurement methods, such as an offline quartz microbalance to analyze layer thickness manufacturing [141], ML virtual metrology predicts product quality measurements or condition from operating sensor data by using these data to train models that can, within the range of operating conditions experienced, associate operating conditions to quality measurement [15]. The advantage of soft sensing lies in its ability to overcome the drawbacks of direct sensing, which can be expensive, time-consuming, untimely and/or labor-intensive [142]. There can be situations in which the measurement costs more than the product. There are also situations in which technical measurement complexity, product conditions, and or measurement objectives make physically impossible to do a direct measurement, e.g., key performance indicators (KPIs).

In contrast, virtual metrology uses data generated during production processes. These are data that embed operational effects for the range of operations experienced. The data, collected from widely available sensors are relatively inexpensive. The engineering of the data and the modeling process, if done systematically and carefully, has been shown to be highly effective in measuring physical parameters that are difficult to directly assess [167]. Benefits from these models can accrue quickly. However, these ML models are limited to operating range experienced. It is therefore crucial to pay close attention to the operating conditions within which the reliability of the model is high. Modern manufacturing processes have become increasingly complex, generating vast amounts of data from dozens or even hundreds of sensors [168]. There can be data volumes and operational complexities that exceed human assimilation. Machine learning and deep learning techniques have emerged as promising methods. They perform well in capturing complex non-linear correlations between inputs and outputs, and it has been demonstrated that neural networks of sufficient scale can approximate any non-linear functions [169].

Several types of neural networks have been successfully implemented in production settings: for example, Convolutional Neural Networks (CNNs) [146] and Recurrent Neural Networks (RNNs) [147]. Transformer networks have been used by companies including Seagate Technology for fault detection in etch tools [149]. Although recent advancements in ML have prompted extensive research into the application of soft sensing across various industries, challenges such as data insufficiency, sensor noise, and the presence of redundant sensors remain significant problems. Furthermore, the issue of process drift over time complicates the effectiveness and reliability of models built on historical data.

In this chapter, ML virtual metrology is tailored for both plasma etching and slider production.

Specifically, datasets from multiple plasma etching and slider production tools were consistently collected and contextualized by applying CESMII's Data Information Model structure, called a Profile [164]. The Profile ensures that the measurements from each of the machines could be concatenated for further cross-machine analyses and uses of the data. Importantly, the chapter recognizes that these AI/ML/DS models depend on, in fact thrive on the "right" data when there is enough that is engineered appropriately to build, train, test, and validate the AI/ML models. A "Data-First" strategy emphasizes the importance of data available in the amounts, forms, scale, and access necessary to achieve benefits in productivity, jobs, market share, sustainability, and growth. A Data-First strategy emphasizes engineering AI-Ready data that is consistently, collected, ingested, contextualized, cleaned, normalized, protected, aggregated, assessed, and selected to draw out the operational value that has been refined from years of experience even when the physics may not be fully understood. Data-First also addresses the primacy of data access in implementing affordable AI/ML solutions including how to obtain, categorize, scale, and use AI-Ready data. This includes how to validate, learn, unlearn, and adjust models when using and reusing this invaluable data.

This chapter explores the impact of data modification and manipulation techniques, including dimension reduction, data aggregation, and data augmentation, on model performance to address these challenges. Principal Component Analysis (PCA) is employed as a dimension reduction technique to eliminate redundant features and reduce noise. Building upon a prior work [16], this study introduces a separate scaling method designed to normalize input datasets more effectively, thereby enhancing model performance. Despite the common use of Feedforward Neural Networks (FNN) in such applications, decision tree methods based on gradient descent boosting, which are well-suited

for handling large feature sets with lower computational training costs, have rarely been explored in this context. To address the issue of process drift, a live-updating transfer learning approach is implemented with FNN models. This approach periodically updates the model using up-to-date data to reduce the need for physical measurements, leveraging a base model trained on historical data. Additionally, this chapter discusses and applies a data augmentation method, linear Mixup, to mitigate commonly encountered data insufficiency issues, which may arise from sensor failures or pre-mature process lines. The Mixup method has been shown to significantly improve model performance as data-adaptive regularization, offering a promising solution to these pervasive challenges [170].

this chapter is organized as follows: Section 8.2.1 provides an overview of the datasets with Section 8.2.2 going more in depth on the preprocessing procedures for both etching and slider production tools, Section 8.2.3 presents the machine learning model construction for both classification and regression tasks, Section 8.3.1 and Section 8.3.2 demonstrate and discuss the model performance results on both tools, and Section 8.4 summarizes the findings of this chapter.

8.2 Data Preprocessing and Modeling

8.2.1 Overview

This section provides an overview of the solution objective, data collection, physical functionality, and specific machine tools for the use case used in this study. As mentioned, the use case encompasses five plasma etching machines at various Seagate factories used in wafer production distinct from those examined in previous research [16], and an additional seven slider tools from the pro-

duction of the slider component used in hard disk drives (HDD). The 12 datasets, one from each machine, made it possible to extend the prior study on data aggregation on similar machines and processes to a study on similar machines but different processes. The solution objective remained the same as the previous study [16], in which the model was constructed to use machine tool data to determine (predict, before metrology) if the final thickness of the deposited layer PASS or FAIL. PASS refers to if the wafer is expected with high probability to fall within a specified quality range, and FAIL is if it does not. As in the previous study, a classification model was applied. In addition, a regression model was also developed for the slider production process to quantitatively predict the depth of the milling process. Both models aim to determine whether the output products of these tools are expected to meet specific metrological standards. If the performance of these models is sufficiently high, this virtual metrology application can increase the machine and staffing productivity of the physical metrology quality assurance process.

The datasets produced by the etching process facilitate the development of a binary classification model to perform fault detection. This model classifies the final thickness of the deposited layer as PASS if it falls within a predefined range and FAIL if it does not. For the slider production process, a regression model is developed to predict the depth of the milling process. This model aims to determine whether the outputs of these tools meet specific metrological standards, thus serving as an effective soft sensor for quality assurance for multiple products in manufacturing processes.

The details of the data collection and description of the tools and modules of datasets were elaborated in a prior work [16] and demonstrated in Figure 8.1. In summary, the manufacturing process requires raw wafers to undergo a sequence of process steps to transform into complete products. Due to the repetitiveness of these processes, a wafer may interact with the same kind of

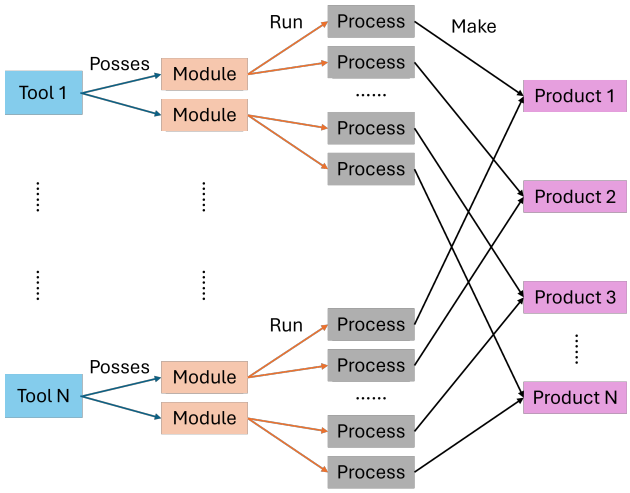


Figure 8.1: The industrial etching equipment's manufacturing system consists of multiple etching reactors, each equipped with several modules capable of running different processes. The production begins with pure silicon wafers, which undergo a series of processing steps before transforming into the final product.

tool multiple times at different processes. this chapter analyzes etching and slider production tools, where each tool functions as a reactor and contains modules that operate as nearly independent reaction chambers to conduct specific processes. Tool-module combinations are designated in the T-PM format; for instance, 'T1-PM1' refers to module 1 in tool 1. For the rest of the chapter, a data point is defined as a data vector of all features after averaging across the duration of the batch. A dataset is the collection of data points obtained from a specific tool-module combination. Given these definitions, this section will explain how each type of industrial dataset is processed and the structure of their corresponding machine learning algorithm.

8.2.2 Data Generation and Preprocessing for Etching and Slider Tools

Wafer Production Etch Tools

Operational data from each wafer production etch tool was collected and consistently organized by the CESMII Information Model or Profile pictured in Figure 8.1. The data from each machine comprised two data types: 32 streamed, time-based numerical sensor measurements and 2 categorical features. The streamed, numerical data, sourced from various physical sensors, include operational measurements such as pressure and gas flow, i.e., state variables that can be measured directly and in real-time. Categorical features describe process and material specifics that are decisions about the use of the machine. The two features included are an alphanumeric process ID that captures the "recipe" of machine functions for a particular product and the substrate family ID. Each family is a finite set of classes. The steamed operational data was averaged for each variable over duration of a batch run which is typically on the order of 20 minutes. Data collected cover five years of operation, from February 2018 to December 2022. The five etch tool datasets are categorized by machine number and process module (single machines that have parallel etch modules) as T51-PM1, T52-PM1, T52-PM2, T53-PM1, and T53-PM2.

Again, the solution objective is to build a binary prediction model by mapping these operating data to PASS and FAIL outcomes. Data from 2018 through 2021 are allocated for training, validation and the tuning of the model hyperparameters. The 2022 data, the most recently collected, were reserved for testing. Selecting data from the oldest rather than the newer sets and then testing with the most recent data ensures that model effectiveness with respect to "normal" operational shifts and machine changes over time are accounted for and evaluated in the context of current and

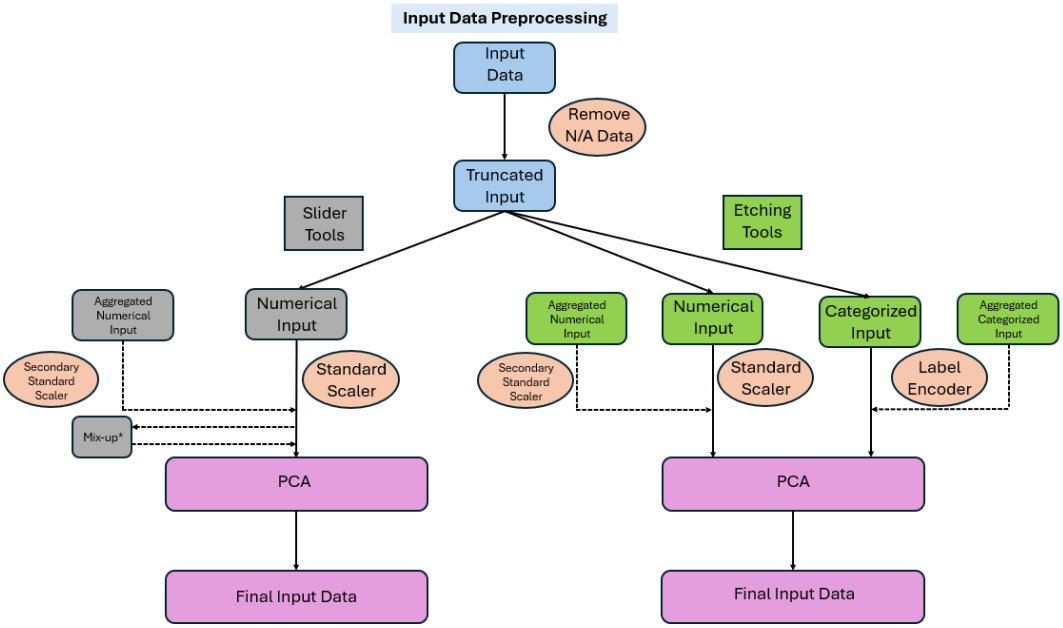


Figure 8.2: This figure illustrates the input data preprocessing workflow for etching tools and slider production tools. Common steps include invalid data removal, normalization, and dimension reduction. Etching tools incorporate encoders for categorical data management, whereas slider tools utilize Mixup to mitigate data scarcity.

future production scenarios.

In addition to data selection, data preprocessing is also crucial for preparing datasets for model training process. Prior studies [151, 171] have highlighted that preprocessing can significantly impact model training performance, underscoring the importance of selecting those preprocessing methods that optimize model performance for an operational situation [172]. In this chapter, preprocessing involves removing or padding invalid/null data points, encoding discrete variables, and normalizing numerical data points for maximum performance accurately. The workflows for preprocessing inputs and outputs are graphically illustrated in Figure 8.2 (input data) and Figure 8.3 (outputs).

With reference to Figure 1, the first step is remove invalid data. An invalid data point is defined

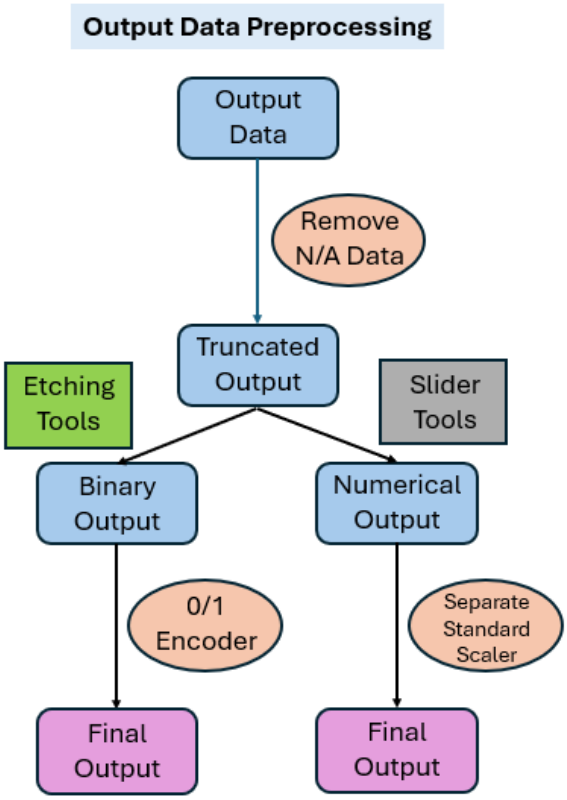


Figure 8.3: This figure illustrates the output data preprocessing workflow for etching tools and slider production tools. Common steps include invalid data removal, etching tools requires 0/1 encoding, and slider tools requires separate scaling.

when critical sensor data are absent, such as the output measurements of oxide thickness, the pass/-fail status, or when a substantial number of the input sensor data is missing. Missing data points can be attributed to the temporary malfunction of sensors within a batch run, not uncommon with industrial data. In scenarios where extensive numbers of sensor data are missing (usually over half, at least 10 features), zeroing all missing features or feature paddings could detrimentally affect the dataset' s integrity and, consequently, the model training performance. For sensor data points in which a small number of the measurements (usually below three features, rarely over five features) are missing, these missing values are replaced by the average value of the remaining data points in the data set. This imputation method helps maintain the consistency and reliability of the dataset, ensuring that the training process is not skewed by gaps in the data. This strategy of handling missing data preserves the underlying statistical relationships and prevents the introduction of bias that could mislead the learning algorithm. We note that the sensor data is collected as time-series measurements and averaged over the duration of each processing step to generate representative values for various features. Once this data is properly recorded, it is treated as accurate and reliable, as the sensors on the machine have been qualified by operations to ensure proper functionality and measurement accuracy.

Conversion of categorical features into numerical formats is essential for their integration into numeric-based machine learning models. Each categorical feature, such as the process ID and the substrate family ID, must be uniquely encoded using methods such as label encoding or one-hot encoding. In this chapter, label encoding was used to convert process name IDs and substrate family IDs into numerical values. To ensure consistency across different machines, substrates, and production processes, the encoding was created using all available datasets combined, covering all

possible entries. This approach prevents issues that could arise if the encoder were trained on a limited dataset. For example, if a tool or module only runs certain processes, an encoder trained only on that data might fail when applied to another tool with different processes, leading to errors due to missing values. By including all possible entries during encoding, we ensure that the data remains compatible across different tools and processes. For the output variable, which is either PASS or FAIL condition, binary encoding scheme is adopted, where '1' represents a PASS and '0' denotes a FAIL.

Normalization and scaling of data are critical for optimal training performance, when the input features span wide numeric ranges and scales. For instance, in this use case some features might range from 0 to 1, while others range from 0 to 100,000. Normalization was critical to avoiding gradient vanishing or gradient explosion [153]. Although gradient-boosted, tree-based methods like XGBoost and AdaBoost are inherently less sensitive to scale differences due to their reliance on decision tree structures, normalization still plays a significant role. Unscaled data can introduce inefficiencies in node splitting such that features with larger value ranges can dominate the feature selection in split decisions, potentially leading to suboptimal splits that do not capture the true underlying patterns in the data effectively.

To ensure all features are normalized consistently across different datasets, a standard scaler is applied to each dataset independently, normalizing the features to a mean value of 0 and a standard deviation of 1. This scaling process adheres to the formula:

$$Z = \frac{X - u}{s} \tag{8.1}$$

where Z is the output vector of a scaled numerical feature, X is the input vector of the original feature, u is the average value of X, and s is the standard deviation of X. In the context of data aggregation, a method detailed in a previous work [16], where model training involves combining multiple datasets to improve the model performance by increasing data variability and data volume, it is found to be beneficial to scale each dataset separately prior to concatenating datasets. In other words, the concatenations should happen between scaled datasets but not raw data. This method is necessary because, within each module's reaction chamber, the distribution of features often varies significantly. For example, although the same feature across different datasets may be similar in scale, direct scaling of the aggregated dataset could distort the inherent distributions of each dataset if the differences in scale and standard deviation are substantial. As shown in Table 8.1, the mean percentage difference of average values across all features exceeds 40% between most tools, indicating significant distribution shifts between datasets. This distribution distortion can mislead the model if the datasets are concatenated before normalization, then impacting its performance. Consequently, separate scaling for each dual-tool combination is implemented to maintain the integrity of their original distributions relative to the output. This approach preserves crucial relationships within the data, ensuring more reliable and accurate model training outcomes.

Table 8.1: Mean Percentage Difference of Average Value of All Features

	T51-PM1	**T51-PM2**	**T51-PM3**	**T52-PM2**	**T53-PM2**
T51-PM1	N/A	64%	14%	90%	43%
T51-PM2	77%	N/A	75%	55%	46%
T51-PM3	20%	86%	N/A	101%	66%
T52-PM2	5865%	3208%	5274%	N/A	9932%
T53-PM2	47%	45%	45%	81%	N/A

After the normalization and aggregation process, dimension reduction is facilitated using Prin-

cipal Component Analysis (PCA). It is worth noting that PCA requires the data to be normalized before its application. Normalization standardizes the range of features, ensuring that each feature contributes equally to the analysis and that features with larger scales do not dominate the principal components. This prerequisite is crucial because PCA is sensitive to any variance in the initial variables. The PCA process begins by calculating the covariance matrix for the data, which identifies the directions in which the data varies the most. If the data are not normalized, features with higher absolute values could disproportionately influence the covariance matrix, leading to biased principal components that do not accurately reflect the underlying data structure. Subsequently, the eigenvalues and eigenvectors of this covariance matrix are computed. The eigenvalues are sorted in descending order, and their corresponding eigenvectors are aligned accordingly. The principal components are selected based on these sorted eigenvalues, those that explain the most variance are retained (in this chapter 99.9% of variances are retained), discarding the less significant components (non-contributing features, noise). This selection is critical because features that exhibit higher variance are considered more influential for the model' s predictive accuracy. The calculation process is shown below:

$$Q = X^T X = W \Lambda W^T \tag{8.2}$$

where X represents the data matrix with dimension $(k \times d)$, where k denotes the number of data points and d represents dimensions of datasets. Q refers to the covariance matrix, which is a square matrix of dimension $(d \times d)$. Λ is a diagonal matrix consisting of eigenvalues of Q arranged in descending order. Correspondingly, W is a matrix of the eigenvectors of Q, aligned in the same

sequence as the eigenvalues in Λ. The linear transformation from the original data space to the principal components space is represented by equation below:

$$T_L = XW_L \tag{8.3}$$

where L is the number of reduced dimensions determined by the proportion of variance to be retained, T_L represents the transformed data matrix with dimension $(k \times L)$, and W_L is the first L columns of W matrix.

Each principal component is a linear combination of the original features, representing a new direction in the feature space. By focusing on these principal components, PCA effectively reduces the dimensionality of the data. This reduction not only compresses the data with minimum loss by emphasizing directions with most significant variations, but also helps in mitigating the risk of overfitting by reducing feature numbers and noise in the data. In this chapter, 99.9% of variances are retained, and the resulting feature space has dimension between 10-15, corresponding to 60% to 75% reduction in dimension.

Slider Production Tools

Slider tools are a different part of the manufacturing process from wafer production where machine tools are used to make specific parts used in the production of data storage devices. For this study, we focused on a milling machine step which is used to establish a critical surface depth for the slider component. The modeling objective is the same virtual metrology solution for wafer production in that milling machine sensor data were collected, contextualized, processed and engineered for building a model to predict milling depth. In this study we have benchmarked data processing

methods and workflow, and we compare and contrast the approach to building the virtual metrology solution for two different production applications. As with wafer production, modeling begins with data analysis. Since the goal is to predict milling depth, a continuous variable, a regression model is developed. This model leverages multiple features, approximately 20 in total, to estimate milling depth with precision. However, an important observation from the data is that milling depth values tend to cluster into discrete ranges, corresponding to specific product specifications. Figure 8.4 illustrates these clusters, with milling depths predominantly falling within groups of 1450–1550, 1650–1850, and 2150–2350. These clusters reflect the limited variety of products, each with its own depth requirement, and are an inherent characteristic of the manufacturing process.

Given this natural grouping, the regression output can be easily mapped into predefined depth ranges, allowing for a multi-class classification approach. Instead of building a separate classification model, the regression predictions are assigned to the nearest cluster, effectively classifying each wafer' s milling depth into a corresponding product category. Additionally, this classification approach aids in identifying failures. Failures are defined as wafers whose milling depth deviates significantly from the expected clusters for a given tool-module combination. For example, if a particular tool is designed to produce only one type of product, any wafer with a milling depth outside its designated range is considered a failure, even if its depth aligns with a valid range for another product. In this context, failures indicate deviations in production quality, highlighting potential process anomalies. This depth-failure classification enhances both the model's practical utility and its ability to detect defects in real-world manufacturing.

For classification, data points outside of the major depth ranges are flagged as failures, with the distribution details of these principal ranges cataloged in Table 8.2. The datasets relevant to

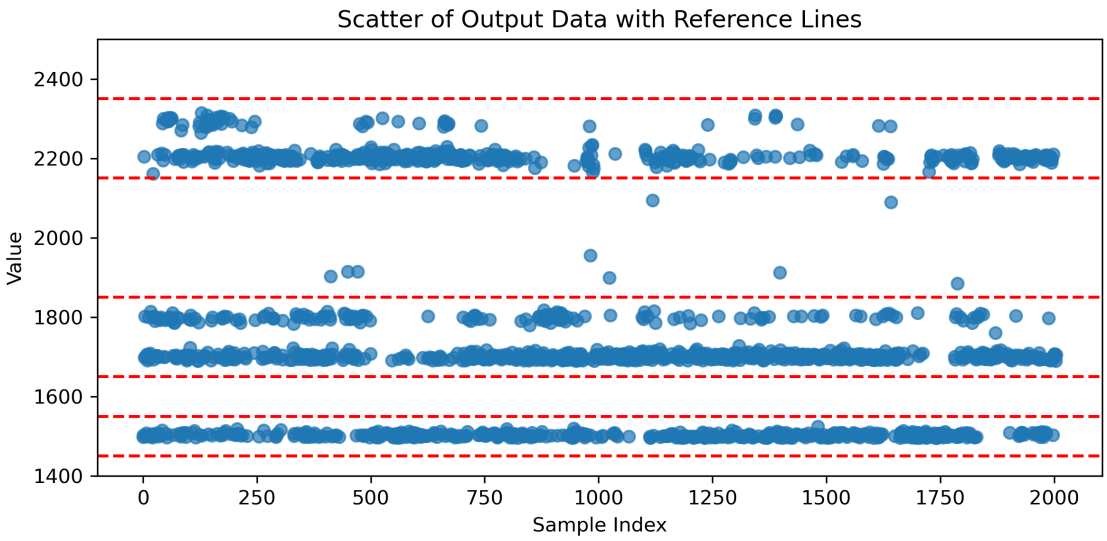


Figure 8.4: Scatter plot of T15-PM1 output, illustrating how data points are primarily grouped into several clusters.

slider production, namely T07-PM1, T07-PM2, T15-PM1, T02-PM1, T02-PM2, T01-PM1, and T05-PM2, consist of 20 features each, similar to the datasets from etching tools but encompassing fewer features and data points. The organization of data points within major depth categories for each dataset is also documented in Table 8.2, where red numbers mean data points in outlier regions.

Table 8.2: Data Distribution Across Different Ranges

	1450–1550	**1650–1850**	**2150–2350**	**Others**	**Total**
T07-PM1	1	869	9	22	901
T07-PM2	7	992	4	11	1014
T15-PM1	518	943	530	15	2006
T01-PM1	98	198	158	11	465
T02-PM1	0	742	108	24	874
T02-PM2	1	685	18	24	728
T05-PM2	0	153	0	3	156

Data preprocessing for these datasets follows a similar approach to that used for etching tools, including the removal of null values, padding of missing data, normalization, aggregation, and

dimension reduction. However, since the slider production datasets do not contain categorical features, the encoding step is not required. As shown in Figure 8.2, unique to slider tools, however, is the application of separate scaling on the output data during aggregation, as the outputs of etching tools datasets are binary values. In this aggregation process, one dataset is designated as the primary or 'major' dataset aimed at performance enhancement, while other datasets serve as supplementary or 'helper' datasets. The output scaler is tailored to the distribution of the major dataset before being applied to the combined dataset to preserve its integrity. This strategy is crucial because using a universal scaler across the aggregated data could distort the distribution of the major dataset of interest, such as the single-region T07-PM1 versus the three-region T15-PM1.

After normalization and aggregation, PCA is applied to the input data to retain at least 99.9% of the original variance, significantly reducing dimensionality while preserving essential information. The number of principal components retained varies by dataset, typically resulting in about 10 principal components, effectively halving the original feature set. This dimensionality reduction is critical in managing the complexity and enhancing the interpretability of the models developed from these high-variance datasets.

Due to the limited data availability for slider production tools as characterized by a shorter data collection period of only one year compared to five years for etching tools and a lower frequency of data recording, enhancements in model performance are necessary. To address this, a data augmentation technique known as Mixup was employed to create artificial data points by interpolation between existing data points.

However, given that the slider data belongs to several discrete regions, interpolating data points between major regions was avoided since doing so could lead to non-existing milling depth. For data

aggregation, each dataset undergoes Mixup independently within its respective major regions, and the resulting post-processed datasets are then concatenated. This ensures that the augmented data remains representative of the true distribution patterns observed in the production environment. To evaluate the impact of different interpolation strategies on model effectiveness, two distinct Mixup schemes were tested. The first scheme is tri-point interpolation, which creates two artificial data points between each two data points. This scheme is formulated as follows:

$$\begin{aligned} x_{mix} &= 0.33x_i + 0.67x_{i+1},\ x_{mix} = 0.67x_i + 0.33x_{i+1} \\ y_{mix} &= 0.33y_i + 0.67y_{i+1},\ y_{mix} = 0.67y_i + 0.33y_{i+1} \end{aligned} \tag{8.4}$$

The second scheme creates one data point directly in the middle of two data points as formulated as follows:

$$\begin{aligned} x_{mix} &= 0.5x_i + 0.5x_{i+1} \\ y_{mix} &= 0.5y_i + 0.5y_{i+1} \end{aligned} \tag{8.5}$$

Beyond data augmentation, Mixup also acts as a form of regularization, effectively smoothing the decision boundaries of the model. This smoothing is achieved by encouraging the model to perform linear interpolations between features and their associated targets in the input space, which can reduce the model's confidence in far-reaching predictions. Such a characteristic is particularly useful in mitigating overfitting, as it prevents the model from learning overly complex patterns that are heavily dependent on the specific training data distribution. Instead, Mixup encourages the model to generalize better to new, unseen data by promoting a broader exploration of the feature space and reducing the likelihood of drastic output changes in response to small variations in input. This regularization effect makes Mixup a valuable technique in enhancing the robustness and

generalizability of machine learning models [170].

