UNIVERSITY OF CALIFORNIA
SANTA CRUZ

NAND FLASH MEMORY CHARACTERIZATION

A thesis submitted in partial satisfaction of the
requirements for the degree of

MASTERS OF SCIENCE

in

COMPUTER ENGINEERING

by

Tanvir S. Heer

December 2019

The Thesis of Tanvir S. Heer
is approved:

Professor Sung-Mo Steve Kang, Chair

Professor Chen Qian

Professor Patrick E. Mantey

Quentin Williams
Acting Vice Provost and Dean of Graduate Studies

Table of Contents

List of Figures

Abstract

NAND Flash Memory Characterization

by

Tanvir S. Heer

The NAND technology has become a popular research area and implementation choice due to its non-volatile flash memory characteristics. There are many engineering challenges when it comes to NAND technology. Some of the limiting factors are reducing the transistor width and increasing read and write performance. The device physics for NAND floating gate cell technology also introduces challenges. Using a floating gate transistor that can address up to 4 bits per cell allows for higher density. However, this introduces a higher internal current leakage that can cause read and program inaccuracies. With floating-gate 3D NAND technology, we are able to better the data retention and have better QLC (Quad-Level Cell) capability. 3D NAND technology has a series of memory cells interconnected that can achieve higher data density and increased storage capacity. In this research, the Gen-4 NAND Flash device functionality is explored by design validation to achieve higher solid-state drive performance. This technology is 144-Layer QLC NAND Flash which is 1024Gb in density. My contributions are implemented showing improved read and write performance for customer operations and how inaccuracies are dealt with from a software and hardware perspective. Silicon wafer testing is also explored to fully characterize and analyze the problem.

Acknowledgments

I would first like to thank my advisor, Professor Sung-Mo Steve Kang for allowing me the opportunity to have him as my advisor and allowing me to implement my work at Intel Corporation. Professor Kang became my advisor through his course he taught in Low-Power CMOS Design. He is a great professor and is very understanding. His coursework and insight led me to choose a topic related to CMOS design; specifically, NAND technology. I have been truly inspired by his course and research and hope to continue what I have learned as a student and continue implementing this knowledge in industry.

Sincere thanks to Professor Chen Qian and Professor Patrick Mantey who are on my thesis committee and have been my professors at the University of California of Santa Cruz. I would like to thank them for allowing me to take their courses and becoming part of my committee.

Humble thanks to my manager Daniel Castro at Intel Corporation. Through Dan, I was able to conduct research at Intel while working as a full-time graduate technical intern in post-silicon validation engineering. The past 12 months has given me a wide variety of knowledge of how the industry works and learning the semiconductor process and analysis. Thank you for allowing me to focus on my thesis and being able to contribute to Intel at the same time.

Finally, I would like to thank everyone who I have worked with at Intel Corporation; specifically the design engineering teams in conducting my research and analysis.

Chapter 1

Introduction

1.0.1 Device Specifications and Architecture

NAND Flash Memory is a technology that provides cost-effective solutions for solid-state drives that allow high density storage. In this project, methods are explored related to design validation, research and testing of the product for the best performance and overcoming certain engineering challenges. The objective presented here is to provide a range of device capabilities and define high speed NAND interface technology that is compatible with today's technology interfaces. Before diving into the features and operations of the NAND device, understanding the general high-level architecture and specifications of the NAND Flash Memory Device will be shown first. Floating gate technology has many applications of where it can be used; such as, FPGAs, replacements for hard drives, serial flash, and embedded systems. This technology replaces ROM memory with Quad-level cell NAND Flash memory.

Figure 1.1 below shows the operating temperature range, VCC, density, and

the block size of the NAND device. The NAND solid-state drive requires supply voltage within 2.7V to 3.6V and 0°C-70°C ranges; however, analysis is conducted at much more conservative voltage levels as low as 2.1V. In terms of temperature, characterization is conducted as low as -20°C to as high as 90°C.

Block and Page Size	Device Size	VCC	Operating Temperature
64 Pages (128K + 4k Bytes) x16: 1,056 words	8GB: 8,192 Blocks (Quad-die stack)	2.7V - 3.6V	0°C - 70°C (Testing: -20°C to 90°C)

Figure 1.1: NAND Device Specifications

In Figure 1.2, the NAND device that will be focused on has a block size of 64 pages, 128k bytes + 4k Spare bytes. Each block contains 64 pages and each page is consisted of 1,056 words, where each word is 32 bytes. In a NAND device, the page is the smallest addressable unit for executing erase, read, and write algorithms. Figure 1.2 gives a detailed explanation of this array structure. A single block has 64 pages (64K + 2k spare) words and each page has (1k + 32 spare) words. Therefore, one block has (1k + 32) words X 64 pages (64K + 2k) words. In one device, there are a total number of 2,048 blocks, totalling in 4Mb.

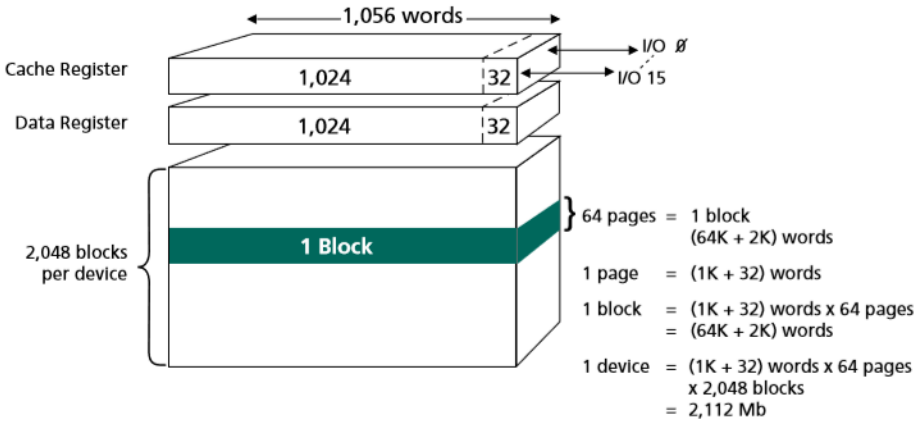


Figure 1.2: Array Organization [3]

Chapter 2

Background

2.0.1 Device Addressing

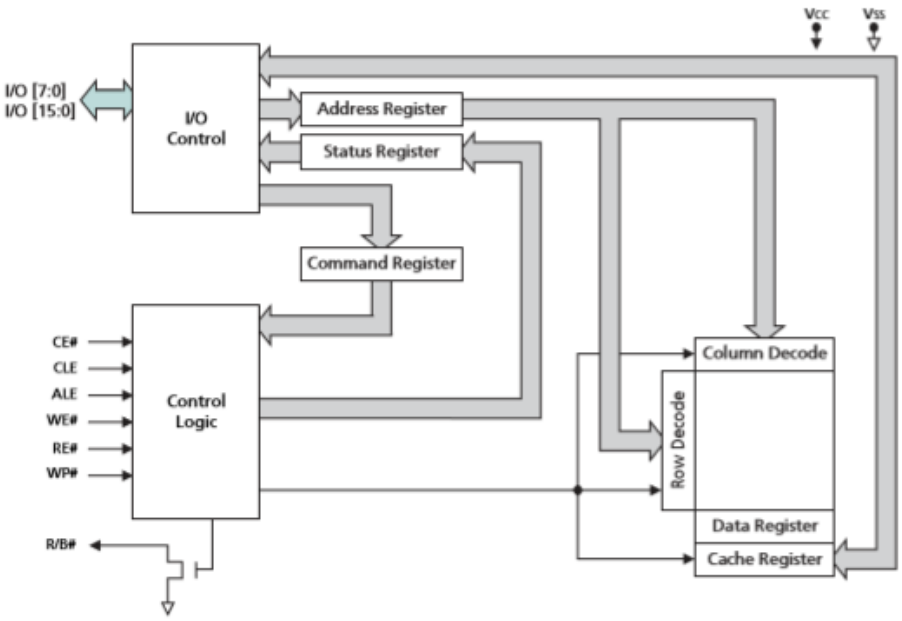


Figure 2.1: NAND Functional Block Diagram [3]