8.2.3 Model Training

Introduction to Machine Learning Models

Feedforward Neural Networks (FNNs) are widely used for modeling processes involving regression and classification under large amounts of data and features. However, tree-based ensemble methods like Extreme Gradient Boosting (XGBoost) demonstrate advantages when dealing with high-dimensional feature spaces and significant noise levels. This is because:

- FNNs require careful normalization, can consume more computational resources, and may suffer from overfitting in complex architectures.
- Tree-based approaches (like XGBoost) require less stringent normalization, often train faster on moderate size dataset, and are generally robust to overfitting due to inherent regularization.

In industrial applications, datasets often present challenges such as high-dimensional feature spaces, missing values, class imbalance, and the presence of significant noise. These characteristics make tree-based approaches particularly well-suited for industrial data modeling. The key advantages include:

- **Robustness to Noisy and Incomplete Data:** Industrial datasets frequently contain sensor readings, production metrics, and operational parameters that may be affected by measurement errors, environmental conditions, or missing entries. Tree-based approaches handles such inconsistencies effectively using its optimized split criteria and missing value handling mechanisms, allowing it to make robust predictions despite imperfect data.

- **Feature Selection and Interpretability:** Unlike FNNs, which often require careful feature engineering and are perceived as black-box models, tree-based approaches naturally identifies the most important features during training. This ability is critical in industrial settings where engineers and decision-makers need interpretable insights for process optimization, fault diagnosis, and predictive maintenance.

- **Computational Efficiency and Scalability:** Industrial datasets in this study range from 10,000 to 30,000 data points. Training deep neural networks on such data is more computationally expensive than XGBoost, even with a GPU, and require extensive hyperparameter tuning. Tree-based approaches, on the other hand, is optimized for speed and scalability for mid size datasets as in this chapter, leveraging parallel processing and tree-pruning techniques to efficiently handle large datasets with minimal computational overhead.

- **Regularization for Generalization:** FNNs, particularly deep architectures, require careful tuning of dropout rates and batch normalization to prevent overfitting. Tree-based approaches can incorporate built-in regularization techniques such as L1 (LASSO) and L2 (Ridge) penalties, along with shrinkage (learning rate adjustment), ensuring better generalization performance without extensive fine-tuning.

Given these advantages, tree-based approaches emerges as a highly effective tool for modeling industrial processes, where robustness, interpretability, and computational efficiency are paramount. XGBoost is one example of a tree-based ensemble method combining the predictions of multiple decision trees in a gradient-boosted framework. The algorithm begins by training a base tree, then computes the residual (the difference between the current prediction and the true label). Subsequent

trees are trained to optimize these residuals. The final prediction for sample x_i after t trees is:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \eta \cdot f_t(x_i), \tag{8.6}$$

where η is the learning rate and f_t is the t-th decision tree. This iterative approach refines the model incrementally, making XGBoost particularly effective in many structured data tasks, which are tasks that involve datasets where features are well-defined, typically organized in tabular format, and consist of numerical, categorical, or ordinal variables. These tasks are common in industrial settings, where data is collected from sensors, production logs, and quality control systems.

Model Training on Etching and Slider Tools

Etch tool data are used to train a ML classification model to perform a binary prediction of PASS or FAIL product at the completion of the machine tool process step. Because the FAIL rate from the process step is small($\sim 2\%$), the data are highly imbalanced between PASS and FAIL. We therefore use a weighted cross entropy loss approach to emphasize FAIL conditions. Cross-entropy loss, also known as log loss, is a commonly used loss function for classification tasks. It measures the divergence between the true labels and the predicted probabilities, penalizing incorrect predictions more heavily. In binary classification, it is defined as:

$$L(y_i, \hat{y}_i) = -w_0(1 - y_i)\log(1 - \hat{y}_i) \; - \; w_1 y_i \log(\hat{y}_i), \tag{8.7}$$

where w_0 and w_1 are class weights set as:

$$w_0 = \frac{1}{n_0}, \quad w_1 = \frac{1}{n_1}, \tag{8.8}$$

and n_0 and n_1 denote the number of PASS and FAIL samples, respectively. It's worth mentioning that in our datasets FAILs are usually represented as 1 instead of 0. This scheme emphasizes the minority class to improve the detection of FAIL events.

80% of the etch tooling dataset (2018–2021) was used for training, while the remaining 20% was separated out for validation. To systematically tune hyperparameters of the machine learning model, we performed a 5-fold cross-validation on the training data. Cross-validation is a technique used to assess the generalizability of a model by splitting the dataset into multiple subsets. In 5-fold cross-validation, the training set is divided into five equally sized folds, where the model is trained on four folds and validated on the remaining fold. This process is repeated five times, with each fold serving as the validation set once. The final model performance is averaged over all iterations, reducing the risk of overfitting and improving robustness.

To further optimize hyperparameters, we apply a grid search exploring various learning rates, the number of estimators, and regularization strengths for both FNN and XGBoost. Grid search systematically evaluates predefined combinations of hyperparameter values to identify the optimal set that maximizes model performance. This approach ensures that different configurations are tested comprehensively, balancing model complexity and predictive accuracy. The final selected hyperparameters are shown in table 8.3 and table 8.4, respectively, where the underlined parameters represent the chosen optimal values.

Table 8.3: Hyperparameters for FNN

Hyperparameter	**Range/Candidate Values**
Number of layers	[1, 2, 3]
Number of neurons	[16, 32, 64]
Activation function	['relu', 'sigmoid']
Dropout ratio	[0.0, 0.2, 0.5]
L2 regularization	[0, 10^{-4}, 10^{-3}, 10^{-2}]
Training epochs	[500, 1000, 2000]
Early stops	['Yes', 'No']

Table 8.4: Hyperparameters for XGBoost

Hyperparameter	**Range/Candidate Values**
Learning Rate	[0.1, 0.3, 0.5]
Max Tree Depth	[6, 8, MAX*]
Subsample Rate	[0, 0.5, 1]
L1 Regularization	[0, 1, 10]
L2 Regularization	[0, 1, 10, 100]

* Dimension after PCA

In all kinds of machine operation that run continuously over a long periods of time, process drift poses a significant challenge to maintain accurate model performance over time, as even the sensors themselves can drift and develop bias. Process drift occurs due to various factors such as equipment aging, accumulation of unwanted deposits on machinery, and other operational changes that alter system behavior with time. For example, in etch machines, process drift can manifest as a gradual reduction in etch rate due to the accumulation of impurities on chamber walls[173]. This buildup alters plasma conditions, leading to deviations in feature dimensions and film thicknesses over time. If not accounted for, such drift can degrade the predictive performance of machine learning models trained on historical data, necessitating periodic model recalibration.

As a result, models trained on initial datasets may gradually lose accuracy. Transfer learning addresses this by adapting existing models to new data without full retraining, preserving prior knowledge while integrating new information. This is especially valuable in industrial settings, where collecting sufficient data in different environments is costly due to the need for physical measurements. Additionally, in scenarios such as data drift, continuous updates are required, but new data may never be sufficient for training from scratch before being outdated.

Our implementation of transfer learning followed a three-step procedure designed to improve model performance while minimizing the need for costly physical measurements:

1. **Base Model Training:** The procedure begins with developing a base model by training the feedforward neural network (FNN) on old data, which in our case is the data from 2018-2021, to establish foundational performance metrics. This base model serves as the starting point for all subsequent updates and evaluations.

2. **Incremental Model Updating with SGD:** Once the base model is established, it is incrementally retrained on new data using stochastic gradient descent (SGD). The training data window size (i.e., mini-batch size) is a tunable hyperparameter, with examples including mini-batches of 1 or 10 data points, referred to as 'train data length' in Table 8.5. To prevent data leakage, each new data point is first evaluated using the original, pre-updated model before being used for training. Although the true label is available, the model is tested as if it had not yet seen this data point. Only after recording this evaluation is the model updated with the new data using SGD. This ensures an unbiased assessment of how well the model adapts over time while accurately measuring the effectiveness of the updating algorithm. The use of SGD with a carefully selected learning rate enables efficient updates, striking a balance between adapting to new information and preserving existing knowledge to prevent model destabilization.

3. **Model Evaluation on Test Data:** After the model is updated in the previous stage, it is now evaluated on new test data points. At this stage, the model is not updated—only its performance is measured. The ratio between the number of test data points and the training data points from the previous step is defined as the 'Test/Train Length Ratio' in Table 8.5. This evaluation step ensures that the model can make accurate predictions on new data without additional retraining. The goal of transfer learning here is to improve model performance while minimizing the need for physical measurements, which are costly and time-consuming. If the model were retrained on every new data point, it would require collecting new labels through physical measurements, defeating the purpose of transfer learning. Although all data in this

chapter are labeled through physical measurements, the labels are used solely to evaluate the effectiveness of transfer learning. From the model's perspective, it does not have prior knowledge of the labels during training. By testing the transfer learning scheme in this controlled setting, its effectiveness can be validated before future real-world applications. After completing this evaluation step, the process returns to the previous stage, where the model is updated with the next batch of new data points, continuing the learning cycle. After completing Step 3, the process returns to Step 2, where the model is updated with the next batch of new data points. This continuous cycle of evaluation and updating is known as real-time update, allowing the model to adapt dynamically to evolving data patterns.

Hyperparameters such as the learning rate, training data length, and test/train length ratio are optimized to achieve robust model performance. This optimization is conducted through a systematic grid search, as detailed in Table 8.5. For each tool, all combinations of train data length and train/test length ratio are thoroughly tested to examine the effectiveness of transfer learning.

Table 8.5: Hyperparameters for Transfer Learning Model Training

Parameter	**Hyperparameters**
Learning Rate	[0.0001, 0.001, 0.01]
Training Data Length	[1, 10]
Test/Train Length Ratio	[1, 2, 4]

In summary, transfer learning efficiently maintains model accuracy amid process drift by leveraging knowledge from historical data and incorporating incremental updates. This approach sustains performance while reducing the costs of full retraining, making it well suited for evolving industrial processes and the practical constraints of manufacturing environments.

For the slider datasets, the task is a regression problem instead of binary classification. Therefore, the Mean Squared Error (MSE) is employed as the primary loss function:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(\hat{y}_i - y_i\right)^2, \tag{8.9}$$

where $\hat{y}_i$ and y_i are the predicted and actual values for sample i, and N is the total number of samples.

Due to the limited dataset size (collected over a shorter time period), fewer layers and neurons are used in the FNN to prevent overfitting. XGBoost, however, retains its standard set of hyperparameters, exploiting its tree-based mechanism to handle even relatively small datasets. To prevent overly small partitions, we use an 80%–20% train/validation split without a separate time-based test set. Given the limited data volume and its confinement to a one-year range, time-based prediction is less critical.

Table 8.6: Hyperparameters for FNN

Hyperparameter	**Range/Candidate Values**
Number of layers	[1*, 2, 3]
Number of neurons	[8, 16, 32]
Activation function	['relu', 'tanh']
Dropout ratio	[0.0, 0.2, 0.5]
L2 regularization	[0, 10^{-4}, 10^{-3}, 10^{-2}]
Training epochs	[500, 1000, 2000]
Early stops	['Yes', 'No']

* Single Training

By applying FNN and XGBoost with these tailored hyperparameters and the MSE loss, we

achieve reliable regression performance within the constraints of a smaller dataset. Given the limited data, the FNN architecture is kept simple with fewer neurons to avoid overfitting and reduce computational costs.

8.3 Results and Discussion

8.3.1 Etching Tools

The performance of the machine learning models introduced in Section 8.2.3 that were trained on industrial data we received are best evaluated within the bounds of the specific machines, operational environments, and conditions in which they will be applied. As discussed in a previous study, the etch machines in this study perform well with low percentages of product FAIL rates; therefore, the data distribution is highly imbalanced toward PASS vs. FAIL runs [16]. There are four possible outcomes in a binary classification:

1. **True Positive**: Correctly identifying a PASS.
2. **False Negative**: Incorrectly labeling a PASS as a FAIL.
3. **True Negative**: Accurately detecting a FAIL.
4. **False Positive**: Incorrectly labeling a FAIL as a PASS.

True positives and true negatives indicate that the classification model is correctly qualifying the product state. False negatives, though undesirable, can be mitigated since all flagged failures undergo an additional metrology inspection where false negatives can be corrected by manual measurements. The larger concern in these operations are false positives, as they are a much more

expensive miscategorization compared to false negatives. False positives are more costly because downstream operational resources continue to process the defective product, wasting resources and time. In most cases, false positives will eventually be identified through later metrology steps or final product reliability testing [158].

An acceptable classification model for this operating situation needs to have sufficient accuracy and precision in identifying true FAIL and PASS results but also in minimizing false positives to reduce the cost of classification errors. In operational terms, this virtual metrology solution leads to significant economic benefit by debottlenecking the physical metrology step and increasing production volume.

The performance metrics for the classification models must reflect the ability of a model to control its false positive rate. Performance is adjusted by modifying classification thresholds that affect the balance among all four possible classifications. Increasing the sensitivity to misclassified wafers reduces false positives but raises false negatives and vice versa. For this application there is a need to demonstrate the ability to tune the model by considering performance across a range of threshold settings. Confusion matrices, while commonly used to evaluate classifier performance, only address results at a specific threshold [174]. They fail to capture overall model performance for various settings. Accuracy evaluation is unsuitable because of the extreme PASS vs. FAIL imbalance. For instance, a model that always predicts "PASS" may appears highly accurate because the vast majority of classifications are PASS, but is functionally useless with a 100% false positive rate.

Each model does not have a single, fixed false positive rate; rather, its performance can be adjusted by modifying classification thresholds. Increasing sensitivity to misprocessed wafers re-

duces false positives but raises false negatives, whereas a more lenient approach does the opposite. Confusion matrices, while commonly used to evaluate classifier performance, only depict results at a specific threshold and fail to capture overall model capability across various settings. Other common evaluation metrics, like accuracy, are also unsuitable due to the extreme class imbalance in manufacturing datasets. For instance, a model that always predicts ”PASS” may appear highly accurate but is functionally useless with a 100% false positive rate.

The Receiver Operating Characteristic (ROC) analysis is a better analysis tool for this use case because it directly plots the true positive rate (TPR) and false positive rate (FPR) across the entire span of thresholds. The output of the classifier produces continuous scores in the range [0,1] of the classification threshold. The default threshold is of 0.5, which is equivalent to guessing an outcome. We therefore want to be able to adjust this threshold to get an acceptable performance. Setting the threshold to 0 results in a TPR of 100% and an FPR of 100% (always PASS), while a threshold set at 1 results in TPR and FPR rates of 0% (always FAIL). The optimal threshold therefore requires balancing sensitivity (TPR) angainst false alarms (FPR). To objectively compare the performance of models, the Area Under the ROC Curve (AUC) is used since it is a metric that quantifies overall performance across all threshold settings when plotting TPR against FPR. A perfect model would achieve an AUC of 1, meaning 100% sensitivity with no false positives. In contrast, a model making random predictions would yield an AUC of 0.5. Since manufacturing data is typically imbalanced, ROC-AUC provides a reliable measure of classification effectiveness [159].

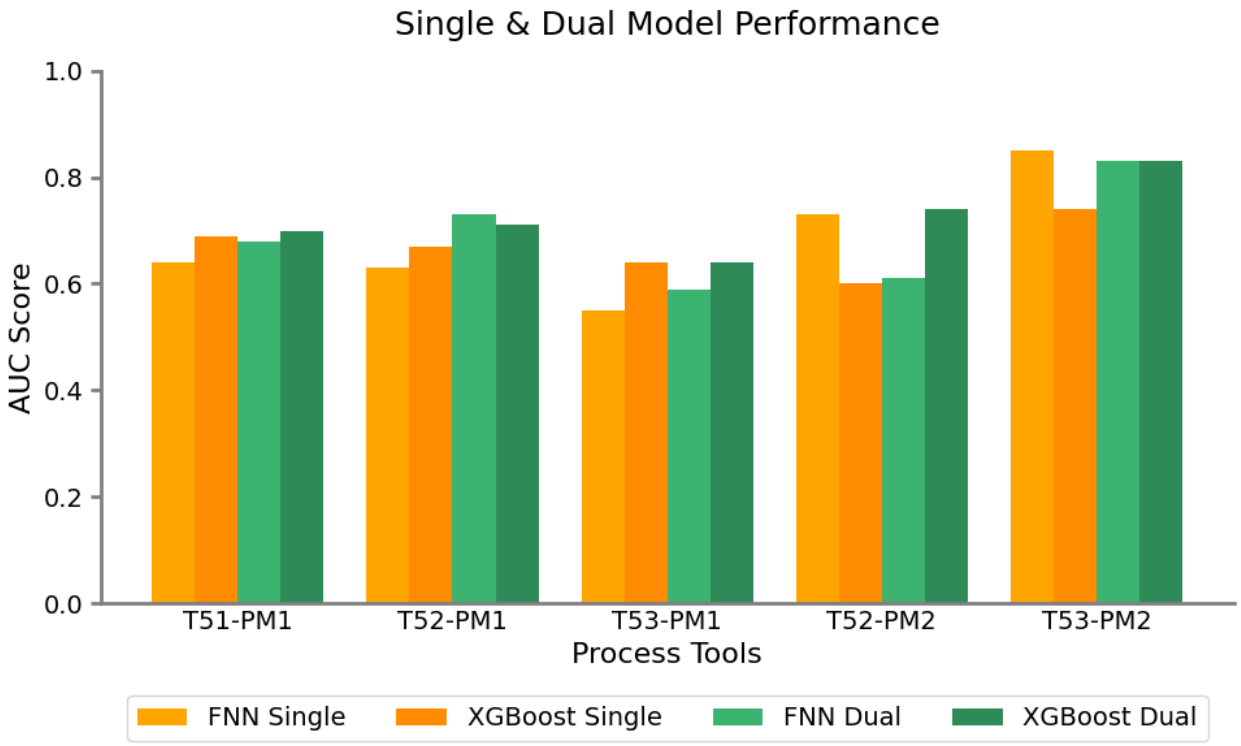


Figure 8.5: This figure shows the AUC scores for the FNN and XGBoost models applied to etching tool data using single tool and dual tool training.

Single and Dual Tool Training

The AUC scores of the FNN and XGBoost models on each of the five etching machine datasets with single-tool (trained only on one dataset) and dual-tool (trained on the aggregation of two unique datasets, the aggregated dataset is picked by best available score) training are shown in Figure 8.5:

The plotted results represent the mean values obtained from five-fold cross-validation. In this procedure, the training data are divided into five subsets, with four subsets used for training and one for validation in each fold. The resulting models from each fold are then evaluated on a separate test set, and the test performance is averaged across the five models. The standard deviation of the ROC-AUC scores across folds is approximately 0.03, indicating good model stability. An exception

is observed for tool 52-PM2, which exhibits a higher standard deviation exceeding 0.1, primarily due to its significantly smaller data volume. For single-tool training, we observed an increase in AUC scores for 3 out of the 5 tools when using the XGBoost model compared to the FNN model. The largest increase was 0.09 in T53-PM1, while the largest decrease was 0.13 in T52-PM2. In dual-tool training, XGBoost demonstrated better or similar performance compared to FNN across all tools. The largest AUC increase was 0.13 in T52-PM2 and 0.05 in T53-PM1, with the largest decrease being minimal at 0.02 in T52-PM1. An improvement in the ROC-AUC score indicates a better ability to distinguish between PASS and FAIL classifications. For instance, an AUC increase of 0.13 can suggest that under controlled 80% true positive rate (TPR) the false positive rate (FPR) has decreased at least 10% (the actual level depends on the specific behavior, the AUC score does not have a fixed relationship with TPR/FPR improvement) at various classification thresholds. In practical terms, this means that XGBoost reduces misclassifications that fewer failed cases are incorrectly predicted as pass, or more true failures are correctly identified. Conversely, the AUC drop of 0.13 in T52-PM2 indicates a decline in this ability, likely due to the dataset' s small size. To illustrate the practical implications, an AUC score of 0.8 on T53-PM2 typically enables detection of over 90% of defective products (i.e., fulfilling the <10% FPR requirement) and reduces more than half of the associated physical measurements needed for further inspection (False negative predictions). In general, every 0.1 increment in the ROC-AUC score yields approximately 10% reduction in the required physical measurements. However, these improvements are not uniformly distributed; gains realized closer to the ideal AUC of 1.0 tend to be both more significant and more difficult to achieve.

Despite some decreases in single-tool training, the reductions were negligible in dual-tool

training, and AUC scores either improved or remained stable for all datasets. This confirms that XGBoost is a competitive classifier for PASS/FAIL classification in industrial datasets. The notable fluctuations in T52-PM2 are likely due to its limited dataset size—only 863 data points compared to over 10,000 in other datasets. This small sample results in a very limited test set with just 5 fail data points, introducing bias and randomness that undermine the reliability of performance metrics for this tool. Furthermore, aggregating data across tools continues to be valuable. The dual-tool training advantage observed in prior work with FNN remains valid for XGBoost, with dual-training AUC scores consistently outperforming single-tool training across all five tools. Coupled with its superior or comparable performance, significantly lower computational resource requirements, and easier hyperparameter tuning compared to FNN, XGBoost, along with other boosted decision tree models, emerges as a highly competitive candidate for PASS/FAIL classification in industrial applications.

Transfer Learning

The transfer learning results for three different train/test ratios in single-tool training are presented in Figure 8.6. As stated in Section 8.2.3, the train/test ratios (row) indicates the proportion of data points used for model updates within a given time period, the train data length (column) indicates the frequency of model updating. The heat maps for each tool are generated by plotting AUC scores for each Test/Train Ration and Train Data Length pair. A darker shade of blue indicates a higher AUC score, meaning better model performance. The heat map indicates that in most cases, variations in the train/test ratio, ranging from 1:1 to 1:4, do not significantly affect the AUC score when the train data length remains constant. The performance differences between frequent model updates (small

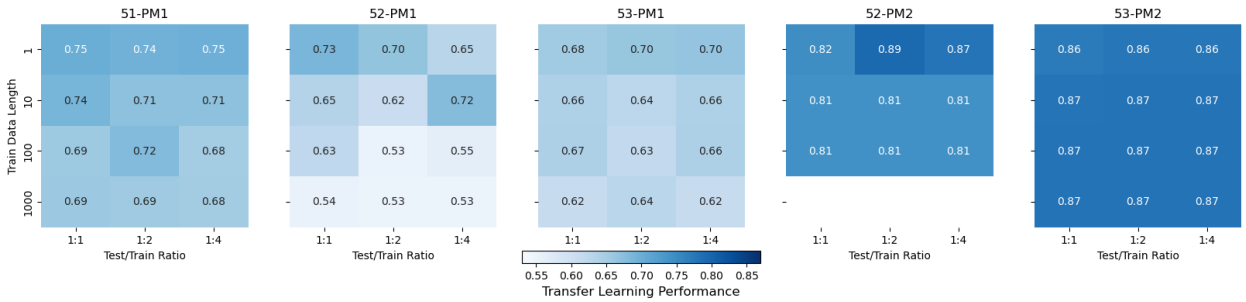


Figure 8.6: Single tool transfer learning results in heat map form. The performance difference in the same row is minimal (different train/test length ratio), but the performance difference in the same column (different train data length) is significant.

train/test ratio) and less frequent updates (large train/test ratio) are minimal. This indicates that the model does not require persistent updates to maintain its predictive accuracy. It can be trained on a subset of the data at periodic intervals and then applied over extended periods without substantial performance degradation. This finding has practical implications. In settings where labeled data collection requires physical measurements, a reduced train/test ratio translates to fewer necessary measurements in a given time period. For example, in a 1:4 train/test ratio scenario, only 20% of the available data is used for model updates within an update period. This implies that if physical measurements are required for labeling, the measurement workload can be significantly reduced, as labeling is only needed during model training. In contrast, during the prediction phase, which accounts for 80% of the time, no physical measurements are required. In addition, in semiconductor manufacturing, where data collection is automated and continuous, an important takeaway is that effective model updates do not require all collected data. Instead, only a small subset is sufficient to maintain optimal model performance within a given time period. This insight allows for optimizing data usage, focusing computational resources on periodic model retraining rather than continuously processing every newly collected data point.

However, when comparing different training data lengths, a clear trend emerges: models trained with shorter datasets, which require more frequent updates in a time-series context, consistently outperform those with longer training datasets. This suggests that frequent adaptation of the model to recent data improves performance, likely because it enables the model to better capture the dynamics of the evolving system and mitigate temporal variations in the data distribution. In contrast, when the training data length is large, the model updates occur less frequently, potentially making it less responsive to shifts in the underlying data patterns. In particular, when the training data length is set to 1000 (excluding T52-PM2, which lacks sufficient data points), the performance closely resembles that of regular single tool training, indicating that transfer learning provides little to no benefit. In conclusion, the model should be updated frequently but in a low-density manner to improve the performance under process drift with maximum savings computational resources for retraining the model.

The results of transfer learning for dual tool training are presented in Figure 8.7 in the form of a heat map. Overall, the trends observed in single-tool training are largely retained, and while dual-tool training leads to slight performance improvements in certain cases, such as for T52-PM1 and T52-PM2, the results for most other tools remain comparable to single-tool training. This suggests that although data aggregation can enhance model robustness by exposing it to a broader range of conditions, its impact under transfer learning is limited.

One key reason for this is that transfer learning inherently allows the model to dynamically adjust to shifts in the data distribution over time. This continuous adaptation reduces the effectiveness of the generalization from aggregating because the model is already evolving to capture new patterns. Additionally, while data aggregation improves generalization across different tools,

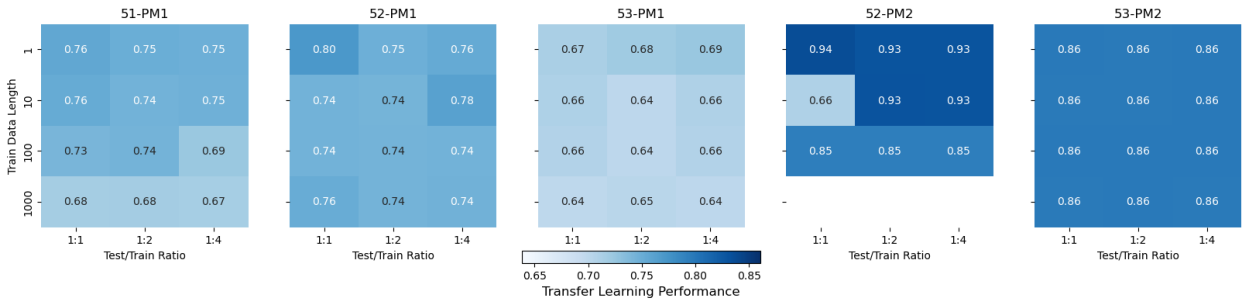


Figure 8.7: Dual tool transfer learning results in heat map form. The performance shows slight improvements compared to single-tool training and follows similar trends observed in single-tool training.

transfer learning, particularly through fine-tuning, has the opposite effect by making the model more specialized for a specific tool on the most recent dataset. As a result, the benefit of model generalization due to aggregated data may be overridden by the model' s progressive adaptation to recent data. Furthermore, transfer learning itself tends to be biased towards recent data, as it incrementally modifies the model to fit the latest available information. Systematic application of transfer learning to the model is arguably the most significant reason why data aggregation is less effective .In short,since the model is continuously optimized for the most recent dataset, the contribution of earlier aggregated data diminishes over time. Consequently, while dual-tool training offers advantages in other cases, the overall benefits of data aggregation in a transfer learning framework remains negligible.

8.3.2 Slider Tools

The FNN and XGBoost regression models for the seven slider milling tools introduced in section 8.2.3 are evaluated with two metrics: Median Absolute Error (MAE), measures the magnitude of prediction error, and median value is applied to remove the influence from outliers; and Coef-

ficient of determination score (R^2), evaluates how model explains the variance of original dataset. Minimizing MAE and maximizing R^2 are essential for improving model performance.

The Median Absolute Error (MAE) measures the average absolute difference between the predicted and actual values. It is mathematically defined as:

$$\text{MAE} = MED(|y_i - \hat{y}_i|)$$

where y_i is the true value vector and $\hat{y}_i$ is the predicted value vector, MED is the operator that calculates the median value of a vector. Minimizing median absolute error is desirable because, unlike mean absolute error, the median is less affected by large deviations. Also, MAE can be beneficial over Mean Squared Error (MSE) because it is more robust to outliers, as it measures the median deviation rather than squaring all errors, which reduces the impact of extreme values. A smaller MAE indicates that the model's predictions are closer to the actual values on average.

The R^2 score, also known as the coefficient of determination, indicates how well a model explains the variance in the target variable. It is calculated using the following formula:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

Here, $\bar{y}$ represents the mean of the actual values. The numerator is the sum of the squared errors, while the denominator is the total variance in the data. A higher R^2 score, approaching 1, suggests that the model accounts for a significant portion of the data's variance and it has a strong fit. R^2 is often used to compare the performance of different models, where a positive R^2 implies that the model performs better than a model that simply predicts the mean value. Achieving a high R^2

score alone is not sufficient, because there may still be large individual prediction errors, or the model may overfit the data. Cross-validation is critical for confirming generalization capability of the model.

Balancing MAE and R^2 ensures that the model maintains both low error magnitudes and strong variance explanation. A model with low MAE and a low R^2 may fail to capture underlying patterns in the data, while a model with high R^2 but large MAE can suffer from overfitting or prediction instability.

Single and Dual Tool Training

Both the FNN and XGBoost models are trained similarly to the etch tools: single training, where training data only comes from the tool it is being tested on, and dual training, where training data comes from the tested tool and one of the other seven slider tools. Since the data range is primarily between 1450 and 2350, the goal is to ensure predictions remain within the controlled range and do not shift to another group. Therefore, an error below 20($\sim$ 1%) is considered small enough for effective multi-class classification, and an error below 10 (approximately 0.5%) is regarded as a near-perfect score. In the slider production tool, the mean absolute error (MAE) may be loosely compared to that of an etching tool by treating predictions within the specified major ranges as positive (true) and those outside as negative (false). Since this is a direct regression model, it is not feasible to assign multiple thresholds for classification; consequently, only a single pair of TPR and FPR can be produced, permitting indirect comparison with ROC results. In general, an MAE of around 20 corresponds to an AUC of approximately 0.70–0.75, whereas an MAE below 10 indicates an AUC exceeding 0.85. In this chapter, the R^2 score is less critical than the absolute error,

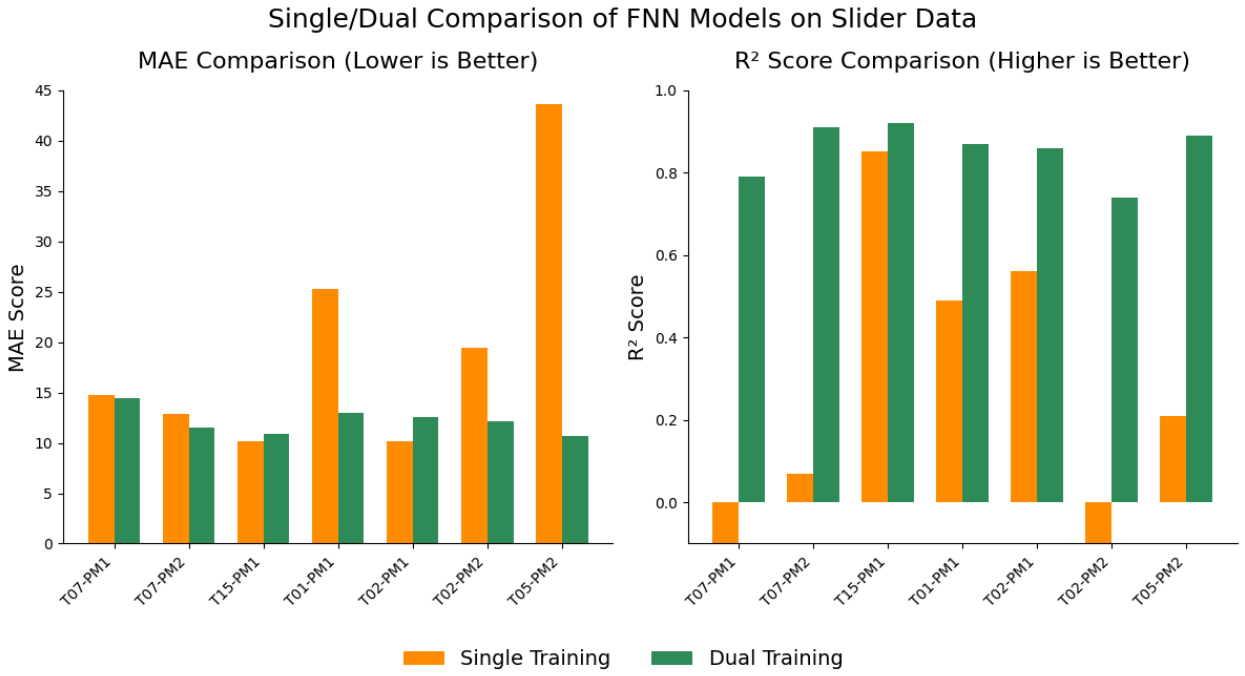


Figure 8.8: This figure shows the MAE and R^2 scores for the FNN model applied to slider tool data. MAE generally decreases when training data is aggregated with another tool. R^2 generally increase with this aggregation as well.

as the primary goal is to accurately predict the group to which each data point belongs. However, R^2 remains a useful metric for assessing how well the model captures the overall variance in the dataset, providing insight into its explanatory power. The performances of both FNN and XGBoost models using single and dual training are shown in Figure 8.8 and Figure 8.9.

The standard deviation of the mean absolute error (MAE) across five-fold cross-validation on the slider production milling process is from about 0.7 to 1.5, reflecting some non-negligible variance due to the relatively limited data volume. While this variability across folds introduces a degree of uncertainty, it remains within a range that allows for drawing reliable qualitative insights and moderately robust quantitative assessments. Importantly, the use of cross-validation averaging helps mitigate the effects of data partition randomness, enhancing the overall stability and credibility



Figure 8.9: This figure shows the MAE and R^2 scores for the XGBoost model applied to slider tool data. MAE generally decreases when training data is aggregated with another tool, but not as much as seen in the FNN model. R^2 generally increase with this aggregation as well.

of the reported results. For the FNN model, MAE decreased for 5 tools and increased for 2 tools after aggregation. The largest decrease was 32.85 ($\sim$ 2%), while the largest increase was 2.36 ($\sim$ 0.1%). In general, the decreases in MAE were more significant than the increases. R^2 scores increased for all 7 tools in the FNN model, with the largest increase being 1.37, change from a score below baseline to a score over 0.8.

For the XGB model, MAE decreased for 5 tools and increased for 2 tools. The decrease in MAE is not as large as seen in the FNN model but the increases in MAE was less than the one seen in the FNN model. The largest decrease in MAE was 3.86($\sim$ 0.2%) while the largest increase was 1.54($\sim$ 0.07%). Significant increase in R^2 scores are also seen in the XGB model. R^2 scores increased for 6 tools, the largest increase being 0.47 while the only decrease being -0.03. Although the decrease in MAE from dual-tool training is small for XGBoost model, the MAE score predicted by XGBoost is lower for 6 out of 7 tools than FNN.

The results demonstrate that aggregating training data from another tool improves the performance of both models in terms of reducing MAE and increasing R^2 scores. However, the magnitude of improvement varies between the two models, with FNN showing more pronounced changes in performance metrics. This indicates that FNN benefits significantly from data augmentation in most cases, resulting in greater prediction accuracy across tools. Furthermore, the R^2 scores for the FNN model improved for all seven tools, reflecting enhanced model fit and a better ability to explain variance in the target variable. The large increases in R^2, particularly for tools with low single-tool training performance, suggest that the dual training approach helps the FNN model capture important patterns that are not present when training on single-tool data alone. This improvement can be attributed to FNN's sensitivity to additional data, as the model's continuous learning structure

allows it to better generalize and reduce overfitting when exposed to diverse datasets.

The XGBoost model exhibited a similar trend, with MAE reductions observed for five tools and increases for two tools, with a mean change of -1.20, which is a smaller average decrease than observed in the FNN model. This smaller decrease may be due to several factors. Firstly, XGBoost's single-tool training performance was already high, leaving less room for noticeable improvement through data aggregation. Additionally, XGBoost's tree-based architecture, with its strong regularization mechanisms and robust splitting criteria, inherently minimizes overfitting, making it less sensitive to the benefits of additional synthetic or aggregated data. On the other hand, the average increase in MAE for the two tools was 0.92, lower than that observed in the FNN model, suggesting that while the gains from data augmentation were smaller, XGBoost maintained more stable and consistent performance across different tools.

In terms of R^2 scores, XGBoost showed strong overall improvements, with increases in six out of seven tools. The only tool with a decrease experienced a minor drop of -0.03, indicating that dual training generally enhances the model's ability to capture variance, although not as uniformly improving as observed in the FNN model. The improvements in R^2 suggest that data aggregation still contributes to better generalization for XGBoost, even though its architecture limits the extent of performance changes compared to FNN.

These findings highlight the importance of data aggregation in improving model performance in regression tasks, particularly when single-tool training data is insufficient to capture underlying patterns. While both models benefited from dual training, FNN exhibited greater performance gains, particularly in reducing MAE and enhancing R^2 scores. This difference can be attributed to the models' architectures: FNN models rely heavily on complex feature interactions and benefit

from additional data to reduce overfitting and improve generalization. In contrast, XGBoost's tree-based ensemble structure, with its inherent robustness to data variability and strong regularization, makes it less sensitive to the size and diversity of the training data. Overall, FNN requires more extensive data (through data aggregation) to fully extract meaningful patterns, while XGBoost maintains stable performance even with limited data, explaining the more modest benefits from data aggregation.

Linear Mixup Data Augmentation

The performance differences for the slider datasets augmented using the two previously discussed Mixup schemes—one with 100% augmentation and the other with 200%, compared to the non-augmented datasets, are presented in Figure 8.10 and Figure 8.11. The plots primarily illustrate the impact of Mixup: bars are green if the change is positive (lower MAE, higher R^2) and red if the change is negative (higher MAE, lower R^2). The first scheme creates two artificial data points between two real data points, and the second scheme creates one data point between two real data points. Training remained as before with single-tool and dual-tool training.

Because slider production datasets cluster into three major regions, the virtual metrology function needs to classify each data point correctly into their major regions and detect FAILS (act like outlier points in training data). Minimizing the median absolute error (MAE) is therefore crucial to ensure predictions remain within the correct class region. Under the first Mixup scheme for the FNN model, MAE decreased in five tools and increased slightly in two, resulting in an average change of -4.13 (largest decrease: 16.2; largest increase: 2.06). In the second scheme, MAE decreased for all but one tool (T01-PM1), with an average change of -2.73 (largest decrease: 12.7;

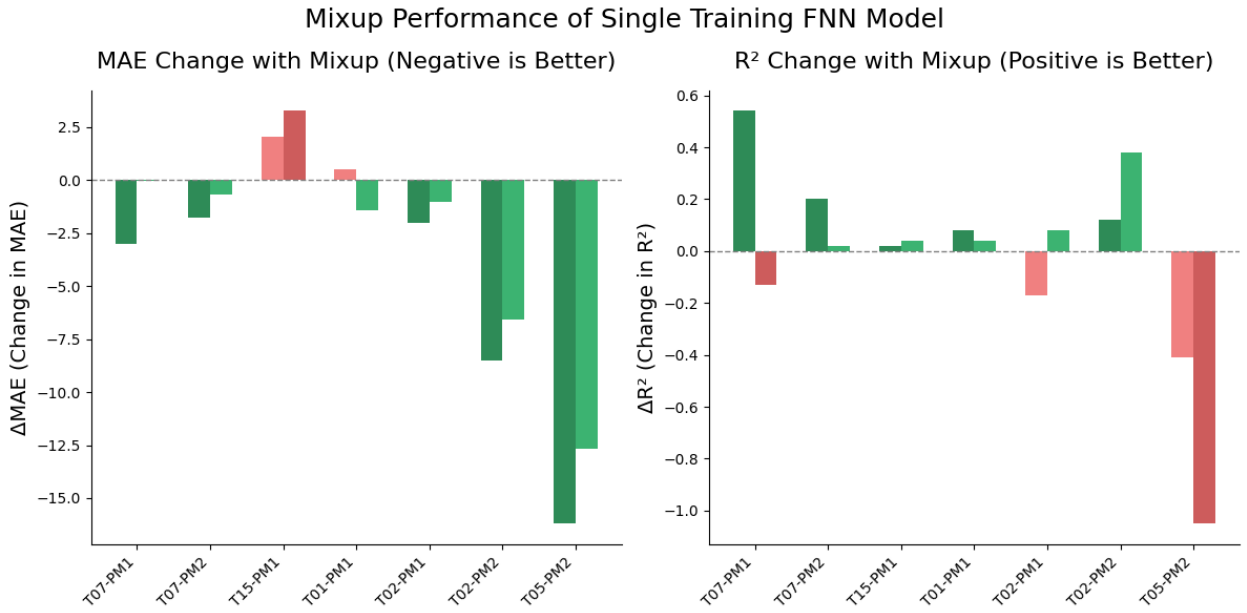


Figure 8.10: MAE and R^2 scores for the FNN model trained with data from a single slider tool using linear Mixup. In general, there is a decrease in MAE and an increase in R^2 scores across most slider tools.

largest increase: 3.27). The large decrease in MAE especially on T02-PM2 (close to -10) and T05-PM2 (around -15) can significantly improve the multi-class classification accuracy. Although R^2 is less indicative in explaining performance of multi-class classification, it is still useful for assessing the model's ability to capture overall variance. Specifically, the first scheme yielded an average R^2 increase of 0.05 (largest increase: 0.54; largest decrease: 0.41), whereas the second scheme produced an average change of -0.09. The relatively small changes in R^2 suggest that the original model may be adequately explain the variance in the target variable.

On the other hand, the greater variation in MAE suggests that the model's absolute error is still influenced by the specific conditions of each tool. A higher variation in MAE could indicate that certain tools require additional data to improve prediction stability or that some regions in the feature space are underrepresented. The observed reductions in MAE across most tools confirm

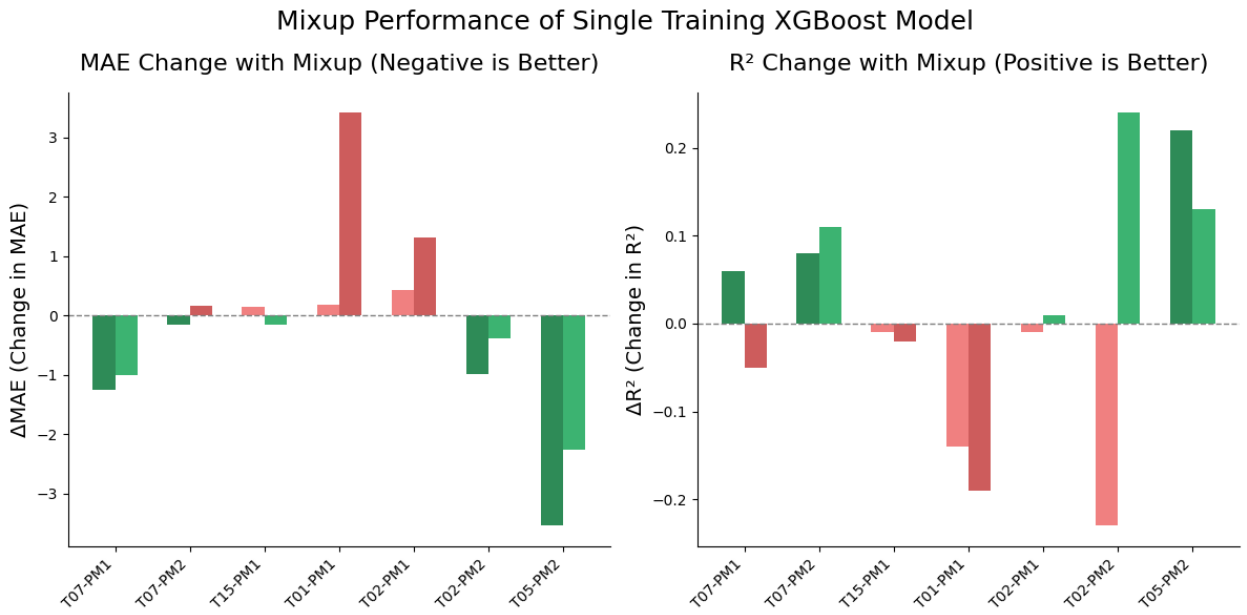


Figure 8.11: MAE and R^2 scores for the XGBoost model trained with data from a single slider tool using linear Mixup. Similar trends of decreased MAE and increased R^2 scores are observed for most tools, except for T05-PM2 where we saw a marked decrease in R^2 scores.

that the Mixup augmentation was generally beneficial, but the slight increases in MAE for a few tools highlight areas where additional adjustments, such as targeted data augmentation or feature refinement, may be needed. For the first Mixup scheme with XGBoost, MAE decreased in four tools while increasing slightly in three, yielding an average change of -0.74 (largest decrease (positive): 3.54; largest increase (negative): 0.42). The second Mixup scheme showed a similar pattern, with an average MAE change of 0.15 (largest decrease (negative): 2.27; largest increase (positive): 3.41, this scale of change in MAE won't have a significant impact on the model performance practically. In terms of R^2, results were mixed: the first scheme improved scores for three tools (average change of -0.004), while the second scheme saw gains in four tools (average change of 0.03). Despite these outcomes, XGBoost benefits less from Mixup compared to FNN, primarily because Mixup generates linearly interpolated data that complements FNNs' continuous, gradient-based learning,

helping smooth decision boundaries. By contrast, XGBoost relies on discrete tree splits, where such interpolations offer limited value for split decisions. Furthermore, XGBoost's strong built-in regularization reduces the added benefits of Mixup, while FNNs gain considerable regularization advantages, mitigating overfitting more effectively.

The results for both Mixup schemes on the FNN and XGBoost models using dual training are presented in Figure 8.12 and Figure 8.13. For both figures, each tool has two values for MAE and R^2, the darker hue being the first Mixup scheme and the second value being the second Mixup scheme. A green color indicates a positive change (like a decrease in MAE or increase in R^2) while a red color indicates a negative change (like an increase in MAE or decrease in R^2).

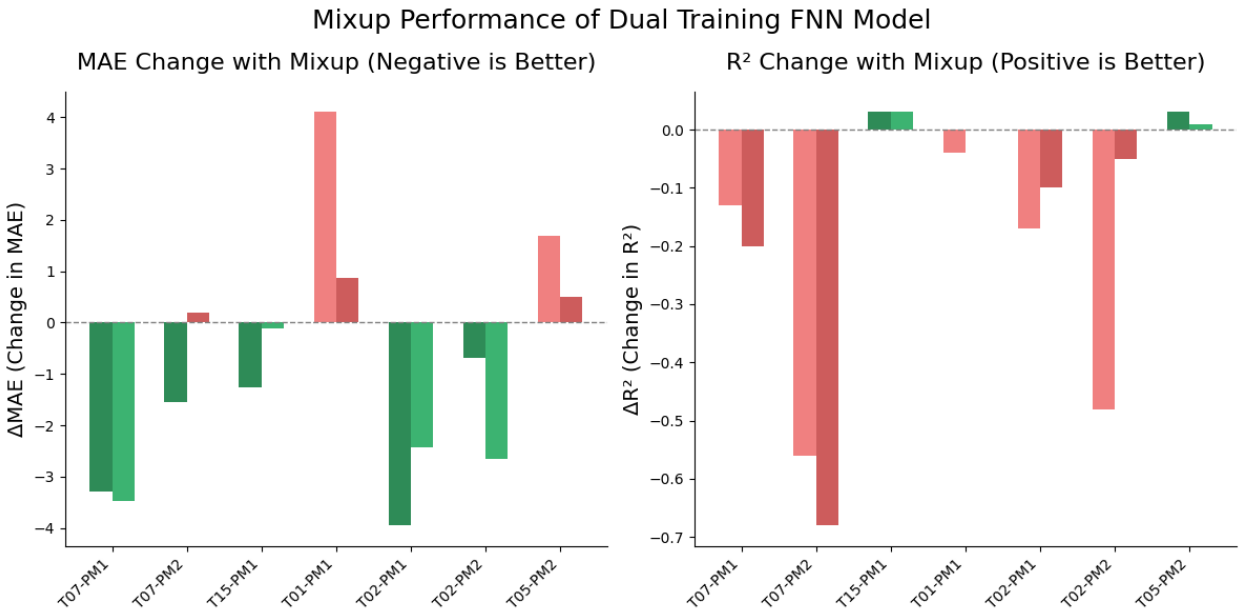


Figure 8.12: MAE and R^2 scores for the FNN model trained with data from two slider tools using linear Mixup. There is a decrease in MAE in 5 tools and an increase in 2 tools for both schemes. R^2 scores decreased in 5 tools for both schemes, with a slight increase in two tools.

In the first Mixup scheme for the FNN model, MAE decreased for five tools and increased for two, averaging a change of -0.71 (largest decrease: 3.95; largest increase: 4.1). Under the second

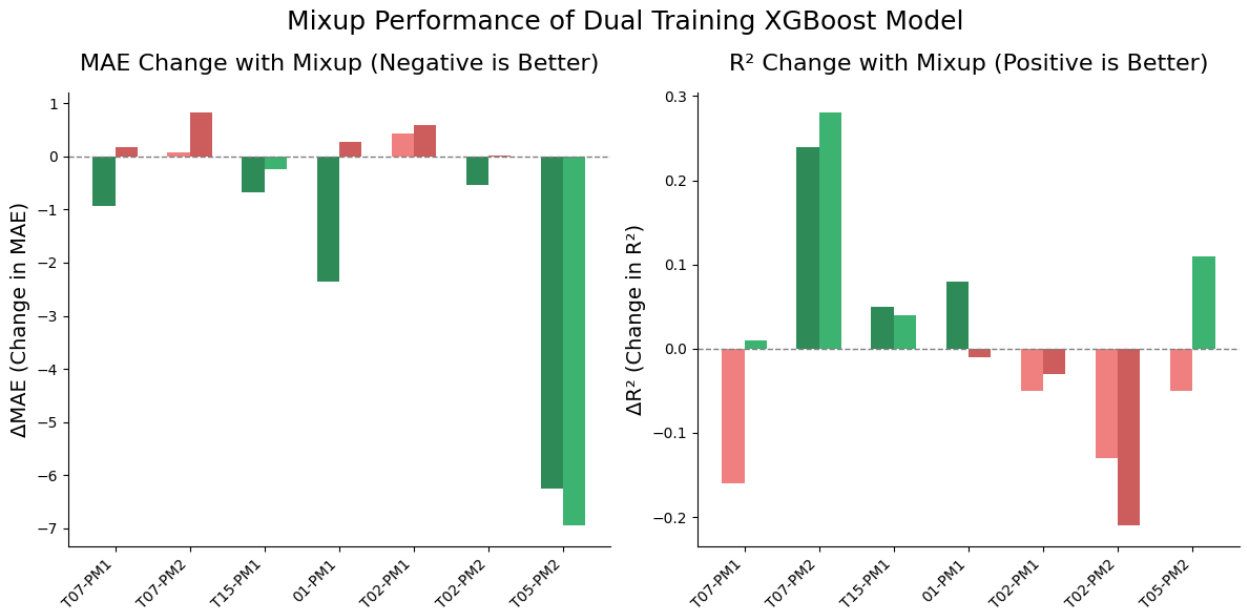


Figure 8.13: MAE and R^2 scores for the XGBoost model trained with data from two slider tools using linear Mixup. MAE generally decreased across most tools, with increases being very slight. R^2 scores saw a mixed pattern of increasing and decreasing scores.

scheme, MAE decreased in four tools and increased in three, yielding an average change of -1.01 (largest decrease: 3.47; largest increase: 0.87). Although the change in MAE is not large enough to significantly impact real-world model performance, it is still worth noting that Mixup in data aggregation has strong potential for improving model performance. Meanwhile, R^2 declined in five tools and rose slightly in two for both schemes, with average changes of -0.19 and -0.14, respectively, indicating that improvements in error reduction did not consistently translate into higher overall variance capture.

Under the first Mixup scheme with XGBoost, MAE decreased in five tools while rising slightly in two, averaging a change of -1.43 (largest drop: 6.26; largest increase: 0.44). In the second scheme, MAE fell for two tools and rose for five, though some increases were under 1%. The largest decrease was 6.95, with an average change of –0.76. The decrease of MAE on T05-PM2 is

significant enough to improve the model performance, but the changes in other tools are negligible. For R^2, the first scheme raised scores in three tools, resulting in an average change of -0.003, while the second scheme improved four tools with an average increase of 0.03. The largest R^2 gain was 0.28, and the largest decrease was 0.21.

The summary of the overall model performance change results of MAE and R^2 above is tabulated in table 8.7 and table 8.8. The first value in each cell is mean change, while the second value (blue) is median change:

Table 8.7: MAE Change

	FNN		XGB	
Scheme	Single	Dual	Single	Dual
Scheme 1	-4.13/-2	-0.71/-1.26	-0.74/-0.15	-1.43/-0.67
Scheme 2	-2.73/-1	-1.01/-0.12	0.15/-0.16	-0.76/0.18

Table 8.8: R2 Change

	FNN		XGB	
Scheme	Single	Dual	Single	Dual
Scheme 1	0.05/0.08	-0.19/-0.13	-0.004/-0.01	-0.003/-0.05
Scheme 2	-0.09/0.04	-0.14/-0.17	0.03/0.01	0.03/0.01

These results demonstrate the varying impact of linear Mixup data augmentation across different models and training schemes by demonstrating the mean and median value of change in MAE and R^2 after Mixup. While improvements are observed in both metric scores in many cases, the effectiveness of the Mixup technique appears to depend on both the model type and data aggregation status.

In the FNN model, dual training with linear Mixup produced mixed results. For the first

scheme, MAE decreased across five tools with an average reduction of 17.0%, while increases were observed in two tools, averaging 23.6%. Similarly, for the second scheme, MAE decreased by an average of 16.6% in five tools and increased by 4.4% in two tools. In terms of R^2 scores, the results were less favorable, with decreases in five tools and only slight increases in two tools under both schemes. The largest increase of R^2 score in both schemes was 0.03 while the largest decrease was -0.68. These findings suggest that while Mixup can reduce errors, it may not always lead to improved model fit, particularly when the models are trained on data from two tools.

In contrast, the XGBoost model demonstrated more stable performance with Mixup applied under dual training. For the first scheme, MAE decreased across five tools by an average of 12.8%, while the increases, observed in two tools, averaged only 2.6%. The second scheme showed a less favorable balance, with MAE decreasing in two tools (average reduction of 11.4%) and increasing in five tools, though some increases were minor (average increase of 3.8%). For R^2, the results showed mixed trends: under the first scheme, R^2 increased for three tools and decreased for four tools, whereas in the second scheme, four tools saw an increase and three tools experienced a decrease. The largest increase of R^2 score in both schemes was 0.28 while the largest decrease was -0.21.

The differences between the FNN and XGBoost models highlight the role of model architecture in how linear Mixup affects regression performance. FNN models appear to be more sensitive to both improvements and degradations in MAE and R^2, due to their reliance on complex feature relationships that can be influenced by synthetic data interpolation. XGBoost, with its tree-based architecture, shows smaller variations, due to its robust splitting criteria and regularization mechanisms.

According to Table 8.7 and Table 8.8, the impact of linear Mixup on regression performance

differs significantly between FNN and XGBoost models. For XGBoost, the mean and median differences in MAE is much smaller compared to FNN for single training, and slightly smaller than FNN for dual training; the R^2 value change is negligible. The small MAE and R^2 changes indicate that Mixup has little effect on its performance. This can be attributed to XGBoost' s tree-based architecture, which relies on robust splitting criteria and inherent regularization mechanisms that make it less sensitive to interpolated synthetic data.