Moving forward with the background of the NAND device, bus operations and features are the fundamental area to conceptualize before implementation. First a good understanding must be established regarding the NAND Flash functional block diagram as shown in Figure 2.1. At a very high level, the NAND technology uses set commands and addressing functionality via the I/O control logic. The data is transferred from flash memory array through the data and cache registers. PROGRAM and READ operations are done via page based by issuing a set page number; whereas, ERASE operations are done by block operations. The NAND device uses a highly multiplexed 8-bit[7:0] I/O control for transferring data, commands, and instructions to have optimal control functionality. The I/O is crucial, these pins are bi-directional pins that allow the user to communicate to the device with via the given data address with instruction type.

The control logic has 6 command pins – CE, CLE, ALE, WE, RE, and WP. These pins are used for interfacing the NAND device using bus interface protocol. The addressing for NAND device is broken down to row address and column address. The control logic allows to create a communication link to input user data to NAND device via the 8-bit I/O[7:0]. The ALE input signal allows to transfer address information from the I/O[7:0] directly to the chip address register, depending on if this signal needs to be I/O high or low. The CE input signal is chip enable. CE input signal establishes communication between the host system and the NAND device. The CLE input signal sets transfer command information, it is similar to the ALE signal but the usage here is for enabling the command latch and transferring I/O[7:0] to the command registers for

the chip. RE and WE are input signals that are important for read and write enable. This allows the internal circuitry to get data transferred from NAND device to host system and host system to the NAND device. One other important pin is the R/B. The pin is called the Ready-Busy pin, this signal tells the chip if it is undergoing any operations. For example, if a READ operation is in process, the Read-Busy pin will indicate that data is being transferred from array to data registers. This signal goes active low when this process is occurring for any operations and goes back high once the operation has been completed.

2.0.2 Array Addressing

As mentioned in Figure 2.1 regarding the array organization (Column and Row Decoding), the row address is composed of the page, block and LUN (Logical Unit Number) that needs to be accessed by the NAND device. The column address allows access to the bit/byte(s) for that certain page. Both row and column address must be defined and issued before sending an instruction; such as, reading and writing to the device. An Erase operation does not necessarily need both the row and column addresses, only the row address needs to be specified. The data and cache register seen in the in Figure 2.1 are used to increase programming bandwidth. It is important to incorporate caching for better performance. The NAND device uses a 5-cycle addressing sequence that must be issued in order to execute a sequence of operations, this is shown in Figure 2.2 on the next page. There are 5 cycles, each cycle has a set column address (CA) and Row Address (RA) that must be set before any READ or PROGRAM

operation is executed. Both column and row address bits are required to be issued. Addressing is defined for single plane and multi-plane and it is important to note that the address cycles required change depending on which command is issued to the device. An example of issuing a program command (80h) is shown: [CMD: 80h - ADDR: Column Row - DIN: Data Input - CMD: 10h]. In this command, the column address is first defined before the row address for READ and PROGRAM operations. Only the block address is specified if issuing an ERASE operation since the column address does not matter when erasing an entire block.

Cycle	I/O[15:8]	I/O7	I/O6	I/O5	I/O4	I/O3	I/O2	I/O1	I/O0
First	LOW	CA7	CA6	CA5	CA4	CA3	CA2	CA1	CA0
Second	LOW	LOW	LOW	LOW	LOW	LOW	CA10	CA9	CA8
Third	LOW	RA18	RA17	RA16	RA15	RA14	RA13	RA12	RA11
Fourth	LOW	RA26	RA25	RA24	RA23	RA22	RA21	RA20	RA19
Fifth	LOW	LOW	LOW	LOW	LOW	LOW	LOW	RA28[1]	RA27

Figure 2.2: Array Addressing [3]

2.0.3 Bus Operations and Status Register

Data input is the next important topic. As mentioned earlier, data is written from the I/O[7:0] input to the data register. There are defined user inputs in order to properly input data via the control signals (CE, WE, RE, CLE, ALE, and WP) as discussed above. These commands are written to the command register when CE, ALE are active low and CLE is active high. One other important criteria that is checked is if the device is in busy state. If the device is busy, commands can not be issued to the device. This is why the ready busy signal is important for the NAND device.

Electrically, the Ready-Busy (RB) is implemented as an open drain circuit as shown in Figure 2.3 below. There is a pull-up resistor that is connected that determines the internal rise time of RB. Internally, the status register is set to bit 6 for the device, allowing to read the status to see if the device is ready. In Figure 2.4 on the next page, the table shows the type of operation, cycling address, and the ready-busy state. The cycling is based from Figure 2.2 shown on the previous page.

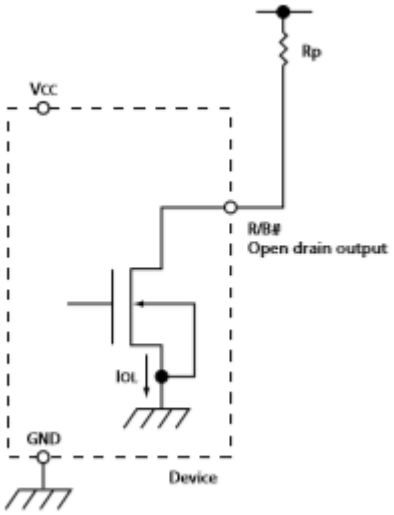


Figure 2.3: Ready Busy Circuit [4]

Operation	Cycle 1	Cycle 2	Valid During Busy
PAGE READ	00h	30h	No
PAGE READ CACHE MODE START[1]	31h	–	No
PAGE READ CACHE MODE START LAST[1]	3Fh	–	No
READ for INTERNAL DATA MOVE[2]	00h	35h	No
RANDOM DATA READ[3]	05h	E0h	No
READ ID	90h	–	No
READ STATUS	70h	–	Yes
PROGRAM PAGE	80h	10h	No
PROGRAM PAGE CACHE[1]	80h	15h	No
PROGRAM for INTERNAL DATA MOVE[2]	85h	10h	No
RANDOM DATA INPUT for PROGRAM[4]	85h	–	No
BLOCK ERASE	60h	D0h	No
RESET	FFh	–	Yes

Figure 2.4: Command List [3]