In contrast, the FNN model shows more pronounced effects from Mixup. In single-tool training, Mixup leads to notable improvements in both MAE and R^2, with performance enhancing as more synthetic data points are added (Scheme 1). This could be due to Mixup acting as an effective regularizer, improving generalization by encouraging the model to learn smoother decision boundaries and reducing overfitting to noisy patterns in the original data. However, in dual-tool training, Mixup with a smaller synthetic dataset (Scheme 2) yields better results. This could be because Mixup operates directly at the data level, interpolating between samples from two potentially different data distributions. When the distributions diverge significantly, excessive synthetic data may introduce conflicting patterns, making it harder for the model to generalize effectively. Thus, a smaller Mixup dataset can preserve the distinct characteristics of each tool' s data while still providing regularization benefits to achieve lower MAE.

8.4 Conclusion

This chapter presents an analysis of the data quality and engineering requirements for machine learning-based virtual metrology for common machine tools in the semiconductor industry. The

virtual metrology application studied was the binary PASS/FAIL classification of the product quality at the end of a machine tool step. Two multi-line, multi-machine semiconductor manufacturing operations were studied: five plasma etching tools used in separate wafer production lines and seven milling tools used in separate slider production lines. The study focused on optimum processing of the data by considering the conditions, functions, and data requirements of the machines together and how to leverage commonality.

Feedforward Neural Network (FNN) and XGBoost algorithms were used and compared for both the wafer and slider production machines. For wafer production the algorithms were applied directly for PASS/FAIL classification. For slider production the algorithms were formulated as regression models to predict product thickness that stratified into three clusters for different products. Outlier identification was used to then classify PASS or FAIL. The following data processing/engineering approaches were examined:

- Data aggregation across one or more machines to increase variation and operational coverage.
- Dimensionality reduction techniques were applied to reduce noise and enhance feature extraction.
- A separate scaling approach was implemented during data aggregation to improve model performance by aligning the feature distributions of different datasets and to mitigate discrepancies caused by varying single-feature distributions despite similar feature dependencies.
- Linear Mixup algorithm was applied to augment data and address the scarcity of collected data from slider production tools.

- FNN with transfer learning with live model update was used for plasma etching to address process drift. Varying train/test and data length ratios were also tested.

For the wafer production machines, the XGBoost algorithm demonstrated better or comparable performance to FNN in both single-tool and dual-tool training while requiring fewer computational resources and offering greater robustness, making it the better choice for this solution objective. Transfer learning significantly improved model performance across all tested datasets, demonstrating the effectiveness of live updating. Notably, using only 20% of new online data for updates led to substantial performance gains, reducing the need for frequent offline physical measurements and associated costs. Transfer learning also overrode the advantages of data aggregation. In general, given the operational drift with the wafer production tools, transfer learning proved to be a fruitful endeavor.

For the slider tools, XGBoost consistently outperformed or matched FNN in regression tasks in terms of Median Absolute Error (MAE) and R2. Given better performance for both slider and wafer production, XGBoost demonstrated versatility for both classification and regression tasks. Due to its tree-based structure and inherent regularization from ensemble learning, XGBoost was less sensitive to data aggregation compared to the FNNs, though data aggregation still yielded positive effects across all datasets. Mixup strategies demonstrated notable performance improvements for the FNN models, particularly for single-tool training, due to its regularization effect which promoted better generalization. In contrast, Mixup impact on XGBoost was minimal likely because of XGBoost's tree-based architecture and robust splitting criteria. With respect to dual-tool aggregation, FNN performed better with smaller synthetic datasets, possibly due to distribution differences

between datasets that can introduce conflicting patterns when overly mixed. For XGBoost, modest MAE improvements were observed, indicating a resilience to data variations. Overall, the effectiveness of Mixup depends on model architecture and data characteristics.

While no single method universally outperforms others, the combination of Mixup, data aggregation, and transfer learning generally enhanced model performance. When systematically applied as a data engineering procedure, the combination of methods offered significant improvement to this in industrial virtual metrology PASS/FAIL application. The main usage for the virtual metrology models developed in this chapter is to automatically detect faulty product and reduce the number of metrology steps. This would greatly decrease the overall manufacturing time for each product and increase overall product throughput to meet the inevitable increase in demand for semiconductor products.

Chapter 9

Playbook for Industrial Data Processing and Modeling

General Description of Preprocessing

The preprocessing of raw datasets ensures data quality and consistency before analysis, or **AI-Ready**. Missing values must be removed or imputed, non-contributing features eliminated, and feature names aligned for consistency. Normalization, scaling, and dimensionality reduction enhance model performance and prevent overfitting. When aggregating data from multiple sources, separate normalization maintains comparability. These steps ensure clean, structured data for analysis.

Title	Description & Example
Text	texts/filename in ' '; e.g. '51-PM1.xlsx'
DateTime	texts of time; e.g. '2020-01-01T00:00:00+00:00'
Table	built-in/numpy 2D array
Container	pandas-readable data container; e.g. *.parquet, *.pkl
Model	Trained ML Model file; e.g. *.pth (torch), *.json (XGB)
DataList	List with elements og pandas DataFrame object
Scaler	Float number

Table 9.1: Custom Table with Full Document Width

Data Type Description

List of Contents

1. Convert Data into Parquet and Separate by Time
2. N/A Data Padding & Necessary Feature Extraction
3. Categorical Feature Encoder
4. Data Aggregation & Correlation Analysis
5. Normalization (Scaling) & Dimension Reduction (Single Tool)
6. Scaling & Dimension Reduction in Data Aggregation
7. FNN Model Training (Binary Classification)
8. XGBoost Model Training (Binary Classification)
9. FNN Model Training (Regression)

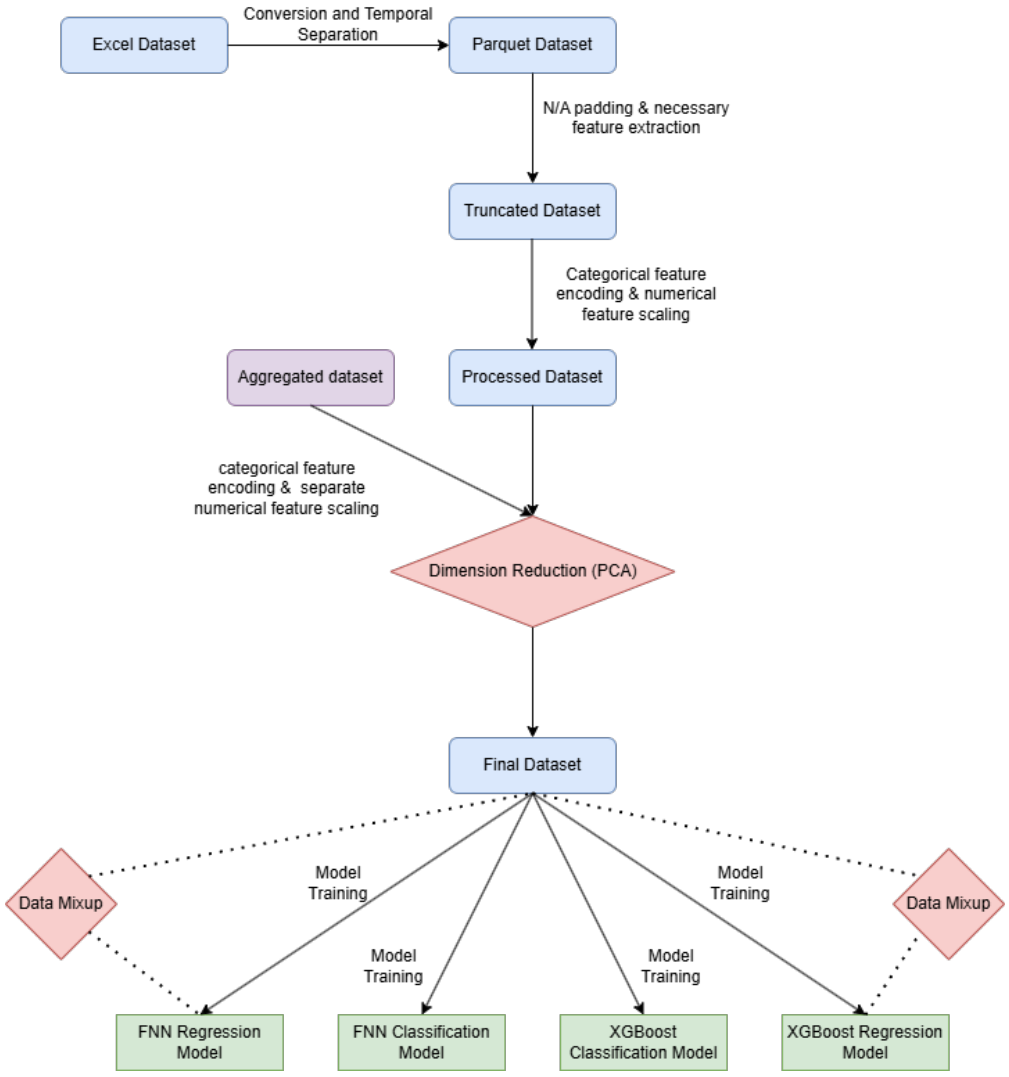


Figure 9.1: General Flow Chart

10. XGBoost Model Training (Regression)

11. Test Model Performance on Future Data

12. Advanced: LightGBM Training (Binary Classification)

13. Advanced: Physics Structure Informed Graph Attention Network Training

9.1 Convert Data into Parquet and Separate by Time

Reading large-scale CSV or Excel (.xlsx) files can be time-consuming and inefficient for repeated testing. A common practice is to convert the data into the **Parquet** format, which allows for significantly faster reading and writing using Pandas in Python. Additionally, the dataset must be split appropriately, with **training data** consisting of **past years'** data and **test data** comprising the **most recent year's** data.

9.1.1 Situation to Use

- Dataset size is large (>10000 rows or 30k total volume, or file size over 50 MB)
- Model would be applied to predict future
- **Recommend to do this step everytime**

9.1.2 Inputs & Outputs Description

- Inputs: **your_file_name** Text, **your_sheet_name** Text, **your_data_name** Text, **cut_off_date** DateTime
- Outputs: **data_o** Table, **data_n** Table, ***data.parquet** Container

9.1.3 Sample Code

```
# Run this code block if the input dataset is EXCEL file
# Import necessary packages
import numpy as np
```

```
import pandas as pd

# Input Variables
excel_name = your_file_name # *.xlsx
sheet_name = your_sheet_name
saved_data = your_data_name # *.parquet
cutoffdate = cut_off_date # e.g. '2020-01-01T00:00:00+00:00'

#Read the Excel file with the specified sheet name
df = pd.read_excel(excel_name, sheet_name=sheet_name)

# Convert the timestamp column to datetime format
df['timestamp'] = pd.to_datetime(df['timestamp'])

# Save the DataFrame as a Parquet file
df.to_parquet(saved_data, index=False)

# Read the Parquet file
data = pd.read_parquet(saved_data)

# Define the cutoff date, change by your_last_year
cutoff_date = pd.Timestamp(cutoffdate)

# Option 1: Split the dataset based on the timestamp condition
data_o = data[data['timestamp'] < cutoff_date] # Old Data Output
data_n = data[data['timestamp'] >= cutoff_date] # New Data Output

# Option 2: Split the dataset by visual observation (When time stamp format is
    different)
# data_o = data.iloc[0:44611]
# data_n = data.iloc[44611:57818]
```

```
# Use this code block if your input file is in CSV format
# Import necessary packages
import numpy as np
import pandas as pd

# Input Variables
csv_file = 'your_file_name.csv'      # CSV file (first row should be the title
   /header)
saved_data = 'your_data_name.parquet'  # Parquet file name
cutoffdate = '2020-01-01T00:00:00+00:00'  # e.g. '2020-01-01T00:00:00+00:00'

# Read the CSV file; the first row is automatically used as the header
df = pd.read_csv(csv_file, header=0)

# Convert the timestamp column to datetime format
df['timestamp'] = pd.to_datetime(df['timestamp'])

# Save the DataFrame as a Parquet file
df.to_parquet(saved_data, index=False)

# Read the Parquet file
data = pd.read_parquet(saved_data)

# Define the cutoff date
cutoff_date = pd.Timestamp(cutoffdate)

# Split the dataset based on the timestamp condition
data_o = data[data['timestamp'] < cutoff_date]  # Old Data Output
data_n = data[data['timestamp'] >= cutoff_date]  # New Data Output
```

```
# Option 2: Split the dataset by visual observation
# data_o = data.iloc[0:44611]
# data_n = data.iloc[44611:57818]
```

Usages

- Reduce dataset reading time from 30s to instant
- Provide separation between train & test set

Common Troubleshooting

- Reduce dataset reading time from 30s to instant
- Provide separation between train & test set

9.2 N/A Data Padding & Necessary Feature Extraction

N/A entries with low frequency, likely missing at random, can be filled with certain default values or the average of other valid values (may need to communicate with process engineer). Values like -999999 should also be treated as missing. After imputation, unnecessary features (e.g., timestamp, processrunid, pdatetime) should be removed. In most datasets, these nonessential features are grouped together, making it convenient to drop them by column index. In other cases, feature names can be used to specify which columns to omit or retain.

9.2.1 Situation to Use

- Dataset has missing features
- Dataset has redundant features

- **Recommend to do this step everytime**

9.2.2 Inputs & Outputs Description

- Inputs: original numerical features dataset **data_o** Table, **data_n** Table

- Outputs: cleaned numerical features dataset **data_o_num** Table, **data_n_num** Table

9.2.3 Sample Code

```
# For plasma etching processes
# Import necessary packages
import numpy as np
import pandas as pd

# -------------------------
# Option 1: Fill N/A values with a specific default value, 0 in this example
data_o.fillna(0, inplace=True)
data_n.fillna(0, inplace=True)

data_o.replace(-999999, 0, inplace=True)
data_n.replace(-999999, 0, inplace=True)

# -------------------------
# Option 2: Impute numerical columns with the column mean and drop rows with
    missing or invalid values in categorical columns

def compute_valid_mean(series):
    """Compute the mean of a column, ignoring NaNs and -999999 values."""
    valid_values = series[(series != -999999) & (~series.isna())]  # Exclude
        NaN and -999999
```

```
    return valid_values.mean() if not valid_values.empty else 0  # Avoid NaN
        mean

def process_dataframe(df, reference_means=None):
    """
    Fill NaNs and -999999 with column mean for numeric columns.
    Drop rows with NaN/-999999 in non-numeric columns.

    If `reference_means` is provided, it will be used for imputation instead
        of computing new means.
    """
    new_means = {}  # Store computed means for numeric columns

    for col in df.columns:
        if pd.api.types.is_numeric_dtype(df[col]):
            df[col] = df[col].replace(-999999, np.nan)  # Replace -999999 with
                NaN
            # Use reference means if available (for data_n to avoid leakage)
            mean_value = reference_means[col] if reference_means and col in
                reference_means else compute_valid_mean(df[col])
            df[col] = df[col].fillna(mean_value)  # Fill missing values with
                mean
            new_means[col] = mean_value  # Store the computed mean
        else:
            # For non-numeric columns, drop rows where the value is -999999 or
                NaN
            df = df[df[col] != -999999]
            df = df.dropna(subset=[col])

    return df, new_means  # Return processed dataframe and computed means
```

```
# Step 1: Process data_o (old) and store means
data_o, data_o_means = process_dataframe(data_o)

# Step 2: Process data_n (new) using means from data_o to avoid data leakage
data_n, _ = process_dataframe(data_n, reference_means=data_o_means)

# -------------------------
# Select the necessary features by column names.
# The list below contains the desired features. If the dataset does not
    contain a feature,
# it will be skipped and the overall order is maintained.
feature_names = [
    'forward_rf_power', 'gas_channel_1_flow', 'gas_channel_2_flow',
    'hi_vac_pressure', 'probe_voltage', 'reflected_rf_power',
    'rough_pressure', 'start_pressure', 'meastype_oxide_target',
    'beam_current', 'beam_voltage', 'suppressor_current',
    'suppressor_voltage', 'body_current', 'body_voltage',
    'discharge_current', 'discharge_voltage', 'filament_current',
    'filament_voltage', 'pbn_gas_flow_rate',
    'cryo_pump_first_stage_temperature', 'cryo_pump_second_stage_temperature',
    'foreline_pressure', 'shield_timer', 'actual_tilt_angle', '
       clamp_claw_timer',
    'clamp_status', 'gas_pressure', 'program_tilt_angle', 'rotation_speed',
    'shutter_position', 'stage_gas_flow_rate'
]

# For data_o, select only the available columns from the list
selected_features_o = [col for col in feature_names if col in data_o.columns]
data_o_num = data_o[selected_features_o]

# For data_n, select only the available columns from the list
```

```
selected_features_n = [col for col in feature_names if col in data_n.columns]
data_n_num = data_n[selected_features_n]
```

```
# In slider production processes
exclude_columns = ["grid_id", "process_run_id", "process_start_ts", "tool_id",
    "module_id", "process_name", "timestamp", "probe_voltage", "body_voltage"
   , "discharge_current", "discharge_voltage"]

data_o_num = data_o.drop(columns=exclude_columns + ["meastype_avg"])
data_n_num = data_n.drop(columns=exclude_columns + ["meastype_avg"])
```

Usages

- Clean the dataset to make it trainable

Common Troubleshooting

- **Key Error:** Need to change column names in 'exclude_columns' to fit each individual dataset

9.3 Categorical Feature Encoder

To properly incorporate categorical features into a machine learning model, encoding is necessary. Common methods include **Label Encoding** and **One-Hot Encoding**. One-hot encoding is suitable for categorical features with a limited number of unique values, while label encoding is preferable for high-cardinality features (e.g., substrate family, process name) to prevent large, sparse matrices that consume excessive memory and computational resources. In deep learning, embedding layers can also be used for encoding, but they are not covered here.

9.3.1 Situation to Use

- When categorical features exist (P.S. Index features should be removed not encoded)
- One-Hot Encoder: Small number of unique categories, Categories do not have a meaningful order, more suitable to Non-Tree models (e.g. Neural Network, Polynomial Regression)
- Label Encoder: Large number of unique categories, Categories have ordinal relationship, more suitable to Tree models (e.g. XGBoost, Random Forest)
- Need to do Step 1 for **all available datasets** then make a list of pandas dataframe, if new categorical features are applied, the encoder should be updated and model should be retrained with the new data.

9.3.2 Inputs & Outputs Description

- Inputs: List of dataFrames **data_list** DataList
- Outputs: Encoded substrate family **substrate_family_o** Table, **substrate_family_n** table, Encoded process name **process_name_o** Table, **process_name_n** Table, Encoded output vectors **output_data_o** Table, **output_data_n** Table

9.3.3 Sample Code

```
# Encode 'substrate_family' feature. To include all possible entries, encoder
    should be fit on all available datasets
import numpy as np
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

```
# List of Inputs
data_list = data_list

encoder = LabelEncoder()
encode_data = np.concatenate(
    [
        df['substrate_family'].astype(str).to_numpy()
        for df in data_list
    ],
    axis=0
)
encoder.fit(encode_data)
substrate_family_o = encoder.transform(np.array(data_o['substrate_family']).
    astype('str'))
substrate_family_n = encoder.transform(np.array(data_n['substrate_family']).
    astype('str'))

# Encode 'process_name' feature.
encoder2 = LabelEncoder()
encode2_data = np.concatenate(
    [df['process_name'].astype(str).to_numpy() for df in data_list],
    axis=0
)
encoder2.fit(encode2_data)
process_name_o = encoder2.transform(np.array(data_o['process_name']).astype('
    str'))
process_name_n = encoder2.transform(np.array(data_n['process_name']).astype('
    str'))

# Encode output vector
```

```
encoder3 = LabelEncoder()
output_data_o = np.expand_dims(encoder3.fit_transform(np.array(data_o['
    pass_fail']).astype('str')),1)
output_data_n = np.expand_dims(encoder3.transform(np.array(data_n['pass_fail'
    ]).astype('str')),1)
```

Usages

- Encode categorical features into numerical values to make dataset trainable

Common Troubleshooting

- **Key Error:** Need to change feature names 'substrate_family' and 'process_name' to fit dataset
- **Value Error nan:** Need to do Step 2 first

9.4 Data Aggregation & Correlation Analysis

Data aggregation is frequently applied to improve the data volume and provide more variance to data, which usually improve model performance. However, data aggregation should not be performed arbitrarily but requires careful selection. One approach involves correlation analysis, where a characteristic vector is derived for each dataset by computing correlation coefficients—biserial for binary outputs and Pearson for continuous outputs. The similarity between datasets is then assessed by computing the mean absolute error between their characteristic vectors. The dataset with the smallest difference score is selected, as greater similarity is expected to enhance model performance. The code below will output datasets number and difference score in Ascending order.

9.4.1 Inputs & Outputs Description

- Inputs: List of numerical datasets **data_list** Table, Interested Index **data_interest** Scaler, List of output vectors **output_list** Table
- Outputs: List of sorted difference score **score_list**

9.4.2 Sample Code

```
import numpy as np
from scipy.stats import pointbiserialr


def compute_characteristic_vector(X, y):
    """
    Compute the point-biserial correlation for each feature in X
    with respect to the binary output y.

    :param X: 2D data, either a DataFrame or np.array of shape (n_samples,
        n_features)
    :param y: 1D or 2D np.array of binary labels, shape (n_samples,) or (
        n_samples, 1)
    :return: 1D np.array of correlation scores for each feature (may contain
        NaN)
    """
    # If X is a DataFrame, convert to NumPy
    if hasattr(X, 'to_numpy'):
        X = X.to_numpy()  # (n_samples, n_features)
    # Flatten y if it's 2D
    if y.ndim == 2 and y.shape[1] == 1:
        y = y.ravel()
```

```
    n_features = X.shape[1]
    corr_scores = []
    for i in range(n_features):
        feature_col = X[:, i]
        # pointbiserialr returns correlation and p-value
        corr_val, _ = pointbiserialr(y, feature_col)
        corr_scores.append(corr_val)
    return np.array(corr_scores)

from scipy.stats import pearsonr

def compute_characteristic_vector_pearson(X, y):
    """
    Compute the Pearson correlation coefficient for each feature in X
    with respect to the continuous output y.

    :param X: 2D data, either a DataFrame or np.array of shape (n_samples,
       n_features)
    :param y: 1D or 2D np.array of continuous values, shape (n_samples,) or (
       n_samples, 1)
    :return: 1D np.array of correlation scores for each feature (may contain
       NaN)
    """
    # If X is a DataFrame, convert to NumPy
    if hasattr(X, 'to_numpy'):
        X = X.to_numpy()  # (n_samples, n_features)

    # Flatten y if it's 2D
    if y.ndim == 2 and y.shape[1] == 1:
        y = y.ravel()
```

```
    n_features = X.shape[1]
    corr_scores = []

    for i in range(n_features):
        feature_col = X[:, i]
        # Pearson correlation returns (correlation coefficient, p-value)
        corr_val, _ = pearsonr(y, feature_col)
        corr_scores.append(corr_val)

    return np.array(corr_scores)

def compare_datasets_by_mae(data_list, output_list, data_interest):
    """
    1) For each (X, y) in data_list/output_list, compute a characteristic
       vector
       (point-biserial correlation).
    2) Exclude features where any dataset's correlation is NaN.
    3) Calculate mean absolute error (MAE) between the dataset at `
       data_interest`
       and all others, using only valid features.
    4) Return a sorted list of (dataset_index, MAE) by ascending MAE.

    :param data_list: list of input datasets (each a DataFrame or 2D array)
    :param output_list: list of corresponding binary outputs (2D or 1D array)
    :param data_interest: int, index of the dataset to compare against all
       others
    :return: list of tuples [(dataset_index, mae), ...] sorted by ascending
       mae
    """
    # Step 1: Compute the characteristic vector for each dataset
    characteristic_vectors = []
```

```
    for X, y in zip(data_list, output_list):
        vec = compute_characteristic_vector(X, y)
        characteristic_vectors.append(vec)

    # Ensure all characteristic vectors are the same length (same # of
        features)
    # They should be if data_list all have the same # of features, but double-
        check:
    lengths = [v.shape[0] for v in characteristic_vectors]
    if len(set(lengths)) != 1:
        raise ValueError("All datasets must have the same number of features."
            )

    # Step 2: Identify valid feature-indices (no NaNs across all datasets)
    n_features = characteristic_vectors[0].shape[0]
    valid_mask = np.ones(n_features, dtype=bool)
    for vec in characteristic_vectors:
        # Update valid_mask to exclude indices where this vec is NaN
        valid_mask &= ~np.isnan(vec)

    # Check if at least one feature is valid:
    if not np.any(valid_mask):
        raise ValueError("No valid features found (all have NaN in at least
            one dataset).")