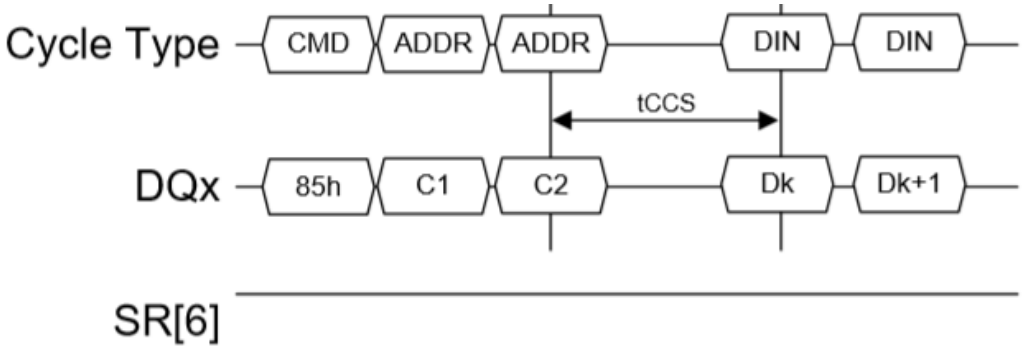


Figure 2.5: Command Sequence [5]

The command list shown in Figure 2.4 translates to the command sequences shown in Figure 2.5. Within the firmware, this is how the cycling is initiated. The cycle type takes in the command type (Read, Erase, or Program). For example, if a READ operation is issued, the internal firmware sets device address to READ and sends this information to the data interface. The SR[6] bit shows the read-busy pin. This is a very high level diagram of how the internal sequence takes place within the firmware.

Customer operations used are program, read, and erase operations. When

these operations are executed, the status register operation (read status) is of primary concern. The read status allows to monitor the NAND device to determine if data has been written and read properly. The status register bit outputs the correct status from the NAND device to see if the operation has completed correctly or resulted in failing bytes. In Figure 2.5, it is shown that the ready-busy signal signifies if the device is undergoing an operation or not. This is read out from status register SR[6], where "0" would indicate the device being busy and "1" indicating the device is ready for the next operation. Looking at an example of a customer operation, status register bit SR[0] will indicate if this was a successful program, read, or erase operation by setting this bit to "0" for failing and "1" for success. Figure 2.6 below shows each status register bit and what each bit indicates.

SR Bit	Page Program	Program Page Cache Mode	Page Read	Page Read Cache Mode	Block Erase	Definition
0	Pass/fail	Pass/fail (N)	–	–	Pass/fail	"0" = Successful PROGRAM/ERASE "1" = Error in PROGRAM/ERASE
1	–	Pass/fail (N-1)	–	–	–	"0" = Successful PROGRAM/ERASE "1" = Error in PROGRAM/ERASE
2	–	–	–	–	–	"0"
3	–	–	–	–	–	"0"
4	–	–	–	–	–	"0"
5	Ready/busy	Ready/busy[1]	Ready/busy	Ready/busy[1]	Ready/busy	"0" = Busy "1" = Ready
6	Ready/busy	Ready/busy cache[2]	Ready/busy	Ready/busy cache[2]	Ready/busy	"0" = Busy "1" = Ready
7	Write protect	Write protect	Write protect	Write protect	Write protect	"0" = Protected "1" = Not protected
[15:8]	–	–	–	–	–	"0"

Figure 2.6: Status Register SR[n] Bits [3]

The SR[1] bit will display previous failing cases. For example, if the customer runs a PROGRAM operation and gets a successful operation, the status readout will be

0x0 – all bits are programmed successfully. If the cause of the failing program operation has been corrected and the customer issues program operation again and it is successful, then the status readout will be 0x2. This is setting SR[1] bit to “1” and SR[0] bit to “0”. This means that the previous program operation failed but the current program issued is successful. Viewing Table 2.4, it can be seen there are other program page capabilities for different modes for the status register; such as, program page cache mode, page read, page cache mode, and block erase. Each SR bit is important to indicate the state of the NAND device.

2.0.4 Program Page Operation

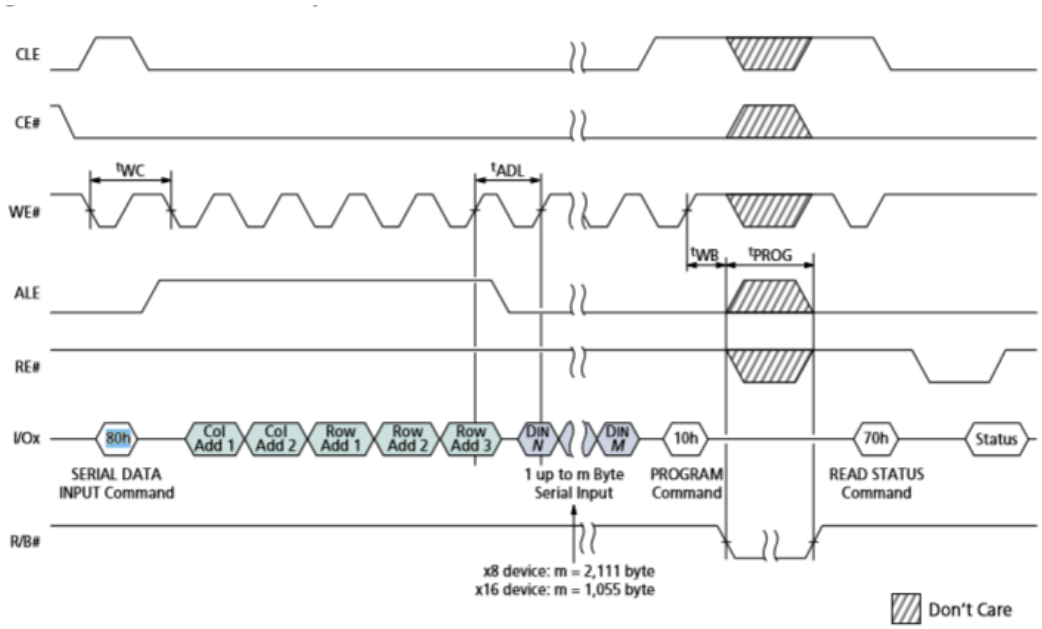


Figure 2.7: Program Page Operation [4]

The first operation examined is PROGRAM algorithm. The program operation is shown in Figure 2.7 above. The program operation programs a page of data which is given by the customer via column address for that certain page register, allowing to write data into the flash array. The pages are programmed consecutively within the blocks. Each block has a defined page address (column address) where the user can program unique data. This architecture was seen in Figure 2.1 of how the device addressing takes place and how column and row decoding take place from the I/O Control.

Looking at Figure 2.7 above, in order to issue a PROGRAM, first a power-up sequence is issued to turn the NAND device on and enter program mode. Once the chip is powered-up, an erase operation is executed on the blocks that are going to be programmed. Issuing an erase is crucial to ensure there is no data loaded onto the latches before issuing read and write operations. The PROGRAM operation is issued by the command sequence 80h-10h. The internal firmware calls for routine depending on the sequence given by the end user. Once the command sequence is interrupted by the firmware, the NAND devices enters that specific mode. In order to go into PROGRAM mode, command sequence 80h-10h is issued. This will be written from the I/O control into the command registers, followed by the 5 cycle addressing requirement (As shown in Figure 2.7).