    # Filter out invalid features in each characteristic vector
    filtered_vectors = [v[valid_mask] for v in characteristic_vectors]

    # Step 3: Compute MAE using valid features only
    reference_vec = filtered_vectors[data_interest]
    score_list = []
```

```
    for i, vec in enumerate(filtered_vectors):
        if i == data_interest:
            continue
        mae = np.mean(np.abs(reference_vec - vec))
        score_list.append((i, mae))

    # Step 4: Sort by ascending MAE
    score_list.sort(key=lambda x: x[1])
    return score_list

if __name__ == "__main__":
    import pandas as pd
    np.random.seed(42)

    # ------------------------------------------
    # Example Inputs
    # ------------------------------------------

    data_list   = [data_o_num, data_2_o_num, data_3_o_num, data_4_o_num,
       data_5_o_num] # Example
    output_list = [output_data_o, output_data_2_o, output_data_3_o,
       output_data_4_o, output_data_5_o] # Example

    # Choose dataset of interest
    data_interest = 1 # Example

    # Compare and print results
    score_list = compare_datasets_by_mae(data_list, output_list, data_interest
        - 1)
    print(f"MAE compared to dataset {data_interest}:")
    for idx, mae_value in score_list:
```

```
        print(f"  Dataset {idx + 1}: MAE = {mae_value:.4f}")
```

Usages

- Give the difference score between interested dataset and other datasets
- Picking the most similar datasets usually results in 0.03-0.05 AUC score advantage

Common Troubleshooting

- Require datasets been properly cleaned in Step 2
- Make sure the feature names are aligned among datasets
- For regression task, change 'compute_characteristic_vector' in function 'compare_datasets_by_mae' to 'compute_characteristic_vector_pearson'

9.5 Normalization (Scaling) & Dimension Reduction (Single Tool)

Scaling is essential in most machine learning workflows to enhance model performance. Additionally, industrial datasets often have high-dimensional features, many of which contribute minimally to the output while adding noise. To address this, dimensionality reduction is necessary to extract the most relevant features, improving efficiency and robustness in high-dimensional spaces.

9.5.1 Situation to Use

Scaling:

- Features have different units and ranges
- Distance based model applied (e.g. PCA dimension reduction)

- Gradient descent based model applied (e.g. FNN, SVM)
- For tree-based method, scaling is not a must but recommended

Dimension Reduction (PCA):

- High dimension data (>30 features)
- Noisy data (most manufacture data has high noise)

9.5.2 Inputs & Outputs Description

- Inputs: cleaned numerical dataset **data_o_num** Table, **data_n_num** Table; substrate family vector **substrate_family_o** Table, **result_n** Table; process name vector **process_name_o** Table, **result2_n** Table; **variance_retain** Scaler
- Outputs: Complete input data matrix (Single Training) **input_data_pca** Table

9.5.3 Sample Code

```
from sklearn.preprocessing import StandardScaler
# List of Inputs
v = variance_retain


scaler = StandardScaler()
scaler.fit(data_o_num)
data_o_num_scaled = scaler.transform(data_o_num)


from sklearn.decomposition import PCA
pca = PCA(n_components=0.999)
data_o_num_scaled_pca = pca.fit_transform(data_o_num_scaled)
```

```
input_data_pca = np.concatenate((data_o_num_scaled_pca,np.expand_dims(
    substrate_family_o,1),np.expand_dims(process_name_o,1)),1) # The
    categorical features should not be normalized
```

Usages

- Scaling deal with the gradient vanashing and explosion problem
- No normalization usually results in random guessing level performance
- PCA usually improve AUC score of 0.03-0.05 (FNN), 0.01-0.03 (Tree)

Common Troubleshooting

- **Value Error Mismatched Dimension:** Check 'result_o' and 'result2_o' dimension (Need to be 1-D vector)
- **Value Error nan:** Need to do Step 2 first

9.6 Scaling & Dimension Reduction in Data Aggregation

Due to differences in the internal distribution of datasets, directly aggregating raw datasets and subsequently applying a global scaling or normalization can negatively impact model performance. To mitigate this issue, it is essential to apply separate scaling techniques, ensuring that each individual dataset is normalized independently before concatenation. By normalizing each dataset separately, the mean of all features is set to zero and the standard deviation to one, then the distributional disparities among datasets are effectively minimized, leading to improved model robustness and performance.

9.6.1 Inputs & Outputs Description

Input Variables:

- Inputs: 1^{st} cleaned numerical dataset **data_o_num** Table, 2^{nd} cleaned numerical dataset **data_2_o_num** Table, 1^{st} categorical features **substrate_family_o** Table, **process_name_o** Table, 2^{nd} categorical features **substrate_family_2_o** Table, **process_name_2_o** Table
- Outputs: aggregated input data matrix: **input_data_pca** Table

Output Variables:

- Inputs: 1^{st} output vector **output_data_o** Table, **output_data_2_o** Table
- Outputs: Aggregated output vector **y_data_scaled** (Slider Production Process) Table, **output_data** (Plasma Etching Process) Table

9.6.2 Sample Code

```
# Plasma Etching Tools, binary output
import numpy as np
from sklearn.preprocessing import StandardScaler

# Need to make sure the data dimensions of aggregated datasets are the same
if data_o_num.shape[1] != data_2_o_num.shape[1]:
    print('Data dimension not matched!')

scaler_1 = StandardScaler()
scaler_2 = StandardScaler()
scaler_1.fit(data_o_num) # Interested Dataset
```

```
scaler_2.fit(data_2_o_num) # Aggregated Dataset
input_data_left_scaled = np.concatenate((scaler_1.transform(data_o_num),
    scaler_2.transform(data_2_o_num)),0) # Concatenate scaled datasets
input_data_right_1 = np.concatenate((substrate_family_o,substrate_family_2_o)
    ,0) # Concatenate substrate family column
input_data_right_2 = np.concatenate((process_name_o,process_name_2_o),0) #
    Concatenate process name column

from sklearn.decomposition import PCA

# Initialize PCA and specify the explained variance ratio
pca = PCA(n_components=0.999)
input_data_left_scaled_pca = pca.fit_transform(input_data_left_scaled)

input_data_pca = np.concatenate((input_data_left_scaled_pca,np.expand_dims(
    input_data_right_1,1),np.expand_dims(input_data_right_2,1)),1)

output_data = np.concatenate((output_data_o,output_data_2_o),0) # Change Here
```

```
# Slider Production Tools, numerical output
import numpy as np
from sklearn.preprocessing import StandardScaler

scaler.fit(output_data_o.reshape(-1,1)) # scaler only fit on output vector of
    interested dataset
y_data_scaled = scaler.transform(np.concatenate((output_data_o,output_data_2_o
    ),0).reshape(-1, 1)).flatten() # Concatenate multiple datasets then
    normalize
```

Usages

- Separate Scaling usually results in at least 0.05 AUC score improvement

Common Troubleshooting

- **Value Error input array dimension:** Make sure multiple datasets are pre-cleaned to have same column numbers

9.7 FNN Model Training (Binary Classification)

The training process of the Feedforward Neural Network (FNN) model involves several key steps to ensure optimal performance and reproducibility. First, the dataset is partitioned into training and testing subsets. A custom weighted cross-entropy loss function is implemented to address class imbalance effectively (around 1-2.5% fail rate). The other methods to deal with data imbalance problem like oversampling methods are thoroughly tested and proved to be not improving the performance, as well as the nonsupervised and semi-supervised method. The model architecture is then constructed, with Xavier (Glorot) random initialization applied to the network weights to facilitate stable gradient flow. To ensure reproducibility, a manually selected random seed is set. The model is transferred to a CUDA-enabled GPU, leveraging accelerated computation for faster training. Finally, the model is trained, and the best-performing version determined by the lowest validation loss is selected for deployment.

9.7.1 Situation to Use

- Large amounts of available data

- Complex nonlinear pattern in data
- Have Cuda-Ready GPU for accelerated training (Optional)
- Do NOT have high interpretability requirements

9.7.2 Inputs & Outputs Description

- Inputs: final input data matrix **input_data_pca** Table, final output data matrix **output_data** Table, neural network hyperparameters Scaler
- Outputs: The trained Model **Optimized.pth** Model

```
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.model_selection import train_test_split
# Train Test Split to make train/validation datasets
input_data_train, input_data_test, output_data_train, output_data_test =\
train_test_split(input_data_pca, output_data, test_size=0.2, shuffle=True,
    random_state=10, stratify=output_data)

input_tensor = torch.tensor(input_data_train).float()
output_tensor = torch.tensor(output_data_train).float()

input_tensor_test = torch.tensor(input_data_test).float()
output_tensor_test = torch.tensor(output_data_test).float()

# Weights to deal with imbalanced datasets
w_0 = len(output_data)/(len(output_data)-np.sum(output_data))
```

```
w_1 = len(output_data)/(np.sum(output_data))

# Custom weighted cross entropy loss
class CustomLoss(nn.Module):
    def __init__(self):
        super(CustomLoss, self).__init__()

    def forward(self, predictions, targets):
        # Binary cross-entropy loss
        bce_loss = nn.BCELoss(reduction = 'none')(predictions, targets)

        # Calculate the modified loss
        modified_loss = torch.where(targets == 0, w_0 * bce_loss, bce_loss)
        modified_loss_2 = torch.where(targets == 1, w_1 * modified_loss,
            modified_loss)

        # Return the modified loss
        return torch.mean(modified_loss_2)

# Neural Network Building in Torch, adjust the layer number and neurons
class NeuralNetwork(nn.Module):
    def __init__(self):
        super(NeuralNetwork, self).__init__()
        self.fc1 = nn.Linear(input_tensor.shape[1], 32)  # 11 input features,
            32 neurons in the first hidden layer
        self.fc2 = nn.Linear(32, 32)  # 32 neurons in the second hidden layer
        self.fc3 = nn.Linear(32, 1)   # 1 output neuron

        self.sigmoid = nn.Sigmoid()
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout(0.1)
```

```
    def forward(self, x):
        x = self.sigmoid(self.fc1(x))
        #x = self.dropout(x)
        x = self.sigmoid(self.fc2(x))
        #x = self.dropout(x)
        x = self.fc3(x)
        x = self.sigmoid(x)
        return x

# Random initialize the model parameters, define loss function, optimizer and
    learning rate
torch.manual_seed(10)
model = NeuralNetwork()

def weights_init(m):
    if isinstance(m, nn.Linear):
        torch.nn.init.xavier_uniform_(m.weight)
        if m.bias is not None:
            torch.nn.init.constant_(m.bias, 0)

model.apply(weights_init)

loss_function = CustomLoss()

# Define the optimizer
optimizer = optim.Adam(model.parameters(), lr=0.001)
model = model.to('cuda')

# Define epochs and train the model on cuda-ready graphic card
num_epochs = 3000
```

```
min_loss_test = 100


input_tensor,output_tensor = input_tensor.to('cuda'), output_tensor.to('cuda')
input_tensor_test,output_tensor_test = input_tensor_test.to('cuda'),
    output_tensor_test.to('cuda')


for epoch in range(num_epochs):
    optimizer.zero_grad()  # Zero the gradients
    outputs = model(input_tensor)  # Forward pass
    loss = loss_function(outputs, output_tensor)  # Calculate loss
    loss_test = loss_function(model(input_tensor_test),output_tensor_test)
    loss.backward()  # Backward pass
    optimizer.step()  # Update weights

    if loss_test < min_loss_test:
        min_loss_test = loss_test
        torch.save(model,'optimized.pth')

    if (epoch + 1) % 10 == 0:
        print(f"Epoch [{epoch+1}/{num_epochs}], Train Loss: {loss.item()},
            Test Loss: {loss_test.item()}")
```

Usages

- Weighted loss function, sigmoid activation function and early stop helps raise ROC-AUC score to at least 0.7

Common Troubleshooting

- **CUDA Error:** Check if you have cuda-ready graphic card & install cuda version Torch
- **Module Error:** Check if properly install Torch package
- **Value Error input dimension:** Check if the input matrix and output vector have the same row numbers

9.8 XGBoost Model Training (Binary Classification)

The XGBoost model training process does not incorporate early stopping or the selection of the best model based on validation performance; instead, it relies solely on a predefined set of hyperparameters. Additionally, the implementation is more straightforward, as XGBoost provides built-in functionality within its Python module, eliminating the need for extensive custom training logic.

9.8.1 Situations to Use

- Tabular data with missing values and mixed data types (usually true in semiconductor industry)
- Have relatively small datasets (less than required by neural networks)
- Requires Interpretability to an extent

9.8.2 Inputs & Outputs Description

- Inputs: final input data matrix **input_data_pca** Table, final output data matrix **output_data** Table, XGBoost hyperparameters Scaler

- Outputs: The trained Model object **xgmodel** Model

```
import numpy as np
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score
class_ratio = (output_data == 0).sum() / (output_data == 1).sum()
xg_model = xgb.XGBClassifier(scale_pos_weight=class_ratio, use_label_encoder=
    False, eval_metric="auc")
xg_model.fit(input_data_train, output_data_train)

# Predict probabilities on the training set and calculate train ROC AUC score
y_train_pred_proba = xg_model.predict_proba(input_data_train)[:, 1]
train_roc_auc = roc_auc_score(output_data_train, y_train_pred_proba)
print("Train ROC AUC Score:", train_roc_auc)

# Predict probabilities on the test set and calculate test ROC AUC score
y_test_pred_proba = xg_model.predict_proba(input_data_test)[:, 1]
test_roc_auc = roc_auc_score(output_data_test, y_test_pred_proba)
print("Test ROC AUC Score:", test_roc_auc)
```

Usages

- XGBoost with auc evaluation metric and class weights can improve the model performance to at least over 0.7

Common Troubleshooting

- **Value Error input dimension:** Check if the input matrix and output vector have the same row numbers

9.8.3 Tuning Instruction

- **n_estimators:** Larger number can improve performance but may bring overfitting, default value 100 is enough for most cases
- **max_depth:** Increase number of splits before determination can improve performance but may bring overfitting, normally 10 is a reasonable number after PCA.
- **gamma:** Higher gamma means higher performance gain requirement when further splitting a node, which makes the model conservative but not easy to overfit. XGBoost is not prone to overfit as FNN, so set to default value 0 is acceptable under most conditions
- **reg_alpha, reg_lambda:** First & Second order regularizer, default value (0,1) usually works
- **subsample:** Subsample chooses part of the datasets to train, when dataset is large enough (over 10k rows), a subsample of 0.8 can be applied to improve generalize performance, when dataset is small, set subsample to be 1.0
- **scale_pos_weight:** In imbalance dataset, need to specify the class weights to XGB model
- **objective:** Loss function applied to train the model, for regression task, 'reg:squarederror' for regression task and 'binary:logistic' for binary classification (which is default when calling XGBClassifer module).
- **eval_metric:** Score applied to evaluate on the validation set, if not specified, the default metric is the objective function.

9.9 Data Mixup

Data Mixup is a linear interpolation technique used for data augmentation, providing data-side regularization to mitigate overfitting and enhance model performance, particularly for small datasets. This method has shown notable improvements in Feedforward Neural Networks (FNNs) and moderate improvements in XGBoost. In the semiconductor industry, it is crucial to apply Mixup only to data points from the same product in regression tasks; otherwise, mixing data from different products may generate non-existent product variations, leading to unrealistic and suboptimal results. Mixup is a very flexible method and is not limited to linear interpolation of one value between two adjacent data points.

9.9.1 Situation to Use

- Dataset is small (< 5000 data points)
- Works well with neural networks based models
- Can be applied on part of the data to conduct oversampling or undersampling to mitigate imbalance

9.9.2 Inputs & Outputs Description

- Inputs: cleaned input data matrix **input_data**, cleaned output data matrix **output_data**
- Outputs: final input data matrix **input_data_pca**, final output data matrix **output_data**

```
import numpy as np
from sklearn.preprocessing import StandardScaler
# Only major regions data is mixed up, fail data points are not
mask_1 = np.array(output_data >= 1450) & np.array(output_data <= 1650)
mask_2 = np.array(output_data >= 1650) & np.array(output_data <= 1850)
mask_3 = np.array(output_data >= 2150) & np.array(output_data <= 2350)

scaler_1 = StandardScaler()
scaler_2 = StandardScaler()
scaler = StandardScaler()

# If 1450-1650 range is not major range, comment out this block
mixup_inputs_1 = input_data[mask_1][0:-1]*0.5 + input_data[mask_1][1:]*0.5
mixup_outputs_1 = output_data[mask_1][0:-1]*0.5 + output_arrays[num][mask_1
    ][1:]*0.5

# If 1650-1850 range is not major range, comment out this block
mixup_inputs_2 = input_data[mask_2][0:-1]*0.5 + input_data[mask_2][1:]*0.5
mixup_outputs_2 = output_data[mask_2][0:-1]*0.5 + output_data[mask_2][1:]*0.5

# If 2150-2350 range is not major range, comment out this block
mixup_inputs_3 = input_data[mask_3][0:-1]*0.5 + input_data[mask_3][1:]*0.5
mixup_outputs_3 = output_data[mask_3][0:-1]*0.5 + output_data[mask_3][1:]*0.5

# concatenate mixup datasets from multiple ranges
mixup_inputs = np.concatenate((mixup_inputs_1,mixup_inputs_2,mixup_inputs_3)
    ,0)
mixup_outputs = np.concatenate((mixup_outputs_1,mixup_outputs_2,
    mixup_outputs_3),0)
```

```
# concatenate mixup datasets with original datasets
X_data_scaled = scaler_1.fit_transform(np.concatenate((input_data,mixup_inputs
    ),0))
output_data = scaler.fit_transform(np.concatenate((output_data,mixup_outputs)
    ,0).reshape(-1, 1)).flatten()

from sklearn.decomposition import PCA

# Initialize PCA and specify the explained variance ratio
pca = PCA(n_components=0.999)
input_data_pca = pca.fit_transform(X_data_scaled)
```

```
# More Mixup data points Scheme
# If 1450-1650 range is not major range, comment out this block
mixup_inputs_1 = np.concatenate((input_data[mask_1][0:-1]*0.33 + input_data[
    mask_1][1:]*0.67,\
                                 input_data[mask_1][0:-1]*0.67 + input_data[
                                     mask_1][1:]*0.33),0)
mixup_outputs_1 = np.concatenate((output_data[mask_1][0:-1]*0.33 + output_data
    [mask_1][1:]*0.67,\
                                 output_data[mask_1][0:-1]*0.67 + output_data[
                                     mask_1][1:]*0.33),0)

# If 1650-1850 range is not major range, comment out this block
mixup_inputs_2 = np.concatenate((input_data[mask_2][0:-1]*0.33 + input_data[
    mask_2][1:]*0.67,\
                                 input_data[mask_2][0:-1]*0.67 + input_data[
                                     mask_2][1:]*0.33),0)
mixup_outputs_2 = np.concatenate((output_data[mask_2][0:-1]*0.33 + output_data
    [mask_2][1:]*0.67,\
                                 output_data[mask_2][0:-1]*0.67 + output_data[
```

```
                                    mask_2][1:]*0.33),0)

# If 2150-2350 range is not major range, comment out this block
mixup_inputs_3 = np.concatenate((input_data[mask_3][0:-1]*0.33 + input_data[
    mask_3][1:]*0.67,\
                                 input_data[mask_3][0:-1]*0.67 + input_data[
                                    mask_3][1:]*0.33),0)
mixup_outputs_3 = np.concatenate((output_data[mask_3][0:-1]*0.33 + output_data
    [mask_3][1:]*0.67,\
                                 output_data[mask_3][0:-1]*0.67 + output_data[
                                    mask_3][1:]*0.33),0)
```

```
# Dual Dataset Mixup Guide
# ......
X_data_scaled_1 = np.concatenate((input_data,mixup_inputs),0)
output_data_mixup_1 = np.concatenate((output_data,mixup_outputs),0)
# Execute each dataset through here, then feed the input matrix and output
    vector to Step 6
```

Usages

- Augment small dataset to have over 15% decrease in regression MAE on neural network model, and 5–10% decrease on decision tree based model
- In data aggregation, reference to the third code block

Common Troubleshooting

- Check sum(mask_1), sum(mask_2), sum(mask_3) to know the major regions in each dataset to appropriately comment out the blocks
- If only one major region, the concatenate function won't work; replace with specific 'mixup_inputs_x' and 'mixup_outputs_x'

9.10 FNN Model Training (Regression)

The difference between regression and the binary classification system was mainly in activation function and final layer configuration. The binary classification task prefers sigmoid as activation function, but regression task usually applied ReLU. The classification task has sigmoid (binary) or softmax (multi-class) function after final layer to convert predicted value into probabilities sums up to 1, but regression task doesn't have it.

- Inputs: final input data matrix **input_data_pca**, final output data matrix **output_data**, neural network hyperparameters
- Outputs: The trained Model **Optimized.pth**

```
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.model_selection import train_test_split
# Neural network model
class RegressionModel(nn.Module):
    def __init__(self, input_size):
        super(RegressionModel, self).__init__()
        self.hidden1 = nn.Linear(input_size, 8)
        self.hidden2 = nn.Linear(8, 8)
        self.output = nn.Linear(8, 1)
        self.dropout = nn.Dropout(0.5)

        # Xavier initialization
        torch.manual_seed(42)
```

```
        nn.init.xavier_uniform_(self.hidden1.weight)
        nn.init.xavier_uniform_(self.hidden2.weight)
        nn.init.xavier_uniform_(self.output.weight)
        nn.init.zeros_(self.hidden1.bias)
        nn.init.zeros_(self.hidden2.bias)
        nn.init.zeros_(self.output.bias)

    def forward(self, x):
        x = torch.relu(self.hidden1(x))
        #x = self.dropout(x)
        x = torch.relu(self.hidden2(x))
        return self.output(x)

# Split dataset
X_train, X_test, y_train, y_test = train_test_split(input_data_pca,
    output_data, test_size=0.2, random_state=10)

# Convert to PyTorch tensors
X_train_tensor = torch.tensor(X_train, dtype=torch.float32).cuda()
y_train_tensor = torch.tensor(y_train, dtype=torch.float32).unsqueeze(1).cuda
    ()
X_test_tensor = torch.tensor(X_test, dtype=torch.float32).cuda()
y_test_tensor = torch.tensor(y_test, dtype=torch.float32).unsqueeze(1).cuda()

# Model, loss function, optimizer
input_size = X_train.shape[1]
model = RegressionModel(input_size).cuda()
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=0.01)

# Training loop
```

```
epochs = 5000
best_model = None
lowest_test_loss = float("inf")

for epoch in range(epochs):
    # Training step
    model.train()
    optimizer.zero_grad()
    y_train_pred = model(X_train_tensor)
    train_loss = criterion(y_train_pred, y_train_tensor)
    train_loss.backward()
    optimizer.step()

    # Validation step
    model.eval()
    with torch.no_grad():
        y_test_pred = model(X_test_tensor)
        test_loss = criterion(y_test_pred, y_test_tensor)

    # Save the best model
    if test_loss.item() < lowest_test_loss:
        lowest_test_loss = test_loss.item()
        best_model = model.state_dict()

    if (epoch + 1) % 100 == 0:
        print(f"Epoch {epoch+1}/{epochs}, Train Loss: {train_loss.item():.4f},
            Test Loss: {test_loss.item():.4f}")

# Load the best model
model.load_state_dict(best_model)
torch.save(model,'optimized.pth')
```

Usages

- FNN with ReLU activation function and MSE loss is applied to achieve reasonable performance

Common Troubleshooting

- Check the FNN binary classification section troubleshooting

9.11 XGBoost Model Training (Regression)

The difference between regression and classification task is mainly in loss function, the regression usually use squared error as loss function. XGBoost is inherently simpler to tune than FNN.

- Inputs: final input data matrix **input_data_pca**, final output data matrix **output_data**, XGBoost hyperparameters
- Outputs: The trained Model object **xgmodel**

```
import xgboost as xgb
import numpy as np
import joblib
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Assume input_data_pca and output_data are NumPy arrays
# Replace this with actual data loading if needed
# input_data_pca = np.load("input_data_pca.npy")
# output_data = np.load("output_data.npy")

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(input_data_pca,
    output_data, test_size=0.2, random_state=42)
```

```
# Define XGBoost hyperparameters (change these as needed)
xgb_params = {
    "n_estimators": 100,         # Number of boosting rounds
    "learning_rate": 0.05,       # Step size shrinkage to prevent overfitting
    "max_depth": 6,              # Maximum depth of trees
    "min_child_weight": 1,       # Minimum sum of instance weight in child
       nodes
    "subsample": 0.8,            # Fraction of samples used for training each
       tree
    "colsample_bytree": 0.8,     # Fraction of features used for training each
       tree
    "objective": "reg:squarederror",  # Squared error for regression
    "random_state": 42
}

# Initialize and train the XGBoost model
xgmodel = xgb.XGBRegressor(**xgb_params)
xgmodel.fit(X_train, y_train)

# Evaluate the model on test data
y_pred = xgmodel.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
print(f"Test MSE: {mse:.4f}")

# Save the trained model
joblib.dump(xgmodel, "xgmodel.pkl")
print("Trained XGBoost model saved as 'xgmodel.pkl'.")
```

Usages

- Regression task with tree method requires

more hyperparameter tuning to have usually better than FNN performance

- Check the binary classification XGBoost troubleshooting

Common Troubleshooting

9.12 Test Model Performance on Future Data (Binary Classification)

Future data should be scaled using the scaler fitted on the training/validation data to maintain consistency in feature distributions. Similarly, dimensionality reduction should be applied using the PCA transformation fitted on the training/validation data to ensure that the test data follows the same transformation space. In other words, none of the preprocessing steps (scaling, PCA, or any other transformation) should be fitted on the test data, as this would introduce data leakage and compromise the model's generalization ability.

- Inputs: future data matrix **data_n_num**, future data categorical features vectors **substrate_family_n**, **process_name_n**, future output data **output_data_n**, corresponding scaler **scaler (single)**, **scaler_1 (aggregation)**, trained model **model_used** Model
- Outputs: Resulting ROC-AUC Score for future data

```
import numpy as np
import torch
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve, roc_auc_score
# Neural Network
# Future Categorical Feature
```

```
model_used = torch.load('optimized.pth')
result_1 = substrate_family_n
result2_1 = process_name_n

input_1 = data_n_num
input_1 = scaler_1.transform(input_1)
input_data_1 = np.concatenate((input_1,np.expand_dims(result_1,1),np.
    expand_dims(result2_1,1)),1)
input_data_1 = pca.transform(input_data_1)
output_data_1 = output_data_n
input_tensor_1 = torch.tensor(input_data_1).float()
output_tensor_1 = torch.tensor(output_data_1).float()

fpr_o,tpr_o,thresholds = roc_curve(output_data_1.squeeze(1),model_used(
    input_tensor_1.to('cuda')).detach().to('cpu').numpy().squeeze(1))
plt.figure()
plt.plot(fpr_o,tpr_o)
#plt.plot(fpr[1:],thresholds[1:])
print(roc_auc_score(output_data_1.squeeze(1),model_used(input_tensor_1.to('
    cuda')).detach().to('cpu').numpy().squeeze(1)))
plt.xlabel('FPR')
plt.ylabel('TPR')
```

```
import numpy as np
from sklearn.metrics import roc_auc_score
# XGBoost Model Evaluation
result_1 = substrate_family_n
result2_1 = process_name_n

input_1 = data_n_num
input_1 = scaler.transform(input_1)
```

```
input_data_1 = np.concatenate((input_1,np.expand_dims(result_1,1),np.
    expand_dims(result2_1,1)),1)
#input_data_1 = pca.transform(input_data_1)
output_data_1 = output_data_n #Change here
y_pred_proba_n = xg_model.predict_proba(input_data_1)[:, 1]


roc_auc = roc_auc_score(output_data_1, y_pred_proba_n)
print("ROC AUC Score:", roc_auc)
```

Usages

- Test the model performance on future set with original scaler and PCA transformation to prevent data leaking

Common Troubleshooting

- Check if the pre-cleaning is applied on the new data as well
- Check if the scaler applied is the same as the scaler fitted on interested dataset
- Check if the CUDA environment is properly setted

9.13 Advanced: LightGBM Training (Binary Classification)

9.13.1 General Description

This tutorial explains how to train a **single-tool binary classifier** with **LightGBM** when your dataset already contains processed numeric features and a few categorical columns.

The goal is to keep the workflow simple for users who do not have deep machine learning experience:

1. split older rows and future rows by time,

2. keep categorical columns in their native form,

3. let LightGBM handle the category splits internally,

4. train with early stopping,

5. evaluate on validation and future test data.

This public tutorial intentionally uses **generic feature names** such as `num_a`, `num_b`, `cat_1`, and `feat_a`. It does **not** expose any confidential internal signal names.

Plain-language idea

LightGBM is a tree-based boosting model. For many tabular manufacturing problems, it is easier to start with than a neural network. It usually needs less preprocessing, does not usually need numeric scaling, and can use categorical columns directly when they are marked correctly.

9.13.2 Legend

Symbol / Name	Meaning
N	Number of rows
p	Number of predictor columns
X	Predictor table
y	Binary label, where 1 means fail and 0 means pass
`TIME_COL`	Time-based split column, for example `fiscalweek`
`LABEL_COL`	Label column
`CAT_COLS`	List of categorical feature names

Symbol / Name	Meaning
NUM_COLS	List of numeric feature names
train_df	Training subset from older data
val_df	Validation subset from older data
test_df	Future test subset
best_iteration	Best boosting round found by early stopping
scale_pos_weight	Class imbalance weight for positive class

9.13.3 Why LightGBM for a First Baseline?

Situation to Use

- You have a processed single-tool table.
- The label is binary, such as pass/fail.
- The dataset contains a mix of numeric and categorical columns.
- You want a strong, practical baseline before trying more complex neural models.

Inputs and Outputs

- **Inputs:** one processed parquet file, one label column, one time column.
- **Outputs:** one trained LightGBM model and future-set performance metrics.

Why beginners often like it

- Fast to train.
- Usually strong on tabular data.
- Handles missing values well.
- Usually does not need feature scaling.
- Can use categorical columns directly without one-hot encoding.

9.13.4 Recommended Public Data Layout

A simple public layout is shown below.

Suggested columns

- **Time column:** `fiscalweek`
- **Label column:** `label`, where `1 = fail`, `0 = pass`
- **Categorical columns:** for example `cat_1`, `cat_2`, `cat_3`
- **Numeric columns:** for example `num_a`, `num_b`, `num_c`
- **Engineered columns:** for example `feat_a`, `feat_b`, `duration_sec`

Important note about scaling

For LightGBM, you usually **do not need standardization or min-max scaling**. Tree models split by thresholds, so they are much less sensitive to the numeric scale than neural networks.

9.13.5 Split Data by Time

Situation to Use

- The model will be used on future production.
- You want a realistic evaluation.

Sample Code

Listing 9.1: Time-based split for a single-tool dataset

```
import pandas as pd
from sklearn.model_selection import train_test_split


PARQUET_PATH = "processed_single_tool.parquet"
TIME_COL = "fiscalweek"
LABEL_COL = "label"
TEST_CUTOFF = 202552


# Load already-processed data
data = pd.read_parquet(PARQUET_PATH)


# Future test set
future_mask = data[TIME_COL].astype("int32") >= TEST_CUTOFF
pool_df = data.loc[~future_mask].copy()
test_df = data.loc[future_mask].copy()


# Train/validation split inside the old-data pool
train_df, val_df = train_test_split(
```

```
    pool_df,
    test_size=0.2,
    random_state=42,
    stratify=pool_df[LABEL_COL] if pool_df[LABEL_COL].nunique() > 1 else None,
)


print("train:", train_df.shape)
print("val:  ", val_df.shape)
print("test: ", test_df.shape)
```

Common troubleshooting

- If the test set is empty, check the cutoff value.
- If one subset has only one class, your cutoff may be too aggressive.
- Never mix future rows back into training just to improve the score.

9.13.6 Identify Numeric and Categorical Columns

Key idea

LightGBM can use categorical columns internally, but you must mark them clearly and keep category handling consistent across train, validation, and test.

Sample Code

Listing 9.2: Choose feature columns