Once the 5 cycle address is given, the user inputs the data to be written to the device. This serial data is specified by the user to a particular column address. Unique data can be programmed to all pages at once. Once the device gets programmed, it

issues a program verify to check if the data has been written correctly, resulting in the status register readout of a passing 0x0 result. Looking at Figure 2.7, we see that the R/B goes active low when the data is being programmed into the data registers at column address [N]. The time for that data to be programmed properly is given by tPROG, which is the minimum time for the data to be programmed and be latched internally. The control signals are set properly from the internal firmware so data can be written and transferred correctly. Once the data is fully programmed, read status can be issued to view the status register bit. The data can be read out using the READ operation to see if the data was latched on internally correctly.

2.0.5 Program Cache Page Operation

One other feature of program algorithm is program cache. This mode takes in command sequence 80h-15h. This feature uses buffered programming mode. It will follow the same behavior by taking the I/O serial input command 80h to command register with the 5 cycling address requirement. However, now this data is stored into the cache register, hence the 15h cache write command sequence. These cache registers are shown in Figure 2.1 in the architecture for the NAND device. The data is then latched internally and transferred from cache register to data registers depending on how the control signals are set (WE, R/B, etc). Ignoring the control signals for now, looking at Figure 2.8 below, the sequence 80h is written to the command register followed by the addressing cycle, ending with the 15h cache write command. It is seen that there are two times the R/B pin goes active low. There is a time-frame where the data is

being written and enters a buffered programming state. It writes the data and waits for the next command sequence. To verify if it has been written, we issue the read status command sequence (70h) to look at the status register bit. As said before, we can view the status register bit to see if the write was successful. Note that whenever R/B is high in any algorithm, that is the only time data can be written, read, or erased. No algorithm or sequence can be run if the R/B is active low.

Figure 2.8: Program Cache Page Operation [4]

2.0.6 Read Page Operation

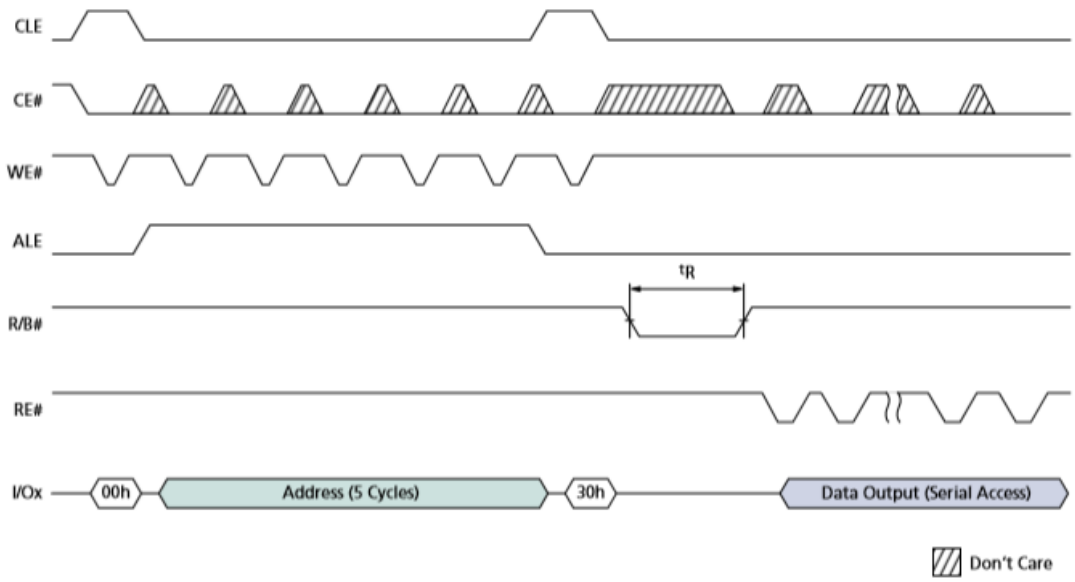


Figure 2.9: Read Page Operation [4]

The next operation is READ operation as shown in Figure 2.9 above. As mentioned earlier, a power-up sequence for the device is issued and then command sequence is issued to enter READ mode. This is issued by the command sequence 00-30h (As shown in Figure 2.4). This sequence 00-30h will be written to the command register with the five address cycling requirement. The 00h command sets the data address and is followed by a 5 address cycle. The column address and page that is to be read from the device is issued by the user. Followed by the 5 address cycle, command 30h is executed. Once 30h is written to the command register, the page data is transferred to the data register. This is when R/B goes active low during the transfer and tells that the device is undergoing an operation. Once the operation is completed, the R/B pin will go active high and will become active low when the READ operation is executed.

The tR is the minimum time the ready-busy pin goes active low. Looking at Figure 2.9, it is seen that the control signals are being set properly within the firmware logic before user inputs any logical operations. One important note here is that there are other read flavors; such as, page read cache mode. The purpose of this is to use the cache registers within the NAND Flash Device to decrease the tR time to read a page. The sequence that is inputted is different than the regular READ operation.

2.0.7 Erase Block Operation

The next algorithm to cover is ERASE. Block erase will erase start block to end block specified by the user for that certain logical unit. This operation is important

before programming any data to the NAND device to make sure the data being read is not corrupted. A successful block erase will result in SR[0] to return a passing condition, for example 0x0. The block erase logic is shown in Figure 2.10 on the next page. As mentioned before, each block has a total of (1k + 32) words X 64 pages, which amounts to total of (64 + 2k) words that are being erased. When a block erase is issued, the firmware routine is called and erases each block consecutively.

Unlike PROGRAM and READ operations, the block erase algorithm only requires the command sequence 60h-D0H written to the command registers. This command sequence is followed by a 3 address input cycle (instead of 5) which is written to the device. The D0h command sequence is written to the command register that sets the firmware call to be in erase mode. During this process, the R/B signal will become active low during the erase process. The minimum time that takes for an erase algorithm to complete is defined by tBERS. In Figure 2.10, it is seen that the I/O takes in the 60h-D0h erase confirmation command sequence from I/O to command registers with the given block information. It is followed by a 70h command sequence to verify the status register bit to see if erase operation has completed successfully. SR[6] identifies if the erase operation is complete and SR[0] identifies if the condition has passed or failed. The control signals are set internally for this transfer to occur properly. If there is a failure, the erase operation is re-issued on same block and the SR[N] bit is updated.