```
DROP_COLS = [TIME_COL, LABEL_COL]
CAT_COLS = ["cat_1", "cat_2", "cat_3"]
```

```
# Keep all remaining non-categorical predictors as numeric features
ALL_FEATURE_COLS = [c for c in data.columns if c not in DROP_COLS]
NUM_COLS = [c for c in ALL_FEATURE_COLS if c not in CAT_COLS]
FEATURE_COLS = NUM_COLS + CAT_COLS

print("numeric columns:", NUM_COLS)
print("categorical columns:", CAT_COLS)
```

Beginner rule

Do not one-hot encode these columns when you want to use LightGBM's built-in categorical logic. Keep them as true categories.

9.13.7 Make Category Handling Consistent

Why this step matters

The same category must mean the same thing in train, validation, and test. A simple safe rule is:

- Learn allowed categories from **training only**,
- Apply the same category type to validation and test,
- Unseen future categories become missing values.

Sample Code

Listing 9.3: Train-only category vocabulary

```
from pandas.api.types import CategoricalDtype
```

```
for c in CAT_COLS:
    # convert to string first so missing handling is explicit
    train_values = train_df[c].astype("string").fillna("__UNK__")
    train_categories = sorted(train_values.unique().tolist())
    cat_type = CategoricalDtype(categories=train_categories, ordered=False)

    train_df[c] = train_values.astype(cat_type)
    val_df[c] = val_df[c].astype("string").fillna("__UNK__").astype(cat_type)
    test_df[c] = test_df[c].astype("string").fillna("__UNK__").astype(cat_type)
```

What happens to unseen future categories?

If a category appears only in future data and not in the training set, it becomes missing under the training-defined category type. That is usually safer than learning new category values from the future set.

9.13.8 Train LightGBM with Built-In Categorical Support

Core idea

Use `lightgbm.Dataset` and tell the model which columns are categorical. No one-hot encoding is required.

Sample Code

Listing 9.4: Training a binary LightGBM model

```
import lightgbm as lgb
import numpy as np
```

```
X_train = train_df[FEATURE_COLS]
y_train = train_df[LABEL_COL].astype(int)

X_val = val_df[FEATURE_COLS]
y_val = val_df[LABEL_COL].astype(int)

n_pos = max(1, int((y_train == 1).sum()))
n_neg = max(1, int((y_train == 0).sum()))
scale_pos_weight = n_neg / n_pos

train_set = lgb.Dataset(
    X_train,
    label=y_train,
    categorical_feature=CAT_COLS,
    free_raw_data=False,
)

val_set = lgb.Dataset(
    X_val,
    label=y_val,
    reference=train_set,
    categorical_feature=CAT_COLS,
    free_raw_data=False,
)

params = {
```

```
    "objective": "binary",
    "metric": "auc",
    "boosting_type": "gbdt",
    "learning_rate": 0.03,
    "num_leaves": 63,
    "min_data_in_leaf": 80,
    "feature_fraction": 0.85,
    "bagging_fraction": 0.85,
    "bagging_freq": 1,
    "lambda_l2": 1.0,
    "scale_pos_weight": scale_pos_weight,
    "verbosity": -1,
    "seed": 42,
}

model = lgb.train(
    params,
    train_set,
    num_boost_round=3000,
    valid_sets=[train_set, val_set],
    valid_names=["train", "val"],
    callbacks=[
        lgb.early_stopping(stopping_rounds=100),
        lgb.log_evaluation(period=50),
    ],
)
```

```
print("best_iteration:", model.best_iteration)
```

What this does well

- Uses native categorical splits.
- Handles class imbalance with `scale_pos_weight`.
- Stops training automatically when validation AUC stops improving.

9.13.9 Predict and Evaluate

Sample Code

Listing 9.5: Basic evaluation on validation and future test

```
from sklearn.metrics import roc_auc_score
import numpy as np


X_test = test_df[FEATURE_COLS]
y_test = test_df[LABEL_COL].astype(int)


p_val = model.predict(X_val, num_iteration=model.best_iteration)
p_test = model.predict(X_test, num_iteration=model.best_iteration)


val_auc = roc_auc_score(y_val, p_val) if y_val.nunique() > 1 else float("nan")
test_auc = roc_auc_score(y_test, p_test) if y_test.nunique() > 1 else float("nan"
    )


print("VAL AUC :", round(val_auc, 6))
print("TEST AUC:", round(test_auc, 6))
```

Optional code: best TPR under an FPR cap

Listing 9.6: Useful metric for imbalanced manufacturing screening

```
def max_tpr_under_fpr_cap(y_true, y_prob, fpr_cap=0.20):
    y_true = np.asarray(y_true).astype(int)
    y_prob = np.asarray(y_prob).astype(float)

    order = np.argsort(-y_prob)
    y = y_true[order]

    P = int((y == 1).sum())
    N = int((y == 0).sum())
    if P == 0 or N == 0:
        return float("nan"), float("nan"), float("nan")

    tp = np.cumsum(y == 1)
    fp = np.cumsum(y == 0)

    tpr = np.concatenate([[0.0], tp / P])
    fpr = np.concatenate([[0.0], fp / N])
    thr = np.concatenate([[np.inf], y_prob[order]])

    mask = fpr <= fpr_cap
    if not np.any(mask):
        return float("nan"), float("nan"), float("nan")

    idx = np.where(mask)[0]
    best_k = idx[np.argmax(tpr[mask])]
```

```
    return float(tpr[best_k]), float(fpr[best_k]), float(thr[best_k])


for cap in [0.05, 0.10, 0.20]:
    tpr, fpr, thr = max_tpr_under_fpr_cap(y_test, p_test, fpr_cap=cap)
    print(f"TEST TPR@FPR<={cap:.2f}: {tpr:.4f} | FPR={fpr:.4f} | thr={thr:.6g}")
```

9.13.10 LightGBM versus XGBoost

Short version

Both LightGBM and XGBoost are strong gradient-boosted tree libraries. For this tutorial, the main practical difference is that LightGBM is often a very smooth starting point for tabular manufacturing work when you want fast training and simple native categorical handling.

Topic	LightGBM	XGBoost
Training style	Leaf-wise growth by default. Often fast and strong, but can overfit if the tree is too flexible.	Commonly level-wise style in many setups. Often a bit more conservative by default.
Categorical data	Native categorical handling is straightforward when columns are marked as categorical.	Also supports native categorical data in current versions, but some categorical behavior depends on supported tree methods and related parameters.
Preprocessing burden	Usually very low for tabular data. Often no scaling and no one-hot encoding needed.	Also low in many cases, but users often still bring older one-hot workflows from earlier practice.

Topic	LightGBM	XGBoost
When beginners compare them	LightGBM is often tried first for speed and convenience.	XGBoost is often used as a second strong baseline to confirm robustness.
Practical advice	Start simple and regularize if overfitting appears.	Use as a comparison baseline, especially if you already have an XGBoost pipeline.

A useful beginner mindset

If LightGBM already gives a good future AUC and stable low-FPR screening performance, you may not need a more complex model immediately. If performance is unstable, compare against XGBoost and a simple dense neural baseline before making the pipeline more complicated.

9.13.11 Hyperparameter Tuning Instructions

Suggested tuning order

Tune the model in this order so the process stays understandable:

1. get a working baseline,
2. tune tree complexity,
3. tune subsampling and regularization,
4. tune learning rate and boosting rounds,
5. revisit imbalance handling if recall is too low.

A practical tuning table

Parameter	What it controls	Beginner guidance
`learning_rate`	Step size per tree	Start around 0.03 to 0.10. Lower values are safer but need more boosting rounds.
`num_leaves`	Tree flexibility	One of the most important controls. Bigger values fit more detail but overfit more easily. Try 31, 63, 127.
`min_data_in_leaf`	Minimum rows in a leaf	Increase this if validation performance is unstable or overfitting appears. Good first values: 50, 80, 120, 200.
`max_depth`	Maximum depth	Optional extra safety limit. Use a small positive value if trees get too deep.
`feature_fraction`	Column subsampling	Try 0.7 to 0.95. Helps regularization and sometimes speed.
`bagging_fraction`	Row subsampling	Try 0.7 to 0.95. Usually set together with `bagging_freq`.
`bagging_freq`	How often row subsampling is used	Usually 1 is simple and works well.
`lambda_l1`, `lambda_l2`	Weight regularization	Increase if the model is too jumpy or overfits strongly.

Parameter	What it controls	Beginner guidance
`min_gain_to_split`	Split gain threshold	Raise this when the model creates too many weak splits.
`scale_pos_weight`	Positive-class weighting	Helpful when fail rows are much rarer than pass rows. Start with `n_neg / n_pos`.
`cat_smooth, cat_l2`	Categorical regularization	Useful when categorical features have many rare levels or noisy small groups.

Simple tuning playbook

Listing 9.7: A very simple manual tuning loop

```
trial_grid = [
    {"num_leaves": 31,  "min_data_in_leaf": 120},
    {"num_leaves": 63,  "min_data_in_leaf": 80},
    {"num_leaves": 127, "min_data_in_leaf": 50},
]

results = []

for trial in trial_grid:
    params_trial = params.copy()
    params_trial.update(trial)

    model_trial = lgb.train(
```

```
        params_trial,
        train_set,
        num_boost_round=3000,
        valid_sets=[val_set],
        valid_names=["val"],
        callbacks=[lgb.early_stopping(100), lgb.log_evaluation(0)],
    )

    p_val_trial = model_trial.predict(X_val, num_iteration=model_trial.
    best_iteration)
    auc_trial = roc_auc_score(y_val, p_val_trial)

    results.append({
        "trial": trial,
        "best_iteration": model_trial.best_iteration,
        "val_auc": float(auc_trial),
    })

for row in results:
    print(row)
```

Interpretation tips

- If train AUC is much higher than validation AUC, reduce complexity.
- If both train and validation AUC are low, slightly increase complexity or improve features.
- If recall at low FPR is poor, review class weighting and decision thresholding.

9.13.12 Common Data Leakage Mistakes

Leakage checklist

- **Do not** split randomly across time if the production goal is future prediction.
- **Do not** learn category vocabularies from the full dataset before splitting.
- **Do not** compute thresholds on the test set and then report them as if they were fixed in advance.
- **Do not** include features that are only known after the prediction point.
- **Do not** tune hyperparameters by repeatedly checking the test set.

Good practice

Use validation for model selection and threshold selection. Use the future test set once at the end for final reporting.

9.13.13 Complete Minimal Example

Listing 9.8: Compact end-to-end example

```
import pandas as pd
import numpy as np
import lightgbm as lgb
from pandas.api.types import CategoricalDtype
from sklearn.model_selection import train_test_split
from sklearn.metrics import roc_auc_score


PARQUET_PATH = "processed_single_tool.parquet"
```

```
TIME_COL = "fiscalweek"
LABEL_COL = "label"
CAT_COLS = ["cat_1", "cat_2", "cat_3"]
TEST_CUTOFF = 202552


# Load
full_df = pd.read_parquet(PARQUET_PATH)


# Split by time
pool_df = full_df[full_df[TIME_COL].astype("int32") < TEST_CUTOFF].copy()
test_df = full_df[full_df[TIME_COL].astype("int32") >= TEST_CUTOFF].copy()
train_df, val_df = train_test_split(
    pool_df,
    test_size=0.2,
    random_state=42,
    stratify=pool_df[LABEL_COL] if pool_df[LABEL_COL].nunique() > 1 else None,
)


# Build feature list
DROP_COLS = [TIME_COL, LABEL_COL]
FEATURE_COLS = [c for c in full_df.columns if c not in DROP_COLS]


# Apply train-only categorical vocabulary
for c in CAT_COLS:
    train_values = train_df[c].astype("string").fillna("__UNK__")
    cat_type = CategoricalDtype(sorted(train_values.unique().tolist()), ordered=
   False)
```

```
    train_df[c] = train_values.astype(cat_type)
    val_df[c] = val_df[c].astype("string").fillna("__UNK__").astype(cat_type)
    test_df[c] = test_df[c].astype("string").fillna("__UNK__").astype(cat_type)

X_train = train_df[FEATURE_COLS]
y_train = train_df[LABEL_COL].astype(int)
X_val = val_df[FEATURE_COLS]
y_val = val_df[LABEL_COL].astype(int)
X_test = test_df[FEATURE_COLS]
y_test = test_df[LABEL_COL].astype(int)

n_pos = max(1, int((y_train == 1).sum()))
n_neg = max(1, int((y_train == 0).sum()))

train_set = lgb.Dataset(X_train, label=y_train, categorical_feature=CAT_COLS)
val_set = lgb.Dataset(X_val, label=y_val, reference=train_set,
    categorical_feature=CAT_COLS)

params = {
    "objective": "binary",
    "metric": "auc",
    "boosting_type": "gbdt",
    "learning_rate": 0.05,
    "num_leaves": 63,
    "min_data_in_leaf": 80,
    "feature_fraction": 0.85,
    "bagging_fraction": 0.85,
```

```
    "bagging_freq": 1,
    "lambda_l2": 1.0,
    "scale_pos_weight": n_neg / n_pos,
    "verbosity": -1,
    "seed": 42,
}

model = lgb.train(
    params,
    train_set,
    num_boost_round=3000,
    valid_sets=[val_set],
    valid_names=["val"],
    callbacks=[lgb.early_stopping(100), lgb.log_evaluation(50)],
)

p_val = model.predict(X_val, num_iteration=model.best_iteration)
p_test = model.predict(X_test, num_iteration=model.best_iteration)

print("VAL AUC :", roc_auc_score(y_val, p_val) if y_val.nunique() > 1 else float(
    "nan"))
print("TEST AUC:", roc_auc_score(y_test, p_test) if y_test.nunique() > 1 else
    float("nan"))
```

9.13.14 Final Notes

Practical recommendation

For a single-tool binary problem, start with this LightGBM tutorial as your first strong baseline. If the result is already stable on future data, keep the workflow simple. Only move to more complex models if you have a clear reason.

Official documentation notes

This tutorial follows the current public documentation for LightGBM and XGBoost categorical support. In particular:

- LightGBM can use categorical features directly and can auto-detect pandas unordered categorical columns.
- XGBoost also supports native categorical data in current versions, but categorical behavior depends on compatible tree methods and related parameters.

9.14 Advanced: Physics Structure Informed Graph Attention Network Training

9.14.1 General Description

This tutorial explains how to train a **Graph Attention Network (GAT)** for binary classification when each production record contains:

- several **grouped numeric signals**,

- a few **categorical context variables**, and
- additional engineered scalar features such as `feat_...` variables.

The public example in this document intentionally uses **generic node names** such as `group_1`, `group_2`, and generic feature names such as `feat_a`, `feat_b`. This keeps confidential signal identities out of open publications while still preserving the full training logic.

The workflow is designed to be beginner-friendly:

1. split old data and future data by time,
2. fit preprocessing on the training side only,
3. build a small graph for each row,
4. train a GAT model,
5. evaluate on validation and future test data.

Plain-language idea

A GAT learns from a graph, which means the model can use both *node content* and *node relationships*. In this tutorial, each row becomes a small graph: grouped process summaries become graph nodes, categorical context becomes context nodes, and extra engineered features stay in a small dense branch outside the graph.

9.14.2 Legend

Symbol / Name	Meaning
B	Batch size
G	Number of grouped process nodes
C	Number of categorical context variables
F_g	Feature dimension for each grouped node
F_c	Embedding dimension for each categorical node
E	Number of auxiliary scalar features
X_g	Grouped node feature tensor with shape (B, G, F_g)
X_c	Context-node tensor with shape (B, C, F_c)
x_{aux}	Auxiliary feature tensor with shape (B, E)
A	Graph adjacency mask
z	Graph-level latent vector
$\hat{y}$	Predicted probability of fail
`group_1, group_2,` …	Public placeholder names for confidential process groups
`feat_a, feat_b, …`	Public placeholder names for engineered scalar features
`category_1, category_2, …`	Public placeholder names for categorical context variables

9.14.3 Recommended Public Data Layout

A simple public layout for the tutorial is shown below.

Suggested columns

- **Time column:** `fiscalweek` or another time index.
- **Label column:** `label`, where `1 = fail`, `0 = pass`.
- **Grouped node columns:** a few summary statistics per public group, for example:
  - `group_1_avg`, `group_1_std`, `group_1_cv`
  - `group_2_avg`, `group_2_std`, `group_2_cv`
- **Categorical columns:** for example `category_1`, `category_2`, `category_3`.
- **Auxiliary columns:** all columns beginning with `feat_`, plus any approved global scalar columns such as cycle time or duration.

Why keep grouped nodes and auxiliary features separate?

Grouped node statistics describe *local process behavior* and belong inside the graph. Auxiliary `feat_...` variables usually describe *cross-group summaries* or high-level engineered indicators and can be concatenated later at the dense head.

A commonly useful public stability metric is the coefficient of variation:

$$\mathrm{cv} = \frac{\mathrm{std}}{\max(|\mathrm{avg}|, \varepsilon)}.$$

9.14.4 Split Data by Time

Situation to Use

- The model will be used on future production.

- You want a realistic estimate of future performance.
- Your table already contains processed numeric columns.

Inputs and Outputs

- **Inputs:** processed table, time column, label column, test cutoff.
- **Outputs:** train/validation pool, future test set.

Sample Code

Listing 9.9: Time-based split

```
import pandas as pd
from sklearn.model_selection import train_test_split


TIME_COL = "fiscalweek"
LABEL_COL = "label"
TEST_CUTOFF = 202552


# data is already processed and saved as parquet
# one row = one production record


data = pd.read_parquet("processed_data.parquet")


pool = data[data[TIME_COL] < TEST_CUTOFF].copy()
test = data[data[TIME_COL] >= TEST_CUTOFF].copy()


train_df, val_df = train_test_split(
```

```
    pool,
    test_size=0.2,
    random_state=42,
    stratify=pool[LABEL_COL] if pool[LABEL_COL].nunique() > 1 else None,
)


print("train:", train_df.shape)
print("val:  ", val_df.shape)
print("test: ", test.shape)
```

Usages

- Mimics the real deployment setting: train on the past, test on the future.
- Avoids accidentally letting future rows influence model fitting.

Common Troubleshooting

- If the test set is empty, check the cutoff value.
- If one split has only one class, adjust the cutoff or collect more rows.

9.14.5 Prepare Public Grouped Features and Auxiliary Features

Situation to Use

- Your real internal signal names are confidential.
- You still want a public tutorial that explains the exact modeling pattern.

Inputs and Outputs

- **Inputs:** processed table with grouped columns, categorical columns, and `feat_...` columns.
- **Outputs:** generic group map and auxiliary feature list.

Sample Code

Listing 9.10: Public feature map

```
GROUP_FEATURE_MAP = {
    "group_1": ["group_1_avg", "group_1_std", "group_1_cv"],
    "group_2": ["group_2_avg", "group_2_std", "group_2_cv"],
    "group_3": ["group_3_avg", "group_3_std", "group_3_cv"],
    "group_4": ["group_4_avg", "group_4_std", "group_4_cv"],
}

CAT_COLS = ["category_1", "category_2", "category_3"]

# public rule: every engineered scalar starting with feat_ is auxiliary
AUX_COLS = sorted([c for c in data.columns if c.startswith("feat_")])

# optional extra approved scalar columns
for extra_col in ["duration_sec", "processed_time"]:
    if extra_col in data.columns:
        AUX_COLS.append(extra_col)

AUX_COLS = sorted(set(AUX_COLS))

print("groups:", list(GROUP_FEATURE_MAP.keys()))
```

```
print("categoricals:", CAT_COLS)

print("auxiliary features:", AUX_COLS)
```

Usages

- Keeps the publication clean and non-confidential.
- Makes it easy for new users to map their own grouped features into the same structure.

Common Troubleshooting

- Every column named in `GROUP_FEATURE_MAP` must exist in the table.
- If one auxiliary feature is all missing, remove it or inspect the upstream processing job.

9.14.6 Fit Preprocessing on the Training Split Only

Situation to Use

- Group statistics and auxiliary features have different scales.
- Categorical variables must be turned into integer IDs.
- You want to avoid leakage.

Inputs and Outputs

- **Inputs:** train/validation/test data frames.
- **Outputs:** train-only scalers and category vocabularies.

Sample Code

Listing 9.11: Train-only preprocessing

```
import numpy as np
from sklearn.preprocessing import StandardScaler


# flatten grouped columns into one list for scaling
GROUP_COLS = [
    col
    for cols in GROUP_FEATURE_MAP.values()
    for col in cols
]


# fit scalers on train only
node_scaler = StandardScaler().fit(train_df[GROUP_COLS])
aux_scaler = StandardScaler().fit(train_df[AUX_COLS])


# fit category vocabularies on train only
cat_vocab = {}
for col in CAT_COLS:
    vals = train_df[col].astype(str).fillna("__UNK__")
    uniq = ["__UNK__"] + sorted(v for v in vals.unique() if v != "__UNK__")
    cat_vocab[col] = {v: i for i, v in enumerate(uniq)}


# transform helper


def encode_category(series, mapping):
    raw = series.astype(str).fillna("__UNK__")
    return raw.map(lambda x: mapping.get(x, 0)).to_numpy(dtype=np.int64)
```

Usages

- Scaling helps gradient-based models train more stably.
- Train-only vocabularies and train-only scalers make future evaluation honest.

Common Troubleshooting

- Never fit the scaler on the validation or test set.
- If you aggregate multiple datasets, either fit one global scaler on the training pool or fit one scaler per dataset, but do not refit using future rows.

9.14.7 Build Graph Tensors

Situation to Use

- Each row should become a graph.
- Grouped numeric summaries become graph nodes.
- Categorical context becomes context nodes.

Inputs and Outputs

- **Inputs:** one data frame, fitted scalers, category vocabularies.
- **Outputs:** grouped node tensor, categorical IDs, auxiliary tensor, labels.

Sample Code

Listing 9.12: Dataset that builds one graph per row

```
import torch
```

```
from torch.utils.data import Dataset


class PublicGraphDataset(Dataset):
    def __init__(self, df, group_feature_map, cat_cols, aux_cols,
                 node_scaler, aux_scaler, cat_vocab, label_col="label"):
        self.df = df.reset_index(drop=True).copy()
        self.group_names = list(group_feature_map.keys())
        self.group_feature_map = group_feature_map
        self.cat_cols = cat_cols
        self.aux_cols = aux_cols
        self.node_scaler = node_scaler
        self.aux_scaler = aux_scaler
        self.cat_vocab = cat_vocab
        self.label_col = label_col

        self.group_cols = [c for cols in group_feature_map.values() for c in cols
    ]

        # scale numeric values once for simplicity
        self.group_matrix = node_scaler.transform(self.df[self.group_cols]).
    astype("float32")
        self.aux_matrix = aux_scaler.transform(self.df[self.aux_cols]).astype("
    float32")
        self.y = self.df[label_col].to_numpy(dtype="float32")

        self.group_slices = {}
        start = 0
```

```
        for g in self.group_names:
            width = len(group_feature_map[g])
            self.group_slices[g] = slice(start, start + width)
            start += width

    def __len__(self):
        return len(self.df)

    def __getitem__(self, idx):
        # grouped nodes
        group_nodes = []
        for g in self.group_names:
            sl = self.group_slices[g]
            group_nodes.append(self.group_matrix[idx, sl])
        x_group = np.stack(group_nodes, axis=0)   # (G, F_g)

        # categorical ids
        cat_ids = []
        for col in self.cat_cols:
            raw = str(self.df.loc[idx, col]) if pd.notna(self.df.loc[idx, col])
    else "__UNK__"
            cat_ids.append(self.cat_vocab[col].get(raw, 0))
        cat_ids = np.asarray(cat_ids, dtype=np.int64)

        # auxiliary branch
        x_aux = self.aux_matrix[idx]
        y = self.y[idx]
```

```
    return (
        torch.tensor(x_group, dtype=torch.float32),
        torch.tensor(cat_ids, dtype=torch.long),
        torch.tensor(x_aux, dtype=torch.float32),
        torch.tensor(y, dtype=torch.float32),
    )
```

Usages

- Keeps the graph-building logic explicit and easy to debug.
- Makes it obvious which inputs belong inside the graph and which remain outside.

Common Troubleshooting

- All groups should have the same number of public summary columns in this simple example.
- If you need different widths per group, add a small projection layer for each group.

9.14.8 Define the GAT Model

Situation to Use

- You want node interactions to matter.
- Context variables should influence the grouped node representation.

Inputs and Outputs

- **Inputs:** grouped node tensor, categorical IDs, auxiliary tensor.

- **Outputs:** one fail logit per row.

Sample Code

Listing 9.13: Small dense GAT model in pure PyTorch

```
import math
import torch
import torch.nn as nn
import torch.nn.functional as F


class AttentionPool(nn.Module):
    def __init__(self, dim):
        super().__init__()
        self.proj = nn.Linear(dim, dim)
        self.score = nn.Linear(dim, 1)

    def forward(self, h):
        z = torch.tanh(self.proj(h))
        a = torch.softmax(self.score(z).squeeze(-1), dim=1).unsqueeze(-1)
        return (a * h).sum(dim=1)

class DenseGATLayer(nn.Module):
    def __init__(self, in_dim, out_dim, heads=4, dropout=0.1):
        super().__init__()
        self.heads = heads
        self.out_dim = out_dim
        self.dropout = dropout
```

```
        self.W = nn.Linear(in_dim, heads * out_dim, bias=False)
        self.a_src = nn.Parameter(torch.empty(heads, out_dim))
        self.a_dst = nn.Parameter(torch.empty(heads, out_dim))
        self.norm = nn.LayerNorm(heads * out_dim)

        nn.init.xavier_uniform_(self.W.weight)
        nn.init.xavier_uniform_(self.a_src)
        nn.init.xavier_uniform_(self.a_dst)

    def forward(self, h, adj_mask):
        B, N, _ = h.shape
        Wh = self.W(h).view(B, N, self.heads, self.out_dim)

        s1 = (Wh * self.a_src.view(1, 1, self.heads, self.out_dim)).sum(-1)
        s2 = (Wh * self.a_dst.view(1, 1, self.heads, self.out_dim)).sum(-1)

        s1 = s1.permute(0, 2, 1)
        s2 = s2.permute(0, 2, 1)
        e = F.leaky_relu(s1.unsqueeze(-1) + s2.unsqueeze(-2), 0.2)

        e = e.masked_fill(~adj_mask, -1e9)
        alpha = torch.softmax(e, dim=-1)
        alpha = F.dropout(alpha, p=self.dropout, training=self.training)

        Wh = Wh.permute(0, 2, 1, 3)
        out = torch.matmul(alpha, Wh)
        out = out.permute(0, 2, 1, 3).contiguous().view(B, N, -1)
```

```
        return self.norm(out)


class PublicPhysicsGAT(nn.Module):
    def __init__(self, num_groups, group_dim, cat_vocab_sizes, aux_dim,
                 hidden=64, cat_emb_dim=16):
        super().__init__()
        self.num_groups = num_groups
        self.num_cats = len(cat_vocab_sizes)
        self.hidden = hidden

        self.group_proj = nn.Linear(group_dim, hidden)

        self.cat_embs = nn.ModuleList([
            nn.Embedding(vocab_size, cat_emb_dim)
            for vocab_size in cat_vocab_sizes
        ])
        self.cat_proj = nn.Linear(cat_emb_dim, hidden)

        self.gat1 = DenseGATLayer(hidden, hidden // 4, heads=4, dropout=0.10)
        self.gat2 = DenseGATLayer(hidden, hidden // 4, heads=4, dropout=0.10)

        self.graph_pool = AttentionPool(hidden)
        self.head = nn.Sequential(
            nn.Linear(hidden + aux_dim, 128),
            nn.ReLU(),
            nn.Dropout(0.10),
            nn.Linear(128, 1),
```

```
        )

    def forward(self, x_group, cat_ids, x_aux, adj_mask):
        # grouped process nodes
        h_group = self.group_proj(x_group)             # (B, G, H)

        # context nodes
        ctx_nodes = []
        for j, emb in enumerate(self.cat_embs):
            ctx_nodes.append(self.cat_proj(emb(cat_ids[:, j])))
        h_ctx = torch.stack(ctx_nodes, dim=1)          # (B, C, H)