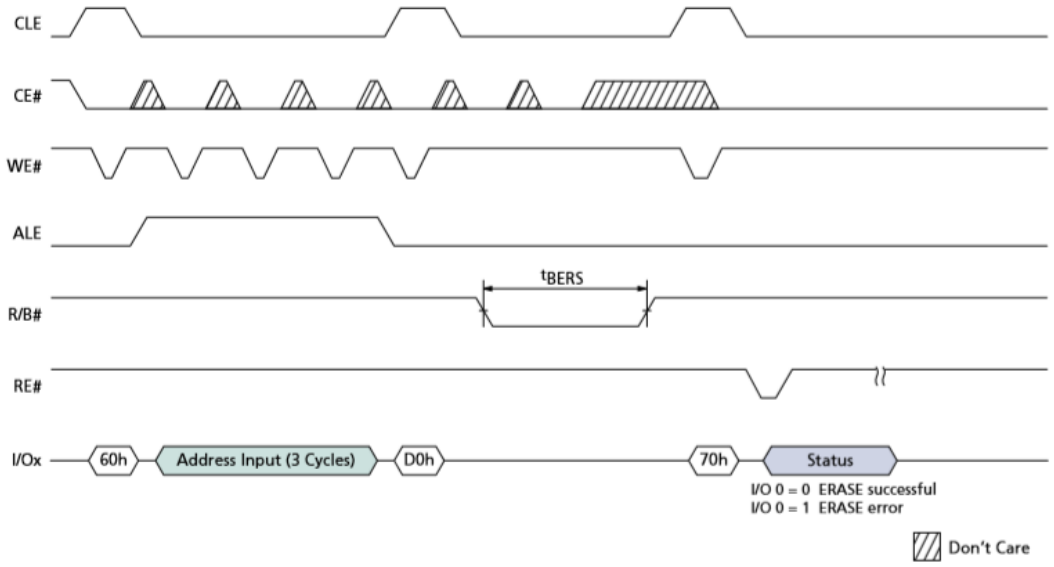


Figure 2.10: Block Erase Operation [4]

2.0.8 Reset Operation

Another important customer command is the RESET operation shown in Figure 2.11 below. This is useful when an algorithm is issued to the device and needs to be aborted by the user. The command sequence for this is given via FFh. Program, Read, and Erase operations can be aborted by issuing reset FFh command sequence to the device. Internally, the firmware will abort the operation and the device will no longer be in busy state. This will clear out the command register information and the device will be waiting for the next command to be issued. The R/B signal will become active high after the sequence is aborted. This can be seen in Figure 2.12. When the device is not executing any algorithm, the output of status read (70h) will be 0x0h.

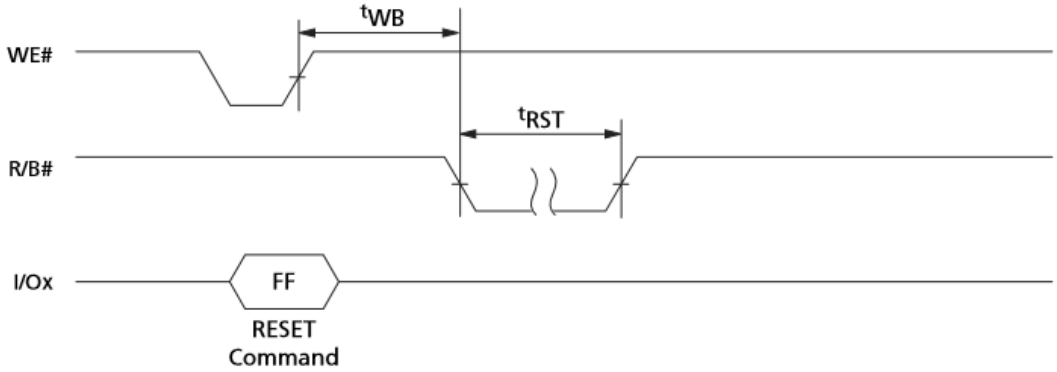


Figure 2.11: Reset Operation [4]

Condition	Status	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Hex
WP# HIGH	Ready	1	1	1	0	0	0	0	0	E0h
WP# LOW	Ready and write protected	0	1	1	0	0	0	0	0	60h

Figure 2.12: Status Register Bit for Reset [4]

Chapter 3

Research Environment

3.0.1 Testing Analysis Background

The laboratory testing and experiments were done within the NAND Lab Solutions Group at Intel Corporation. Post-silicon testing and experiments are done by developing methodical experiments and running temperature controlled experiments on silicon level wafers to understand the transient behavior of the NAND Flash memory. Extreme testing was done at wafer level to fully analyze if internal firmware logic is behaving optimal and customer commands and many sub-features of the READ, PROGRAM, and ERASE algorithms are working properly. Prior to lab-work, analyzing the firmware, running simulations, and learning the NAND device physics are done. Understanding the complete design of how the circuitry logic is behaving pre-silicon vs post-silicon is very important and we expect slight changes since we are looking at the behavior from a simulated environment to a hardware environment.

3.0.2 Bench Setup (Prior steps before testing)

The test setup used is a low-power engineering test system that is used for semiconductor research and development. The DUT (Device Under Test) in this case is one silicon wafer. Depending on the die size, a silicon wafer is capable of having more than 400 die (Silicon Chips) that it can hold. A Teradyne Magnum tester is used to run tests at high efficiency for non-volatile memories (NAND Flash). The tester itself provides 5120 digital channels and provides a 50Mhz pattern rate and 100Mbps DDR rate on its channels. The tester has the capability of analyzing more than 320 NAND Flash devices. The probing station that is used to micro-probe the silicon wafer is by Cascade Summit. This tool is a high precision probe system that allows testing of the wafer and being able to probe desired signals at various temperatures.

This capability allows to further analyze for a broad temperature range from -60°C to 300°C, which is important when looking at the engineering design of the NAND Flash for extreme testing. The experiments that were conducted were done at 90°C, 25°C, and 0°C. One other important device is the probe-card. This is the card that the silicon wafer is positioned on, making a direct contact with the I/O of the NAND device. This probe-card is designed to deliver optimal testing for flash memory with one-touchdown probing and has efficient electrical testing capabilities. A detailed explanation of the test setup is shown in the next section.

3.0.3 Semiconductor Micro-Probe Station

The testing environment of the lab setup is shown in Figure 3.1 on the next page. This includes the Teradyne Magnum tester, Cascade Summit micro-probe station with probe-card, oscilloscope, spectrum analyzer, and the host CPU itself. The host CPU itself is where firmware logic and macros to address certain registers to alter the behavior of the output signals are run. The logical behavior for our testing conditions is written in Perl and C++. Test macros are defined scripts that are written in assembly language that allow to address registers to change the functionality of the circuit.

There is a temperature control unit that is connected in the back of the tester, allowing the capability to conduct testing from -60°C to 300°C. The spectrum analyzer tool is used to characterize the I/V characteristics of the device in order to understand any issues or leakages within the circuitry. The spectrum analyzer is capable of forcing current loads to a specific point within the circuit and thus stress the silicon pad to see the voltage output behavior. Precise user-defined voltage steps can be initialized to understand the transient behavior output.