        # all nodes together
        h = torch.cat([h_group, h_ctx], dim=1)         # (B, G+C, H)

        h = F.relu(self.gat1(h, adj_mask))
        h = F.dropout(h, p=0.10, training=self.training)
        h = F.relu(self.gat2(h, adj_mask))

        z = self.graph_pool(h)
        logit = self.head(torch.cat([z, x_aux], dim=1)).squeeze(1)
        return logit


def build_dense_adj(num_groups, num_cats, device):
    n = num_groups + num_cats
    adj = torch.ones((1, 1, n, n), dtype=torch.bool, device=device)
```

```
    return adj
```

Usages

- This is a complete working teaching model with no extra graph package.
- The model is small enough for new users to read from top to bottom.

Common Troubleshooting

- If training is unstable, lower the learning rate.
- If the hidden size is not divisible by the number of heads, change the hidden size or the head count.

9.14.9 Train the Model

Situation to Use

- You have a binary label with class imbalance.
- You want a simple and reproducible training loop.

Inputs and Outputs

- **Inputs:** training dataset, validation dataset, hyperparameters.
- **Outputs:** trained model state with the best validation score.

Sample Code

Listing 9.14: Training loop

```
import numpy as np
```

```
from sklearn.metrics import roc_auc_score
from torch.utils.data import DataLoader


DEVICE = torch.device("cuda" if torch.cuda.is_available() else "cpu")
BATCH = 256
EPOCHS = 20
LR = 2e-3
PATIENCE = 5


train_ds = PublicGraphDataset(train_df, GROUP_FEATURE_MAP, CAT_COLS, AUX_COLS,
                              node_scaler, aux_scaler, cat_vocab, label_col=
    LABEL_COL)
val_ds = PublicGraphDataset(val_df, GROUP_FEATURE_MAP, CAT_COLS, AUX_COLS,
                            node_scaler, aux_scaler, cat_vocab, label_col=
    LABEL_COL)
test_ds = PublicGraphDataset(test, GROUP_FEATURE_MAP, CAT_COLS, AUX_COLS,
                              node_scaler, aux_scaler, cat_vocab, label_col=
    LABEL_COL)


train_loader = DataLoader(train_ds, batch_size=BATCH, shuffle=True)
val_loader = DataLoader(val_ds, batch_size=BATCH, shuffle=False)
test_loader = DataLoader(test_ds, batch_size=BATCH, shuffle=False)


cat_vocab_sizes = [len(cat_vocab[c]) for c in CAT_COLS]
num_groups = len(GROUP_FEATURE_MAP)
group_dim = len(next(iter(GROUP_FEATURE_MAP.values())))
aux_dim = len(AUX_COLS)
```

```
model = PublicPhysicsGAT(
    num_groups=num_groups,
    group_dim=group_dim,
    cat_vocab_sizes=cat_vocab_sizes,
    aux_dim=aux_dim,
).to(DEVICE)

adj_mask = build_dense_adj(num_groups, len(CAT_COLS), DEVICE)

# class weight from training set only
y_train = train_df[LABEL_COL].to_numpy()
n_pos = max(1, int((y_train == 1).sum()))
n_neg = max(1, int((y_train == 0).sum()))
pos_weight = torch.tensor([n_neg / n_pos], dtype=torch.float32, device=DEVICE)

criterion = nn.BCEWithLogitsLoss(pos_weight=pos_weight)
optimizer = torch.optim.AdamW(model.parameters(), lr=LR, weight_decay=1e-4)



def predict_probs(model, loader):
    model.eval()
    ys, ps = [], []
    with torch.no_grad():
        for x_group, cat_ids, x_aux, y in loader:
            x_group = x_group.to(DEVICE)
            cat_ids = cat_ids.to(DEVICE)
```

```
            x_aux = x_aux.to(DEVICE)
            logit = model(x_group, cat_ids, x_aux, adj_mask)
            prob = torch.sigmoid(logit).cpu().numpy()
            ys.append(y.numpy())
            ps.append(prob)
    return np.concatenate(ys), np.concatenate(ps)

best_auc = -np.inf
best_state = None
bad_epochs = 0

for epoch in range(1, EPOCHS + 1):
    model.train()
    loss_list = []

    for x_group, cat_ids, x_aux, y in train_loader:
        x_group = x_group.to(DEVICE)
        cat_ids = cat_ids.to(DEVICE)
        x_aux = x_aux.to(DEVICE)
        y = y.to(DEVICE)

        logit = model(x_group, cat_ids, x_aux, adj_mask)
        loss = criterion(logit, y)

        optimizer.zero_grad()
        loss.backward()
        torch.nn.utils.clip_grad_norm_(model.parameters(), max_norm=5.0)
```

```
        optimizer.step()
        loss_list.append(float(loss.item()))

    y_val, p_val = predict_probs(model, val_loader)
    val_auc = roc_auc_score(y_val, p_val) if np.unique(y_val).size == 2 else
    float("nan")
    print(f"epoch={epoch:02d} loss={np.mean(loss_list):.4f} val_auc={val_auc:.4f}
    ")

    if np.isfinite(val_auc) and val_auc > best_auc + 1e-4:
        best_auc = val_auc
        best_state = {k: v.detach().cpu().clone() for k, v in model.state_dict().
    items()}
        bad_epochs = 0
    else:
        bad_epochs += 1
        if bad_epochs >= PATIENCE:
            print("early stopping")
            break

if best_state is not None:
    model.load_state_dict(best_state)
```

Usages

- `BCEWithLogitsLoss` keeps the code numerically stable.
- `pos_weight` helps when fail rows are rare.
- Early stopping prevents wasting epochs after validation performance stops improving.

Common Troubleshooting

- If the loss becomes `nan`, check the input table for missing or infinite values before scaling.
- If validation AUC stays near 0.5, inspect whether the time split, labels, and column mapping are correct.

9.14.10 Evaluate on Validation and Future Test Data

Situation to Use

- You want both ranking quality and operating-point quality.
- Your deployment team may care about a maximum false positive rate.

Inputs and Outputs

- **Inputs:** trained model, validation loader, future test loader.
- **Outputs:** ROC-AUC, plus best TPR under an FPR cap.

Sample Code

Listing 9.15: Evaluation metrics

```
from sklearn.metrics import roc_auc_score
```

```
import numpy as np



def max_tpr_under_fpr_cap(y_true, y_prob, fpr_cap=0.20):
    y_true = np.asarray(y_true).astype(np.int64)
    y_prob = np.asarray(y_prob).astype(np.float32)

    order = np.argsort(-y_prob)
    y = y_true[order]
    p = y_prob[order]

    P = int((y == 1).sum())
    N = int((y == 0).sum())
    if P == 0 or N == 0:
        return float("nan"), float("nan")

    tp = np.cumsum(y == 1)
    fp = np.cumsum(y == 0)
    tpr = np.concatenate([[0.0], tp / P])
    fpr = np.concatenate([[0.0], fp / N])
    thr = np.concatenate([[np.inf], p])

    ok = np.where(fpr <= fpr_cap)[0]
    if len(ok) == 0:
        return float("nan"), float("nan")

    best = ok[np.argmax(tpr[ok])]
```

```
    return float(tpr[best]), float(thr[best])



y_val, p_val = predict_probs(model, val_loader)
y_test, p_test = predict_probs(model, test_loader)

print("VAL AUC :", roc_auc_score(y_val, p_val))
print("TEST AUC:", roc_auc_score(y_test, p_test))