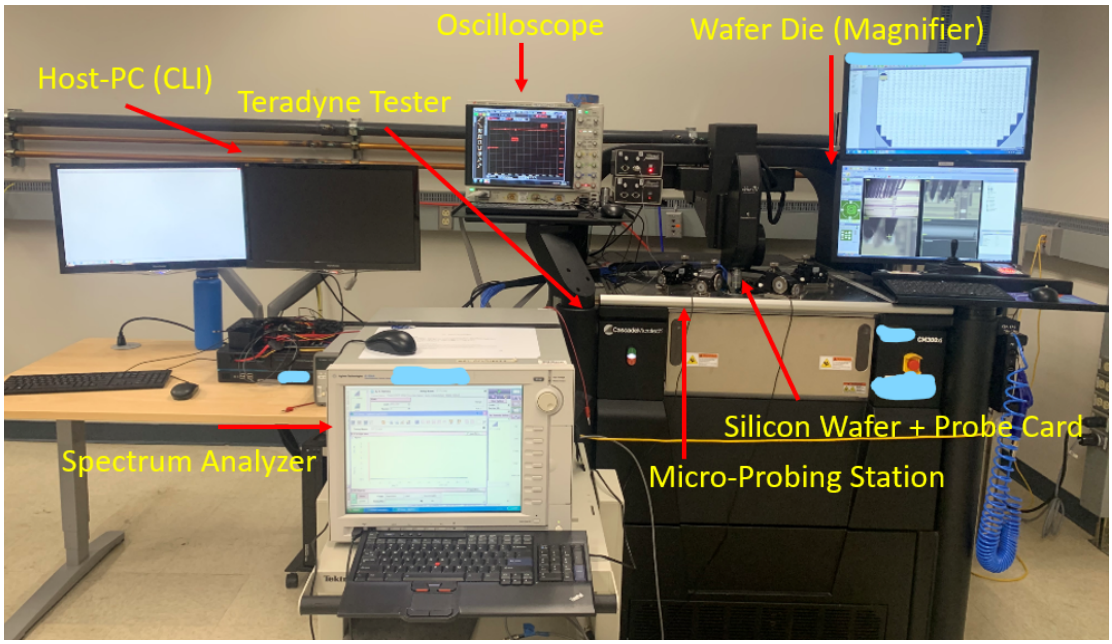


Figure 3.1: Tester Setup

The complete station is shown in Figure 3.1 above. The host PC has its own command line interface (CLI) that takes in user input that gets sent to the Teradyne Magnum tester. The tester interprets this data and allows to have direct communication with the probe-card. The probe-card holds the silicon wafer and connects it electrically. The probe-card station also has four probing grips. These grips allow to land a probe to the desired probe point within the wafer. The probe grip allows to control X,Y,Z direction of probe tip so it is landed properly within the probe point of the wafer. This probing mechanism is shown in Figure 3.2 below.

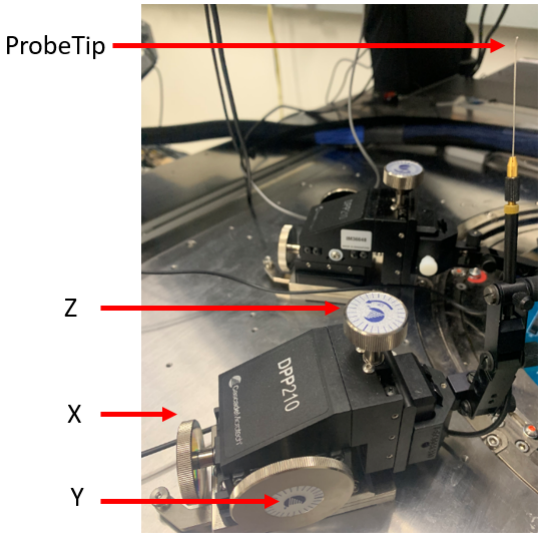
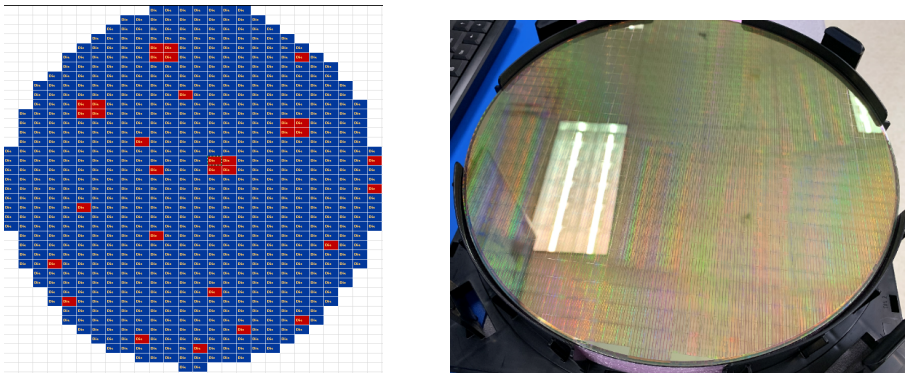


Figure 3.2: Probe Landing Mechanisms

One end of the probe tip is in contact with the wafer at the desired probe point. The other end of the probe is connected to the oscilloscope. Once the silicon is electrically connected, transient analysis can be understood for the NAND device. An example of a test setup would be placing the wafer onto the probe-card properly to ensure electrical contact with the die (LUN). Once contact has been made, a parameter check is issued to see if the probe-card is contacted with the silicon wafer properly. If this parameter check fails, then the contact with the probe-card and silicon wafer are not aligned, resulting in chip initiation fail. The chip initiation verifies that the control signals (CLE, ALE, CE) are set accordingly and verifies that the user input I/O is fully capable of data transfer.

Once this is verified, a power-up sequence is issued from the command line



(a) NAND Wafermap (b) Silicon Wafer

Figure 3.3: NAND Semiconductor Wafermap (Left) and Silicon Wafer (Right)

interface (CLI) host PC that transmits a command to the Teradyne Magnum tester. This information gets sent to the probe card with the silicon wafer and sets the operating voltage for the device. As mentioned before, the status register bit readout will determine to see if electrical contact is correct and if the device is powered on properly. If the SR bit returns correctly, then it is safe to issue NAND operations.

Once the set-up is configured and the silicon wafer is electrically connected to the probe card, the LUN (Die) needs to be verified. As said before, there are about 400 dies per wafer (LUNs). The silicon wafer used in lab is shown in Figure 3.3 above. Each die on the wafer is independent of one another, there could be chances where some die are corrupted and no data can be written or read from the device. Looking at Figure 3.3a above, a wafer-map is generated for each silicon wafer to define a base layer of which die is being tested. This correlates to the actual silicon wafer that is seen to the right of it in Figure 3.3b. The wafer-map shown in Figure 3.3a is marked with good and bad dies. As seen, the die in blue represent that the die has passed rigorous testing and

the die marked in red show that this die may be functional but there are some tests it has failed from the fabrication lab. There could be higher concentration of leakages that may disrupt the behavior of running algorithms on the device, resulting in corrupted data. There can be many factors and reasons for certain fails to consider a die "bad". The bad die are shown in the color red, which will have certain labeling associated with it that defines which tests it passed and which tests it failed in the fabrication lab.