best_tpr, best_thr = max_tpr_under_fpr_cap(y_test, p_test, fpr_cap=0.20)
print("TEST best TPR at FPR <= 0.20:", best_tpr)
print("Threshold used:", best_thr)
```

Usages

- ROC-AUC measures ranking quality across thresholds.
- TPR under an FPR cap is useful when false alarms are expensive.

Common Troubleshooting

- Do not choose the deployment threshold directly on the future test set if you want a perfectly clean final evaluation.
- A cleaner practice is: choose the threshold on validation, then report locked performance on the future test set once.

9.14.11 Data Leakage Checklist

Read this before publishing results

1. **Split by time first.** Future rows must not influence train, validation, or scaler fitting.
2. **Fit imputers, encoders, and scalers on training only.** Then reuse them for validation and test.
3. **If multiple datasets are aggregated, normalize carefully.** Either use one training-only global scaler or one training-only scaler per dataset.
4. **Do not fit PCA, UMAP, or any reducer on future data.**
5. **Do not pick the operating threshold on the future test set.**
6. **Check for duplicated units across splits.** Repeated runs from the same physical unit can make performance look unrealistically high.
7. **Audit engineered features.** A feature is only safe if it can be computed using information available at prediction time.

9.14.12 Quick Summary for New Users

Five-step recipe

1. Prepare a processed parquet table.
2. Split it into past pool and future test by time.
3. Fit train-only scalers and train-only categorical vocabularies.
4. Convert each row into grouped graph nodes plus auxiliary features.
5. Train the GAT, early-stop on validation, and evaluate once on the future test set.

9.14.13 Notes for Internal Customization

To adapt this public tutorial back to an internal project:

- replace `group_1`, `group_2`, …with real internal process groups,
- expand each group with the approved summary statistics,
- keep confidential raw signal names out of public documents,
- keep the time split and train-only preprocessing rules unchanged.

Chapter 10

Conclusion

Summary

This dissertation advanced the modeling, computation, and data-driven monitoring of semiconductor manufacturing processes centered on atomic layer methods. On the physics side, multiscale digital-twin frameworks were developed to couple reactor-scale transport with surface-reaction dynamics for thermal ALE and for ASALD [9, 175]. On the systems side, the dissertation developed control and optimization strategies tailored to cyclic processing, including multivariable run-to-run control and predictive temperature control, to improve robustness and reduce time-to-target under disturbances and practical constraints [176, 14, 177]. Finally, using large-scale industrial data, the dissertation developed machine-learning workflows for soft sensing and virtual metrology, emphasizing data preparation, cross-tool aggregation, and structured modeling to reduce dependence on costly offline measurements [178, 17, 179].

10.1 Synthesis and Cross-Cutting Takeaways

A central theme across all chapters is that manufacturing performance is governed by the interaction of physics, sensing, and decision-making. Self-limiting atomic layer processes can provide excellent precision, but they also create nonlinear saturation behavior and strong sensitivity to delivery dynamics (dose/purge timing, flow paths, and byproduct removal) [180, 181]. Accordingly, reliable scale-up and throughput improvements require methods that connect surface kinetics and reactor transport rather than treating them in isolation [9, 175].

A second theme is that control objectives must match what is measurable. When key quality metrics are not available in real time, cyclic (batch-to-batch) strategies such as run-to-run control can still reduce variability and improve efficiency, especially when integrated multivariable actions are designed explicitly for disturbance sources [176]. When dynamic actuators exist (e.g., lamp arrays), predictive control informed by compact data-driven models can provide fast setpoint achievement without unacceptable overshoot or uniformity loss [14, 182].

A third theme is that industrial value depends on robust learning under imperfect data. Soft sensing and virtual metrology are attractive precisely because they can reduce measurement burden, but they succeed only when data quality, aggregation logic, and model structure are engineered to handle scale, heterogeneity, and drift [183, 17]. In that context, structured learning paradigms such as attention (including graph attention for structured dependencies) provide a principled way to represent relationships among signals rather than relying solely on unstructured feature concatenation [184].

10.2 Chapter-Level Summary

Chapter 2 developed and applied a multiscale CFD framework for thermal ALE of Al_2O_3, enabling systematic evaluation of reactor geometries and operating strategies from the perspective of transport-limited saturation time and precursor utilization [9].

Chapters 3–4 developed multivariable run-to-run control strategies for thermal ALE, demonstrating how integrated control actions can reject kinetic and pressure disturbances more effectively than isolated single-loop policies [176].

Chapters 5–6 addressed thermal management using sparse identification and MPC, showing how compact identified dynamics can support fast ramping to target temperature while controlling overshoot and improving wafer temperature trajectories under actuator constraints [14, 185].

Chapter 7 developed an atomistic-to-mesoscopic model for ASALD that captures chemoselectivity, regioselectivity, and steric effects, providing mechanistic insight into how inhibitor and precursor behavior can enable selective growth [186].

Chapter 8 coupled surface modeling with reactor-scale CFD and demonstrated selectivity recovery through internal ALE steps, quantifying how nonideal precursor delivery changes the etch-time requirements for maintaining high selectivity over many cycles [175].

Chapter 9 developed industrial soft sensing models for etching tools, emphasizing data aggregation and statistically guided dataset selection to improve robustness and predictive accuracy in pass/fail classification and regression tasks [178].

Chapter 10 developed industrial multi-machine learning workflows for virtual metrology, including AI-ready data preparation, cross-machine aggregation, model comparison, and update strate-

gies to address drift while reducing reliance on frequent offline measurements [17].

10.3 Main Contributions

The principal contributions of this dissertation can be summarized as follows.

First, it developed multiscale digital-twin frameworks for thermal ALE that couple reactor-scale CFD with surface reaction dynamics to evaluate reactor configuration and operating conditions in terms of cycle time and precursor utilization [9].

Second, it proposed and studied control strategies aligned with cyclic atomic layer manufacturing, including multivariable run-to-run control under kinetically and transport relevant disturbances and predictive control strategies for thermal trajectories using sparse identified models [176, 14].

Third, it developed mechanistic-to-multiscale modeling for ASALD and introduced intermediate ALE steps as a quantitative selectivity recovery mechanism, explicitly demonstrating the need to account for reactor transport nonidealities when computing "sufficient" selectivity-preserving etch durations [186, 175].

Fourth, it developed industrial machine-learning workflows for soft sensing and virtual metrology, emphasizing data cleaning, multi-machine aggregation, and structured learning (including graph attention) to improve robustness and reduce metrology dependence in practical semiconductor manufacturing settings [178, 17, 184].

10.4 Future Work

Several research directions emerge naturally from the dissertation results.

A first direction is tighter integration of multiscale digital twins with optimization and control loops. For example, multiscale models can be used to generate fast surrogates for real-time decision-making, but doing so reliably requires systematic quantification of model error versus operating regime and disturbance level [9, 182].

A second direction is closed-loop selectivity management in ASALD. The intermediate-ALE selectivity recovery concept motivates feedback policies that adapt etch duration (or inhibitor dosing) based on inferred nucleation/defect states, especially under transport nonidealities and tool aging [175].

A third direction is industrial deployment science for soft sensing and virtual metrology. Future work should further formalize (i) cross-tool transfer and online updating under drift, (ii) uncertainty-aware decision thresholds, and (iii) physics-inspired constraints that improve out-of-distribution behavior and failure-mode sensitivity [17, 179].

A fourth direction is expanding structured representation learning for manufacturing data. Graph-based attention and related architectures could be extended to encode equipment topology, recipe graphs, and temporal-causal structure, enabling models that are more interpretable and more robust under configuration changes [184].

Bibliography

[1] S. Yun, M. Tom, F. Ou, G. Orkoulas, and P. D. Christofides. Multiscale computational fluid dynamics modeling of thermal atomic layer etching: Application to chamber configuration design. *Computers and Chemical Engineering*, 161:107757, 2022.

[2] A. Mameli, M. J. M. Merkx, B. Karasulu, F. Roozeboom, W. E. M. M. Kessels, and A. J. M. Mackus. Area-selective atomic layer deposition of SiO_2 using acetylacetone as a chemoselective inhibitor in an ABC-type cycle. *ACS Nano*, 11:9303–9311, 2017.

[3] S. Yun, F. Ou, H. Wang, M. Tom, G. Orkoulas, and P. D. Christofides. Atomistic-mesoscopic modeling of area-selective thermal atomic layer deposition. *Chemical Engineering Research and Design*, 188:271–286, 2022.

[4] C. A. Mack. Fifty years of Moore's law. *IEEE Transactions on Semiconductor Manufacturing*, 24:202–207, 2011.

[5] A. Razavieh, P. Zeitzoff, and E. J. Nowak. Challenges and limitations of CMOS scaling for FinFET and beyond architectures. *IEEE Transactions on Nanotechnology*, 18:999–1004, 2019.

[6] Y. Lee, G.-H. Kim, B. Choi, J. Yoon, H.-J. Kim, D. H. Kim, D. M. Kim, M.-H. Kang, and S.-J. Choi. Design study of the gate-all-around silicon nanosheet MOSFETs. *Semiconductor Science and Technology*, 35:03LT01, 2020.

[7] S. M. George. Atomic layer deposition: An overview. *Chemical Reviews*, 110:111–131, 2010.

[8] A. I. Abdulagatov and S. M. George. Thermal atomic layer etching of silicon using O_2, HF, and $Al(CH_3)_3$ as the reactants. *Chemistry of Materials*, 30(23):8465–8475, 2018.

[9] S. Yun, M. Tom, F. Ou, G. Orkoulas, and P. D. Christofides. Multiscale computational fluid dynamics modeling of thermal atomic layer etching: Application to chamber configuration design. *Computers & Chemical Engineering*, 161:107757, 2022.

[10] G. N. Parsons and R. D. Clark. Area-selective deposition: Fundamentals, applications, and future outlook. *Chemistry of Materials*, 32(12):4920–4953, 2020.

[11] F. Ou, A. Alghamdi, C.-P. Lin, G. Orkoulas, and P. D. Christofides. Multiscale modeling of area-selective atomic layer deposition of sio2 on al2o3 and sio2 surfaces with intermediate etching steps. *Chemical Engineering Science*, page 123422, 2026.

[12] S. Yun, M. Tom, F. Ou, G. Orkoulas, and P. D. Christofides. Multivariable run-to-run control of thermal atomic layer etching of aluminum oxide thin films. *Chemical Engineering Research & Design*, 182:1–12, 2022.

[13] J. S. Qin and T. A. Badgwell. An overview of industrial model predictive control technology. *AICHE Symposium Series*, 93:316, 1997.

[14] F. Ou, F. Abdullah, H. Wang, M. Tom, G. Orkoulas, and P. D. Christofides. Sparse identification modeling and predictive control of wafer temperature in an atomic layer etching reactor. *Chemical Engineering Research and Design*, 202:1–11, 2024. Early online date in DOI.

[15] P. Kadlec, B. Gabrys, and S. Strandt. Data-driven soft sensors in the process industry. *Computers & Chemical Engineering*, 33:795–814, 2009.

[16] F. Ou, H. Wang, C. Zhang, M. Tom, S. Bom, J. F. Davis, and P. D. Christofides. Industrial data-driven machine learning soft sensing for optimal operation of etching tools. *Digital Chemical Engineering*, 13:100195, 2024.

[17] F. Ou, J. Suherman, C. Zhang, H. Wang, S. Bom, J. F. Davis, and P. D. Christofides. Industrial multi-machine data aggregation, AI-ready data preparation, and machine learning for virtual metrology in semiconductor wafer and slider production. *Digital Chemical Engineering*, 15:100242, 2025.

[18] S. L. Brunton, J. L. Proctor, and N. J. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Applied Mathematics*, 113:3932–3937, 2016.

[19] M. Jurczak, N. Collaert, A. Veloso, T. Hoffmann, and S. Biesemans. Review of FINFET technology. *2009 IEEE International SOI Conference, Foster City, CA, USA*, pages 1–4, 2009.

[20] Y. Lee, G.-H. Kim, B. Choi, J. Yoon, H.-J. Kim, D. H. Kim, D. M. Kim, M.-H. Kang, and

S.-J. Choi. Design study of the gate-all-around silicon nanosheet MOSFETs. *Semiconductor Science and Technology*, 35:03LT01, 2020.

[21] D. Pan, T. Li, T.-C. Jen, and C. Yuan. Numerical modeling of carrier gas flow in atomic layer deposition vacuum reactor: A comparative study of lattice Boltzmann models. *Journal of Vacuum Science & Technology A*, 32:01A110, 2014.

[22] M. R. Shaeri, T.-C. Jen, and C. Y. Yuan. Reactor scale simulation of an atomic layer deposition process. *Chemical Engineering Research and Design*, 94:584–593, 2015.

[23] C. M. De la Huerta, V. H. Nguyen, J.-M. Dedulle, D. Bellet, C. Jiménez, and D. Muñoz-Rojas. Influence of the geometric parameters on the deposition mode in spatial atomic layer deposition: A novel approach to area-selective deposition. *Coatings*, 9:5, 2018.

[24] M. Crose, W. Zhang, A. Tran, and P. D. Christofides. Multiscale three-dimensional CFD modeling for PECVD of amorphous silicon thin films. *Computers & Chemical Engineering*, 113:184–195, 2018.

[25] Y. Zhang, Y. Ding, and P. D. Christofides. Multiscale computational fluid dynamics modeling and reactor design of plasma-enhanced atomic layer deposition. *Computers and Chemical Engineering*, 142:107066, 2020.

[26] Y. Zhang, Y. Ding, and P. D. Christofides. Multiscale computational fluid dynamics modeling of thermal atomic layer deposition with application to chamber design. *Chemical Engineering Research and Design*, 147:529–544, 2019.

[27] E. Granneman, P. Fischer, D. Pierreux, H. Terhorst, and P. Zagwijn. Batch ALD: Characteristics, comparison with single wafer ALD, and examples. *Surface and Coatings Technology*, 201:22–23, 2007.

[28] S. Yun, M. Tom, J. Luo, G. Orkoulas, and P. D. Christofides. Microscopic and data-driven modeling and operation of thermal atomic layer etching of aluminum oxide thin films. *Chemical Engineering Research and Design*, 177:96–107, 2022.

[29] S. Yun, Y. Ding, Y. Zhang, and P. D. Christofides. Integration of feedback control and run-to-run control for plasma enhanced atomic layer deposition of hafnium oxide thin films. *Computers and Chemical Engineering*, 148:107267, 2021.

[30] P. Peltonen, V. Vuorinen, G. Marin, A. J. Karttunen, and M. Karppinen. Numerical study on the fluid dynamical aspects of atomic layer deposition process. *Journal of Vacuum Science & Technology A*, 36:021516, 2018.

[31] T. Lill. *Atomic layer processing*. Wiley-VCH, Freemont, CA, 2021.

[32] K.-E. Elers, T. Blomberg, M. Peussa, B. Aitchison, S. Haukka, and S. Marcus. Film uniformity in atomic layer deposition. *Chemical Vapor Deposition*, 12:13–24, 2006.

[33] Y. Lee and S. M. George. Atomic layer etching of Al_2O_3 using sequential, self-limiting thermal reactions with $Sn(acac)_2$ and hydrogen fluoride. *ACS Nano*, 9:2061–2070, 2015.

[34] ANSYS. *Ansys Fluent Theory Guide*. ANSYS Inc., Canonsburg, PA, 2021.

[35] P. Giannozzi. Quantum espresso: a modular and open-source software project for quantum simulations of materials. *Journal of Physics: Condensed Matter*, 21:395502, 2009.

[36] S. Baroni, P. Giannozzi, and E. Isaev. Thermal properties of materials from ab initio quasi-harmonic phonons. *arXiv preprint arXiv:1112.4977*, 2011.

[37] DIPPR. *DIPPR Project 801*. Design Institute for Physical Properties, AIChE, New York, NY, 2020.

[38] S. Baroni, P. Giannozzi, and E. Isaev. Density-functional perturbation theory for quasi-harmonic calculations. *Reviews in Mineralogy & Geochemistry*, 71:39–58, 2010.

[39] A. Togo and I. Tanaka. First principles phonon calculations in materials science. *Scripta Materialia*, 108:1–5, 2015.

[40] Y. Lee, J. W. DuMont, and S. M. George. Trimethylaluminum as the metal precursor for the atomic layer etching of Al_2O_3 using sequential, self-limiting thermal reactions. *Chemistry of Materials*, 28:2994–3003, 2016.

[41] J. Voas, N. Kshetri, and J. F. DeFranco. Scarcity and global insecurity: The semiconductor shortage. *IT Professional*, 23:78–82, 2021.

[42] A. Razavieh, P. Zeitzoff, and E. J. Nowak. Challenges and limitations of CMOS scaling for FinFET and beyond architectures. *IEEE Transactions on Nanotechnology*, 18:999–1004, 2019.

[43] W. Lu, Y. Lee, J. Murdzek, J. Gertsch, A. Vardi, L. Kong, S. M. George, and J. A. del Alamo.

First transistor demonstration of thermal atomic layer etching: InGaAs finfets with sub-5 nm fin-width featuring in situ ale-ald. In *2018 IEEE International Electron Devices Meeting (IEDM)*, pages 39.1.1–39.1.4, San Francisco, CA, 2018.

[44] D. Metzler, R. L. Bruce, S. Engelmann, E. A. Joseph, and G. S. Oehrlein. Fluorocarbon assisted atomic layer etching of SiO_2 using cyclic Ar/C_4F_8 plasma. *Journal of Vacuum Science & Technology A*, 32(2):020603, 2014.

[45] K. S. Kim, K. H. Kim, Y. Nam, J. Jeon, S. Yim, E. Singh, J. Y. Lee, S. J. Lee, Y. S. Jung, G. Y. Yeom, and D. W. Kim. Atomic layer etching mechanism of MoS_2 for nanodevices. *ACS Applied Materials & Interfaces*, 9(13):11967–11976, 2017.

[46] C. Li, D. Metzler, C. S. Lai, E. A. Hudson, and G. S. Oehrlein. Fluorocarbon based atomic layer etching of Si_3N_4 and etching selectivity of SiO_2 over Si_3N_4. *Journal of Vacuum Science & Technology A*, 34(4):041307, 2016.

[47] S. Yun, M. Tom, J. Luo, G. Orkoulas, and P. D. Christofides. Microscopic and data-driven modeling and operation of thermal atomic layer etching of aluminum oxide thin films. *Chemical Engineering Research and Design*, 177:96–107, 2022.

[48] M. Crose, W. Zhang, A. Tran, and P. D. Christofides. Run-to-run control of PECVD systems: Application to a multiscale three-dimensional CFD model of silicon thin film deposition. *AIChE Journal*, 65:e16400, 2019.

[49] Z. Ning, J. Moyne, T. Smith, D. Boning, E. Del Castillo, J.-Y. Yeh, and A. Hurwitz. A comparative analysis of run-to-run control algorithms in the semiconductor manufacturing

industry. In *IEEE/SEMI 1996 Advanced Semiconductor Manufacturing Conference and Workshop. Theme-Innovative Approaches to Growth in the Semiconductor Industry. ASMC 96 Proceedings*, pages 375–381, 1996.

[50] W. Campbell, S. Firth, A. Toprac, and T. Edgar. A comparison of run-to-run control algorithms. In *Proceedings of the 2002 American Control Conference*, volume 3, pages 2150–2155, Anchorage, AK, 2002.

[51] Y. Zhang, Y. Ding, and P. D. Christofides. Integrating feedback control and run-to-run control in multi-wafer thermal atomic layer deposition of thin films. *Processes*, 8(1):18, 2020, 2020.

[52] J. Moyne, E. Del Castillo, and A. M. Hurwitz. *Run-to-Run Control in Semiconductor Manufacturing*. CRC Press, 2018.

[53] J. S.-I. Kwon, M. Nayhouse, and P. D. Christofides. Multiscale, multidomain modeling and parallel computation: Application to crystal shape evolution in crystallization. *Industrial & Engineering Chemistry Research*, 54:11903–11914, 2015.

[54] M. Crose, J. S.-I. Kwon, A. Tran, and P. D. Christofides. Multiscale modeling and run-to-run control of PECVD of thin film solar cells. *Renewable Energy*, 100:129–140, 2017.

[55] M. Broas, O. Kanninen, V. Vuorinen, M. Tilli, and M. Paulasto-Krockel. Chemically stable atomic-layer-deposited Al_2O_3 films for processability. *ACS Omega*, 2(7):3390–3398, 2017.

[56] A. P. J. Jansen, editor. *An Introduction to Kinetic Monte Carlo Simulations of Surface Reactions*, volume 1. Academic Press, London, 2012.

[57] D. T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, 1976.

[58] Y. Lou and P. D. Christofides. Feedback control of growth rate and surface roughness in thin film growth. *AIChE Journal*, 49:2099–2113, 2003.

[59] C.-F. Chien, Y.-J. Chen, C.-Y. Hsu, and H.-K. Wang. Overlay error compensation using advanced process control with dynamically adjusted proportional-integral R2R controller. *IEEE Transactions on Automation Science and Engineering*, 11:473–484, 2014.

[60] S.-K. S. Fan, B. C. Jiang, C.-H. Jen, and C.-C. Wang. SISO run-to-run feedback controller using triple EWMA smoothing for semiconductor manufacturing processes. *International Journal of Production Research*, 40:3093–3120, 2002.

[61] G. E. P. Box. Evolutionary operation: a method for increasing industrial productivity. *Journal of the Royal Statistical Society, Series C (Applied Statistics)*, 6:81–101, 1957.

[62] T.-C. Chou. Derivation and properties of michaelis-menten type and hill type equations for reference ligands. *Journal of Theoretical Biology*, 59:253–276, 1976.

[63] T.-C. Chou. The mass-action law based algorithms for quantitative econo-green bio-research. *Integrative Biology*, 3:548–559, 2011.

[64] C.-P. Wang, Y.-P. Tsai, B. J. Lin, Z.-Y. Liang, P.-W. Chiu, J.-R. Shih, C. J. Lin, and Y.-C. King. On-wafer FinFET-based EUV/eBeam detector arrays for advanced lithography processes. *IEEE Transactions on Electron Devices*, 67:2406–2413, 2020.

[65] P. Espadinha-Cruz, R. Godina, and E. M. G. Rodrigues. A review of data mining applications in semiconductor manufacturing. *Processes*, 9:305, 2021.

[66] S. Yun, M. Tom, G. Orkoulas, and P. D. Christofides. Multiscale computational fluid dynamics modeling of spatial thermal atomic layer etching. *Computers & Chemical Engineering*, 163:107861, 2022.

[67] S. Yun, M. Tom, F. Ou, G. Orkoulas, and P. D. Christofides. Multiscale computational fluid dynamics modeling of thermal atomic layer etching: Application to chamber configuration design. *Computers & Chemical Engineering*, 161:107757, 2022.

[68] T. J. Gyurcsik, Riley and F. Y. Sorrell. A model for rapid thermal processing: Achieving uniformity through lamp control. *IEEE Transactions on Semiconductor Manufacturing*, 4:9–13, 1991.

[69] F. Roozeboom and N. Parakh. Rapid thermal processing systems: A review with emphasis on temperature control. *Journal of Vacuum Science & Technology B: Microelectronics Processing and Phenomena*, 8:1249–1259, 1990.

[70] A. Theodoropoulou, A. R. Adomaitis, and E. Zafiriou. Model reduction for optimization of rapid thermal chemical vapor deposition systems. *IEEE Transactions on Semiconductor Manufacturing*, 1:85–98, 1998.

[71] E. Dassau, B. Grosman, and D. R. Lewin. Modeling and temperature control of rapid thermal processing. *Computers & Chemical Engineering*, 30:686–697, 2006.

[72] P. J. Timans. Rapid thermal processing technology for the 21st century. *Materials Science in Semiconductor Processing*, 1:3–4, 1998.

[73] H.-K. Oh, S. B. Kang, Y. K. Choi, and J. S. Lee. Optimization of rapid thermal processing for uniform temperature distribution on wafer surface. *Journal of Mechanical Science and Technology*, 23:1544–1552, 2009.

[74] J. Baker and P. D. Christofides. Output feedback control of parabolic PDE systems with non-linear spatial differential operators. *Industrial & Engineering Chemistry Research*, 38:4372–4380, 1999.

[75] Y. M. Cho, A. Paulraj, T. Kailath, and G. Xu. A contribution to optimal lamp design in rapid thermal processing. *IEEE Transactions on Semiconductor Manufacturing*, 7:34–41, 1994.

[76] C. D. Schaper, M. M. Moslehi, K. C. Saraswat, and T. Kailath. Modeling, identification, and control of rapid thermal processing systems. *Journal of the Electrochemical Society*, 141:3200, 1994.

[77] J. Baker and P. D. Christofides. Finite-dimensional approximation and control of non-linear parabolic PDE systems. *International Journal of Control*, 73:439–456, 2000.

[78] K.-E. Elers, T. Blomberg, M. Peussa, B. Aitchison, S. Haukka, and S. Marcus. Film uniformity in atomic layer deposition. *Chemical Vapor Deposition*, 12:13–24, 2006.

[79] J. S. Ponraj, G. Attolini, and M. Bosi. Review on atomic layer deposition and applications of oxide thin films. *Critical Reviews in Solid State and Materials Sciences*, 38:203–233, 2013.

[80] H. S. Fogler. *Elements of chemical reaction engineering*. Prentice Hall PTR, Upper Saddle River, NJ, 4 edition, 2006.

[81] P. Van Overschee and B. De Moor. N4SID: Subspace algorithms for the identification of combined deterministic-stochastic systems. *Automatica*, 30:75–93, 1994.

[82] A. Alanqar, H. Durand, and P. D. Christofides. On identification of well-conditioned nonlinear systems: Application to economic model predictive control of nonlinear processes. *AIChE Journal*, 61:3353–3373, 2015.

[83] Z. Wu, A. Tran, D. Rincon, and P. D. Christofides. Machine learning-based predictive control of nonlinear processes. Part I: theory. *AIChE Journal*, 65:e16729, 2019.

[84] Y. M. Ren, M. S. Alhajeri, J. Luo, S. Chen, F. Abdullah, Z. Wu, and P. D. Christofides. A tutorial review of neural network modeling approaches for model predictive control. *Computers & Chemical Engineering*, 165:107956, 2022.

[85] J. Proctor, S. Brunton, and B. Brunton. Exploiting sparsity and equation-free architectures in complex systems. *The European Physical Journal Special Topics*, 223:2665–2684, 2014.

[86] F. Abdullah and P. D. Christofides. Data-based modeling and control of nonlinear process systems using sparse identification: An overview of recent results. *Computers and Chemical Engineering*, 174:108247, 2023.

[87] J. Wang, J. Moreira, Y. Cao, and B. Gopaluni. Time-variant digital twin modeling through the kalman-generalized sparse identification of nonlinear dynamics. In *2022 American Control Conference (ACC)*, pages 5217–5222. IEEE, 2022.

[88] A. M. Stanković, A. A. Sarić, A. T. Sarić, and M. K. Transtrum. Data-driven symbolic regression for identification of nonlinear dynamics in power systems. In *2020 IEEE Power & Energy Society General Meeting (PESGM)*, pages 1–5. IEEE, 2020.

[89] C. E. Garcia, D. M. Prett, and M. Morari. Model predictive control: Theory and practice—a survey. *Automatica*, 25:335–348, 1989.

[90] D. Q. Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50:2967–2986, 2014.

[91] M. Leskela and M. Ritala. Atomic layer deposition (ALD): from precursors to thin film structures. *Thin Solid Films*, 409:138–146, 2002.

[92] A. J. M. Mackus, M. J. M. Merkx, and W. M. M. Kessels. From the bottom-up: Toward area-selective atomic layer deposition with high selectivity. *Chemistry of Materials*, 31:2–12, 2019.

[93] M. J. M. Merkx, S. Vlaanderen, T. Faraz, M. A. Verheijen, W. M. M. Kessels, and A. J. M. Mackus. Area-selective atomic layer deposition of TiN using aromatic inhibitor molecules for metal/dielectric selectivity. *Chemistry of Materials*, 32(18):7788–7795, 2020.

[94] M. Fang and J. C. Ho. Area-selective atomic layer deposition: Conformal coating, subnanometer thickness control, and smart positioning. *ACS Nano*, 9:8651–8654, 2015.

[95] S. Seo, B. C. Yeo, S. S. Han, C. M. Yoon, J. Y. Yang, J. Yoon, C. Yoo, H. jin Kim, Y. baek Lee, S. J. Lee, J.-M. Myoung, H.-B.-R. Lee, W.-H. Kim, I.-K. Oh, and H. Kim. Reaction

mechanism of area-selective atomic layer deposition for Al_2O_3 nanopatterns. *ACS Applied Materials & Interfaces*, 9:41607–41617, 2017.

[96] R. Chen, H. Kim, P. C. McIntyre, D. W. Porter, and S. F. Bent. Achieving area-selective atomic layer deposition on patterned substrates by selective surface modification. *Applied Physics Letters*, 86:191910, 2005.

[97] A. Sinha, D. W. Hess, and C. L. Henderson. Area selective atomic layer deposition of titanium dioxide: Effect of precursor chemistry. *Journal of Vacuum Science & Technology B: Microelectronics and Nanometer Structures Processing, Measurement, and Phenomena*, 24:2523–2532, 2006.

[98] J. Huang, M. Lee, A. Lucero, L. Cheng, and J. Kim. Area-selective ALD of TiO_2 nanolines with electron-beam lithography. *The Journal of Physical Chemistry C*, 118:23306–23312, 2014.

[99] F. S. Minaye Hashemi, C. Prasittichai, and S. F. Bent. Self-correcting process for high quality patterning by atomic layer deposition. *ACS Nano*, 9(9):8710–8717, 2015.

[100] A. Haider, M. Yilmaz, P. Deminskyi, H. Eren, and N. Biyikli. Nanoscale selective area atomic layer deposition of TiO_2 using e-beam patterned polymers. *RSC Adv.*, 6:106109–106119, 04 2016.

[101] J. Yarbrough, A. B. Shearer, and S. F. Bent. Next generation nanopatterning using small molecule inhibitors for area-selective atomic layer deposition. *Journal of Vacuum Science & Technology A*, 39(2):021002, 2021.

[102] M. J. M. Merkx, A. Angelidis, A. Mameli, J. Li, P. C. Lemaire, K. Sharma, D. M. Hausmann, W. M. M. Kessels, T. E. Sandoval, and A. J. M. Mackus. Relation between reactive surface sites and precursor choice for area-selective atomic layer deposition using small molecule inhibitors. *The Journal of Physical Chemistry C*, 126:4845–4853, 2022.

[103] M. J. M. Merkx, T. E. Sandoval, D. M. Hausmann, W. M. M. Kessels, and A. J. M. Mackus. Mechanism of precursor blocking by acetylacetone inhibitor molecules during area-selective atomic layer deposition of SiO_2. *Chemistry of Materials*, 32(8):3335–3345, 2020.

[104] S. Yun, M. Tom, G. Orkoulas, and P. D. Christofides. Multiscale computational fluid dynamics modeling of spatial thermal atomic layer etching. *Computers & Chemical Engineering*, 163:107861, 2022.

[105] M. M. Folkendt, B. E. Weiss-Lopez, J. P. C. Jr., and N. S. True. Gas-phase proton nmr studies of keto-enol tautomerism of acetylacetone, methyl acetoacetate, and ethyl acetoacetate. *The Journal of Physical Chemistry*, 89:3347–3352, 1985.

[106] J. C. Gamekkanda, A. S. Sinha, J. Desper, M. Đaković, and C. B. Aakeröy. The role of halogen bonding in controlling assembly and organization of Cu(II)-Acac based coordination complexes. *Crystals*, 7:226, 2017.

[107] K. J. Hughes, A. Dube, M. Sharma, and J. R. Engstrom. Initial stages of atomic layer deposition of tantalum nitride on SiO_2 and porous low-κ substrates modified by a branched interfacial organic layer: Chemisorption and the transition to steady-state growth. *The Journal of Physical Chemistry C*, 116:21948–21960, 2012.

[108] S. Baroni, P. Giannozzi, and E. Isaev. Thermal properties of materials from ab initio quasi-harmonic phonons. *arXiv preprint arXiv:1112.4977*, 2011.

[109] P. Giannozzi. QUANTUM ESPRESSO: a modular and open-source software project for quantum simulations of materials. *Journal of Physics: Condensed Matter*, 21:395502, 2009.

[110] P. Pitriana, T. D. K. Wungu, H. Herman, and R. Hidayat. The computation parameters optimizations for electronic structure calculation of $LiPbI_3$ perovskite by the density functional theory method. *IOP Conference Series Materials Science and Engineering*, 434:012026, 2018.

[111] H. Roh, H.-L. Kim, K. Khumaini, H. Son, D. Shin, and W.-J. Lee. Effect of deposition temperature and surface reactions in atomic layer deposition of silicon oxide using bis(diethylamino)silane and ozone. *Applied Surface Science*, 571:151231, 2022.

[112] M. C. Schwille, T. Schössler, F. Schön, M. Oettel, and J. W. Bartha. Temperature dependence of the sticking coefficients of bis-diethyl aminosilane and trimethylaluminum in atomic layer deposition. *Journal of Vacuum Science & Technology A*, 35:01B119, 2017.

[113] G. Lee, B. Lee, J. Kim, and K. Cho. Ozone adsorption on graphene: Ab initio study and experimental validation. *The Journal of Physical Chemistry C*, 113:14225–14229, 2009.

[114] A. P. J. Jansen, editor. *An Introduction to Kinetic Monte Carlo Simulations of Surface Reactions*, volume 1. Academic Press, 2012.

[115] M. Hu, T. C. Hauger, B. C. Olsen, E. J. Luber, and J. M. Buriak. UV-initiated Si–S, Si–Se,

and Si–Te bond formation on Si(111): Coverage, mechanism, and electronics. *The Journal of Physical Chemistry C*, 122(25):13803–13814, 2018.

[116] L. Jin, Y. Li, Z. Hu, and J. Chu. Full three-dimensional morphology evolution of amorphous thin films for atomic layer deposition. *AIP Advances*, 8:045304, 2018.

[117] T. Weckman, M. Shirazi, S. D. Elliott, and K. Laasonen. Kinetic monte carlo study of the atomic layer deposition of zinc oxide. *The Journal of Physical Chemistry C*, 122:27044–27058, 2018.

[118] H. G. Kim, M. Kim, B. Gu, M. R. Khan, B. G. Ko, S. Yasmeen, C. S. Kim, S.-H. Kwon, J. Kim, J. Kwon, K. Jin, B. Cho, J. S. Chun, B. Shong, and H.-B.-R. Lee. Effects of Al precursors on deposition selectivity of atomic layer deposition of Al_2O_3 using ethanethiol inhibitor. *Chemistry of Materials*, 32:8921–8929, 2020.

[119] F. M. Richards. The interpretation of protein structures: Total volume, group volume distributions and packing density. *Journal of Molecular Biology*, 82:1–14, 1973.

[120] D. Maroudas. Multiscale modeling of hard materials: Challenges and opportunities for chemical engineering. *AIChE Journal*, 46:878–882, 2000.

[121] A. Bortz, M. H. Kalos, and J. L. Lebowitz. A new algorithm for monte carlo simulation of ising spin systems. *Journal of Computational Physics*, 17:10–18, 1975.

[122] M. Shirazi and S. D. Elliott. Atomistic kinetic monte carlo study of atomic layer deposition derived from density functional theory. *Journal of Computational Chemistry*, 35:244–259, 2014.

[123] J. Huang, G. Hu, G. Orkoulas, and P. D. Christofides. Dynamics and lattice-size dependence of surface mean slope in thin-film deposition. *Industrial & Engineering Chemistry Research*, 50:1219–1230, 2010.

[124] N. Loubet, T. Hook, P. Montanini, C.-W. Yeung, S. Kanakasabapathy, M. Guillom, T. Yamashita, J. Zhang, X. Miao, J. Wang, A. Young, R. Chao, M. Kang, Z. Liu, S. Fan, B. Hamieh, S. Sieg, Y. Mignot, W. Xu, S.-C. Seo, J. Yoo, S. Mochizuki, M. Sankarapandian, O. Kwon, A. Carr, A. Greene, Y. Park, J. Frougier, R. Galatage, R. Bao, J. Shearer, R. Conti, H. Song, D. Lee, D. Kong, Y. Xu, A. Arceo, Z. Bi, P. Xu, R. Muthinti, J. Li, R. Wong, D. Brown, P. Oldiges, R. Robison, J. Arnold, N. Felix, S. Skordas, J. Gaudiello, T. Standaert, H. Jagannathan, D. Corliss, M.-H. Na, A. Knorr, T. Wu, D. Gupta, S. Lian, R. Divakaruni, T. Gow, C. Labelle, S. Lee, V. Paruchuri, H. Bu, and M. Khare. Stacked nanosheet gate-all-around transistor to enable scaling beyond FinFET. In *2017 Symposium on VLSI Technology*, pages T230–T231, Kyoto, Japan, 2017.

[125] M. F. Vos, S. N. Chopra, M. A. Verheijen, J. G. Ekerdt, S. Agarwal, W. M. Kessels, and A. J. Mackus. Area-selective deposition of ruthenium by combining atomic layer deposition and selective etching. *Chemistry of Materials*, 31:3878–3882, 2019.

[126] R. Vallat, R. Gassilloud, B. Eychenne, and C. Vallée. Selective deposition of ta2o5 by adding plasma etching super-cycles in plasma enhanced atomic layer deposition steps. *Journal of Vacuum Science & Technology A*, 35, 2017.

[127] J. W. DuMont, A. E. Marquardt, A. M. Cano, and S. M. George. Thermal atomic layer etching

of sio2 by a "conversion-etch" mechanism using sequential reactions of trimethylaluminum and hydrogen fluoride. *ACS Applied Mmaterials & Interfaces*, 9:10296–10307, 2017.

[128] R. Rahman, E. C. Mattson, J. P. Klesko, A. Dangerfield, S. Rivillon-Amy, D. C. Smith, D. Hausmann, and Y. J. Chabal. Thermal atomic layer etching of silica and alumina thin films using trimethylaluminum with hydrogen fluoride or fluoroform. *ACS Applied Materials & Interfaces*, 10:31784–31794, 2018.

[129] G. Lee, B. Lee, J. Kim, and K. Cho. Ozone adsorption on graphene: Ab initio study and experimental validation. *The Journal of Physical Chemistry C*, 113:14225–14229, 2009.

[130] F. Pieck and R. Tonner-Zech. Computational ab initio approaches for area-selective atomic layer deposition: Methods, status, and perspectives. *Chemistry of Materials*, 37:2979–3021, 2025.

[131] M. Mokhtarzadeh, M. Carulla, R. Kozak, and C. David. Optimization of etching processes for the fabrication of smooth silicon carbide membranes for applications in quantum technology. *Micro and Nano Engineering*, 16:100155, 2022.

[132] M. Martin and G. Cunge. Surface roughness generated by plasma etching processes of silicon. *Journal of Vacuum Science & Technology B: Microelectronics and Nanometer Structures Processing, Measurement, and Phenomena*, 26:1281–1288, 2008.

[133] W. A. Kimes, E. Moore, and J. Maslar. Design and operation of an optically-accessible modular reactor for diagnostics of thermal thin film deposition processes. *Review of Scientific Instruments*, 83, 2012.

[134] J. E. Maslar and B. Kalanyan. Visualizing molybdenum pentachloride flow during vapor deposition processes using absorption imaging. *Applied Spectroscopy*, 79:1487–1496, 2025.

[135] Y. Ding, Y. Zhang, K. Kim, A. Tran, Z. Wu, and P. D. Christofides. Microscopic modeling and optimal operation of thermal atomic layer deposition. *Chemical Engineering Research and Design*, 145:159–172, 2019.

[136] T. Q. Nguyen, T. Hoang, L. Zhang, O. A. Dobre, and T. Q. Duong. A survey on smart optimisation techniques for 6g-oriented integrated circuits design. *Mobile Networks and Applications*, pages 1–18, 2024.

[137] D. Burg and J. H. Ausubel. Moore ' s law revisited through intel chip density. *PLoS one*, 16:e0256245, 2021.

[138] P. Nguyen, D. Ivanov, and F. Sgarbossa. A digital twin–based approach to reinforce supply chain resilience: Simulation of semiconductor shortages. In *Proceeding of IFIP International Conference on Advances in Production Management Systems*, pages 563–576. Springer, 2023.

[139] P. D. Christofides, J. F. Davis, N. H. El-Farra, D. Clark, K. R. Harris, and J. N. Gipson. Smart plant operations: Vision, progress and challenges. *AIChE Journal*, 53:2734–2741, 2007.

[140] C. Fuchs. Industry 4.0: the digital german ideology. *Triplec: Communication, Capitalism & Critique*, 16:280–289, 2018.

[141] S. N. Songkhla and T. Nakamoto. Overview of quartz crystal microbalance behavior analysis and measurement. *Chemosensors*, 9:350, 2021.

[142] S. Wang, X. Liu, and P. Zhou. The road for 2D semiconductors in the silicon age. *Advanced Materials*, 34:2106886, 2022.

[143] Y. S. Perera, D. Ratnaweera, C. H. Dasanayaka, and C. Abeykoon. The role of artificial intelligence-driven soft sensors in advanced sustainable process industries: A critical review. *Engineering Applications of Artificial Intelligence*, 121:105988, 2023.

[144] P. O' Donovan, K. Leahy, K. Bruton, and D. T. O' Sullivan. Big data in manufacturing: a systematic mapping study. *Journal of Big Data*, 2:1–22, 2015.

[145] Q. Sun and Z. Ge. A survey on deep learning for data-driven soft sensors. *IEEE Transactions on Industrial Informatics*, 17(9):5853–5866, 2021.

[146] S. Coleman, D. Kerr, and Y. Zhang. Image sensing and processing with convolutional neural networks. *Sensors*, 22:3612, 2022.

[147] M. Lee, J. Bae, and S. B. Kim. Uncertainty-aware soft sensor using bayesian recurrent neural networks. *Advanced Engineering Informatics*, 50:101434, 2021.

[148] S. Hong, N. An, H. Cho, J. Lim, I.-S. Han, I. Moon, and J. Kim. A dynamic soft sensor based on hybrid neural networks to improve early off-spec detection. *Engineering with Computers*, 39:3011–3021, 2023.

[149] C. Zhang, J. Yella, Y. Huang, X. Qian, S. Petrov, A. Rzhetsky, and S. Bom. Soft sensing transformer: hundreds of sensors are worth a single word. In *Proceedings of IEEE International Conference on Big Data*, pages 1999–2008, 2021.

[150] I. H. Sarker. Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, 2:160, 2021.

[151] G. Ranganathan. A study to find facts behind preprocessing on deep learning algorithms. *Journal of Innovative Image Processing*, 3:66–74, 2021.

[152] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[153] A. Rehmer and A. Kroll. On the vanishing and exploding gradient problem in gated recurrent units. *IFAC-PapersOnLine*, 53:1243–1248, 2020.

[154] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[155] R. D. King, O. I. Orhobor, and C. C. Taylor. Cross-validation is safe to use. *Nature Machine Intelligence*, 3:276–276, 2021.

[156] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. *arXiv preprint arXiv:1505.00853*, 2015.

[157] A. A. Waoo and B. K. Soni. Performance analysis of sigmoid and relu activation functions in deep neural network. In *Intelligent Systems: Proceedings of SCIS 2021*, pages 39–52. Springer, 2021.

[158] E. A. Elsayed. Overview of reliability testing. *IEEE Transactions on Reliability*, 61:282–291, 2012.

[159] L. Gonçalves, A. Subtil, M. R. Oliveira, and P. de Zea Bermudez. ROC curve estimation: An overview. *REVSTAT-Statistical Journal*, 12:1–20, 2014.

[160] Casanova. Chip sales rise in 2022, especially to auto, industrial, consumer markets, March 2023.

[161] W. Mohammad, A. Elomri, and L. Kerbache. The global semiconductor chip shortage: Causes, implications, and potential remedies. *IFAC-PapersOnLine*, 55(10):476–483, 2022.

[162] X. Wu, C. Zhang, and W. Du. An analysis on the crisis of "chips shortage" in automobile industry——based on the double influence of covid-19 and trade friction. In *Journal of Physics: Conference Series*, volume 1971, page 012100, 2021.

[163] H. Malkani and P. Korambath. Special issue: Smart manufacturing. *Journal of Advanced Manufacturing Processes*, 2022. Special Issue.

[164] J. Davis, H. Malkani, J. Dyck, P. Korambath, and J. Wise. Chapter 4 - cyberinfrastructure for the democratization of smart manufacturing. In M. Soroush, M. Baldea, and T. F. Edgar, editors, *Smart Manufacturing*, pages 83–116. Elsevier, 2020.

[165] A. Tsanousa, E. Bektsis, C. Kyriakopoulos, A. G. González, U. Leturiondo, I. Gialampoukidis, A. Karakostas, S. Vrochidis, and I. Kompatsiaris. A review of multisensor data fusion solutions in smart manufacturing: Systems and trends. *Sensors*, 22:1734, 2022.

[166] J. F. Davis, S. Biller, J. S. Pierre, and S. Jahanmir. Towards resilient manufacturing ecosystems through artificial intelligence. Symposium Report 100-47, National Institute of Standards and Technology (NIST), Gaithersburg, MD, 2022.

[167] Y. Jiang, S. Yin, J. Dong, and O. Kaynak. A review on soft sensors for monitoring, control, and optimization of industrial processes. *IEEE Sensors Journal*, 21:12868–12881, 2020.

[168] J. Konyha and T. Bányai. Sensor networks for smart manufacturing processes. *Solid State Phenomena*, 261:456–462, 2017.

[169] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[170] L. Zhang, Z. Deng, K. Kawaguchi, A. Ghorbani, and J. Zou. How does mixup help with robustness and generalization? *arXiv preprint arXiv:2010.04819*, 2020.

[171] C. Albon. *Machine learning with python cookbook: Practical solutions from preprocessing to deep learning*. O'Reilly Media, 2018.

[172] X. Zheng, M. Wang, and J. Ordieres-Meré. Comparison of data preprocessing approaches for applying deep learning to human activity recognition in the context of industry 4.0. *Sensors*, 18:2146, 2018.

[173] J. P. Card, M. Naimo, and W. Ziminsky. Run-to-run process control of a plasma etch process with neural network modelling. *Quality and Reliability Engineering International*, 14:247–260, 1998.

[174] B. P. Salmon, W. Kleynhans, C. P. Schwegmann, and J. C. Olivier. Proper comparison among methods using a confusion matrix. In *Proceedings of IEEE International Geoscience and Remote Sensing Symposium*, pages 3057–3060, Milan, Italy, 2015.

[175] F. Ou, A. Alghamdi, C.-P. Lin, G. Orkoulas, and P. D. Christofides. Multiscale modeling of area-selective atomic layer deposition of SiO_2 on Al_2O_3 and SiO_2 surfaces with intermediate etching steps. *Chemical Engineering Science*, TODO:123422, 2026. TODO: Fill in volume/issue once finalized.

[176] S. Yun, M. Tom, F. Ou, G. Orkoulas, and P. D. Christofides. Multivariable run-to-run control of thermal atomic layer etching of aluminum oxide thin films. *Chemical Engineering Research and Design*, 182:1–12, 2022.

[177] J. S. Qin and T. A. Badgwell. An overview of industrial model predictive control technology. *AIChE Symposium Series*, 93:316, 1997.

[178] F. Ou, H. Wang, C. Zhang, M. Tom, S. Bom, J. F. Davis, and P. D. Christofides. Industrial data-driven machine learning soft sensing for optimal operation of etching tools. *Digital Chemical Engineering*, 13:100195, 2024.

[179] Q. Sun and Z. Ge. A survey on deep learning for data-driven soft sensors. *IEEE Transactions on Industrial Informatics*, 17(9):5853–5866, 2021.

[180] S. M. George. Atomic layer deposition: An overview. *Chemical Reviews*, 110(1):111–131, 2010.

[181] A. I. Abdulagatov and S. M. George. Thermal atomic layer etching of Al_2O_3 using HF and $Al(CH_3)_3$ as the reactants. *Chemistry of Materials*, 30(23):8465–8475, 2018.

[182] D. Q. Mayne. Model predictive control: Recent developments and future promise. *Automatica*, 50:2967–2986, 2014.

[183] P. Kadlec, B. Gabrys, and S. Strandt. Data-driven soft sensors in the process industry. *Computers & Chemical Engineering*, 33:795–814, 2009.

[184] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018. Also available as arXiv:1710.10903.

[185] S. L. Brunton, J. L. Proctor, and J. N. Kutz. Discovering governing equations from data by sparse identification of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 113:3932–3937, 2016.

[186] S. Yun, F. Ou, H. Wang, M. Tom, G. Orkoulas, and P. D. Christofides. Atomistic-mesoscopic modeling of area-selective thermal atomic layer deposition. *Chemical Engineering Research and Design*, 188:271–286, 2022.