3.0.4 Probing Channel Setup

Specifications	Model 12C	Model 18C	Model 28	Model 29	Model 34	Model 35
Input Capacitance	0.1 pF	0.02 pF	0.04 pF	0.04 pF	0.1 pF	0.05 pF
Input Resistance	1.0 megOhms	10 femtoAmps	10 femtoamps	10 femtoAmps	10 megOhms	1.25 megOhms
Rise/Fall Time	0.8 nS	1.2 nS	350 pS	350 pS	120 pS	14 pS
Frequency Response	dc to 500 MHz	dc to 350 MHz	dc to 1 GHz	dc to 1 GHz	dc to 3 GHz	dc to 26 GHz
Operating Range	-10V to 20 V	0 to 15V	0 to 9 V	-7 V to 2 V	-7 V to 10 V	-6 V to 6 V
Linearity	0.5%	0.2% (5V - 15V)	0.5%(0V - 4V) 2%(0 - 9V)	0.5%(-2V - 2V) 2%(-7 - 2V)	0.5%	2%
Gain Accuracy	3%	5%	3%	3%		
Signal Attenuation	High Impedance 20 to 1 50 Ohm input 40 to 1	High Impedance 10 to 1 50 Ohm input 20 to 1	20 to 1 (50 Ohm)	20 to 1 (50 Ohm)	20 to 1 (50 Ohm)	10 to 1 (50 Ohm)
Active Probe	12C-4-10-1.8MegOhm	18C-4-10HV	28-5-10	28-5-10	34A-4-10	35-1-22-20-30

Figure 3.4: Probe Models

Referring back to Figure 3.2 regarding the probe landing mechanisms, each probe tip provides different capabilities for testing; such as, input impedance, input capacitance, input leakage, frequency response, and voltage operating range which is shown in Figure 3.4 [1] above. With extremely low input capacitance, high input impedance, it is possible of capturing the best performance from the silicon wafer and are able to characterize the circuits with high efficiency. These capabilities listed are put into consideration when it comes to analyzing silicon data. The probe-tip models

used are 12C, 18C, 28C, 29C, 34C, 35C, and passive probes. Each probe has its own purposes and functionalities. For example, if we are interested in capturing the output signal of an oscillator circuit with a frequency that outputs in the Mhz region, model 18C+ probes would be more suited for this analysis. 18C is primarily used when dealing with a fast ramping signal. One other important factor is the maximum operating range of the signal. If the signal to be micro-probed outputs 15V, 28C model cannot be used anymore since it would cap off at 9V. In this case, it is wise to use the 18C for analysis.

If fast ramping signals and where timing analysis is of no concern, then passive probes are used. A passive probe offers reasonable signal integrity to capture the output behavior. Before any experiment is run, all active probes need to be calibrated. This is done by micro-probing the VCC supply of the silicon wafer and measuring the output voltage on oscilloscope. The VCC supplied to the NAND device is set to the operating range of 2.5V to 3.6V. If the VCC supply is set to 3.0V, this voltage is expected to be measured on silicon as well. If this is not the case and a delta is observed, then calibration is needed. The power supply probe is set to the same voltage as the operating range of the device. The ground floor is also calibrated to remove any noise. However there will be internal noise in the range of 50-70mV, therefore an offset of 50-70mV is considered fine. Calibration is important otherwise the signal will have an additional delta in our output analysis. If the ground floor is not calibrated to be around 0mV and is around 400mV instead, a typical signal that should output 6V will show around 6.4V for example. This work will further be described in Section 4.0.2.6.

Chapter 4

Integration

4.0.1 Integration

Design tests and development were created in order to solve engineering challenges to output higher performance. In any engineering design, the goal is to optimize performance at low cost. The goal in any semiconductor design is to reduce yield loss; therefore, there will always be engineering trade-offs made in order to achieve higher output performance for solid-state drives.

Development and analysis will be presented below regarding the analysis of the NAND device. The 3D NAND design itself is a complex device where the analysis of experimentation has to be broken in various sub-tasks to fully understand the behavior and to properly develop a simulation-to-silicon correlation. The analysis developed will allow to better understand how internal data is latched. This will cover all algorithm functionality (READ, PROGRAM, ERASE).

4.0.2 Integration Flow

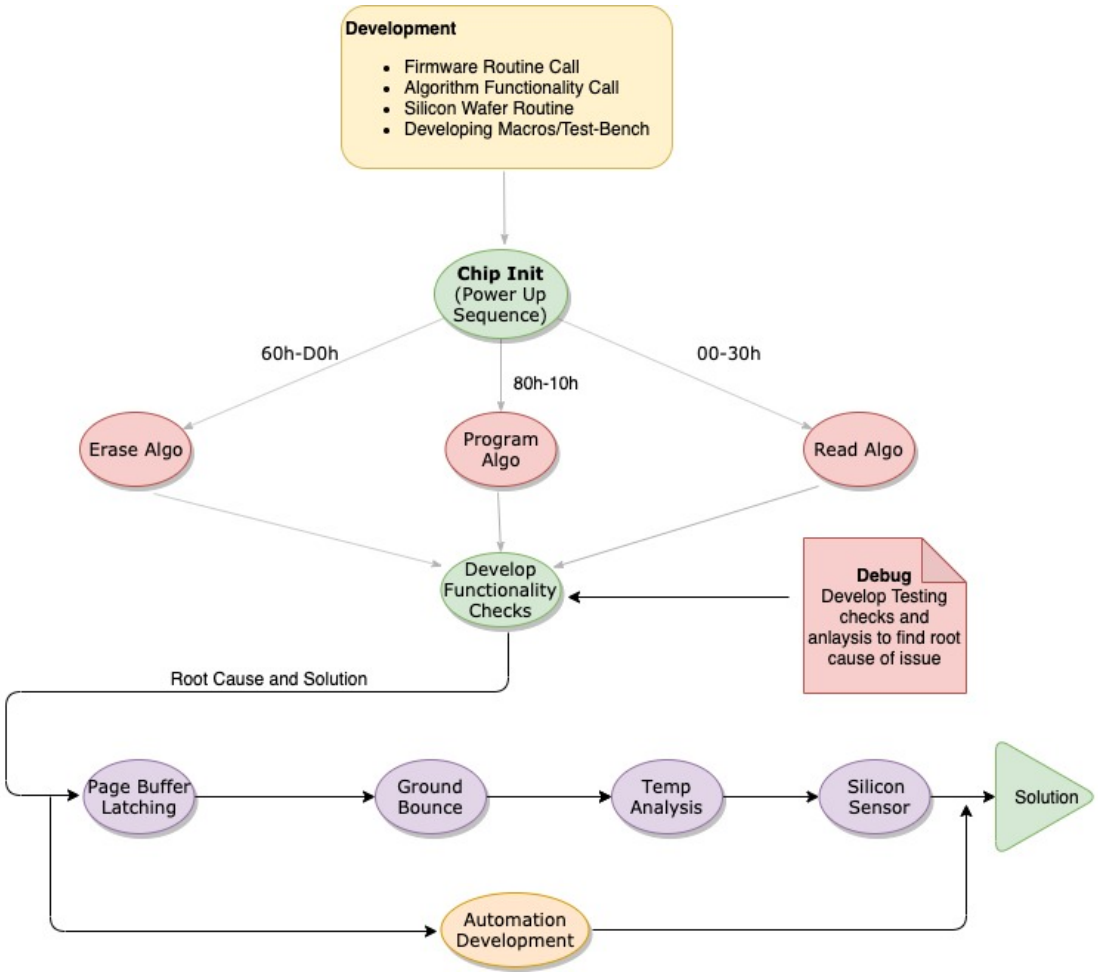


Figure 4.1: Command Sequence

The flow shown in the Figure 4.1 above introduces the development and research process of the NAND device that is carried out. The initial motivation is to research how the NAND device is capable of addressing unique data and latching this data internally by calling the NAND firmware logic. Firmware and test-bench logic were developed to analyze the page buffer latching analysis.

Initially, development was implemented regarding the internal firmware logic and how to process this on actual silicon wafer. NAND page buffer data latching was the objective. In other words, how data is latched internally from the user I/O[7:0] pins to the data and cache registers. The transfer of data uses DDR data transfer mode. In this case, the device uses NV-DDR, NV-DDR2, and NV-DDR3 data interfaces.

Proceeding to analyze internal data flow by analyzing the algorithms, there were other aspects of tests that were developed to further research the issue of failing bytes seen in the internal latching. This included developing understanding of ground bounce, temperature analysis, and silicon contact sensor analysis. Automation was another add-on of this research, to automate the hardware to get better analysis of the NAND device.

4.0.2.1 Page Buffer Data Latching

When data is transferred from I/O[7:0] to cache registers, the data is taken from the host as a unique pattern that is latched to the latches internally. When the internal circuitry is set, this data gets captured and transferred through the latches and gets stored within data and cache registers. The timing mode is important to set when data transfer is occurring, therefore the device has different timing modes for the host to select before data transfer occurs. The data is latched at the rising edge of the clock cycle. The data transfer is shown in Figure 4.2 on the next page.

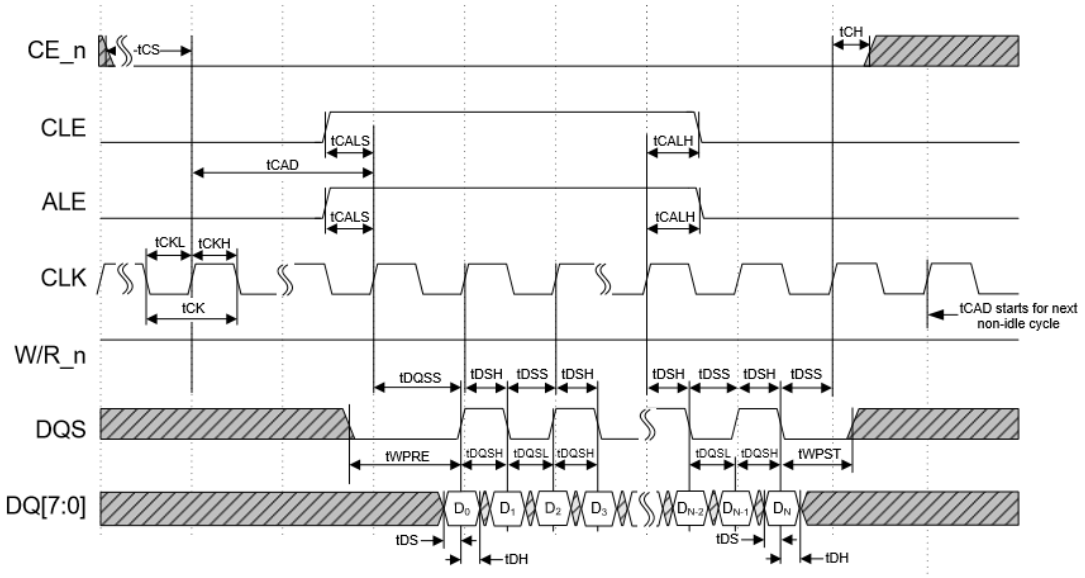


Figure 4.2: Data cycle Latching [5]

Looking at the figure above, the ALE and CLE signals are active high only when the data is being transferred from DQS. The DQS is the DQ bus where the data is transferred from the device to the host. The latching edge of the data cycle is latched during the rising and falling edges of DQS transmission. The internal firmware and analog control logic are set appropriately to take care of this logical behavior.

4.0.2.2 Page Buffer Chip Initialization

The implementation methodology and testing analysis is carried out by using the laboratory hardware setup previously shown in Chapter 3. The initial step is to have the silicon wafer secured within the micro-probing station. Initial power-up sequence is issued (PU;PD) followed by a status register readout [SR] bit to see if the probe-card has direct metal contact with one LUN (die). VCC must be set accordingly, which

```
1  #Power Up Sequence
2  pd;
3  pu;
4  #Status Register Readout
5  #Read Status 70h
6  rs;
7  #Output
8    > CLI Readout: NAND Device SR[N]: 0x0
9  #Supply VCC to the Device
10 (Set_VCC, 3.0)
11 VCC_Readout
12   > CLI Readout: VCC: 3.3V
```

Figure 4.3: Chip Initialization

is supplied from the tester to the the NAND device to establish a reference voltage, this is done on chip initialization. VCC will vary for certain experiments depending on customer use and the datasheet specifications defined. Powering up the device is the first command issued, this is done by issuing the “pd;pu“ sequence shown in Figure 4.3 above. This process is done prior to all testing and development. In line 6 of the chip initialization, if the SR bit is 0x1, then we would have to move to a different LUN and look at the SR bit to see if this die is functional.

4.0.2.3 Page Buffer Block Erase Operation

Once the chip initialization has been configured, the data transferred is checked to determine if it has latched correctly by running erase, program, and read operations. The high level firmware logic that is developed for the block erase algorithm that the firmware interpreter traverses is shown below in Algorithm 1 below. The first process is to issue a block erase algorithm to make sure no data is stored within the data and cache registers. If no erase is issued, there will be corruption causing the bits to be shifted when writing or reading from the device.

Algorithm 1 Block Erase Algorithm

```
1:  procedure BLOCK_ERASE(block_start,block_end)
2:      Command_Address() ← 60h-D0h to enter block erase mode
3:      for (start_block=0; i <= end_block; i++) do
4:          if (start_byte && end_byte != 0xFF) then
5:              block_erase.append()← BLOCK[n] to BLOCK[n+1]
6:              block_erase.statusReadout() ← PASS or FAIL
7:              tBERS_erase_time ← delay
8:          else
9:              block_erase.statusReadout() ← PASS or FAIL
10:             tBERS_erase_time ← delay
11:         end if
12:     end for
13: end procedure
```

```
1 rs;
2   > CLI Readout: NAND Device SR[n] readout: 0x0
3 #Block Erase 60h-D0h
4 block_erase($block_start,$block_end)
5 block_erase(0,500)
6 #If there is a PASS:
7   > CLI Readout: Block Erase readout: 0x0
8 #If there is a FAIL:
9  > CLI Readout: Block Erase readout: 0x1
10       Block - Expected Byte - Stored Byte  - Failing Bytes
11        1          0xFF .          0xA2             4
12        32         0xFF            0x01             10
13  >Total Failing Bytes: 14
```

Figure 4.4: Page Buffer Block Erase Operation

In Figure 4.4 shown above, RS is first issued to check the SR bit to see if the device is functional. Then block erase is issued on the NAND Flash Device. Erase functionality must be verified before executing program and read operations to determine if the data is all 0xFF (Erase). In the code snippet above, both examples are run of a passing and failing case. The first case is we issue a Block Erase Perl script on block 0 to 500 consecutively.

Block Erase is firmware logic that addresses the control and the data registers. The arguments taken are start block and end block. It is seen that the first scenario seen in lines 5-7 show that the block has been completely erased. If the block was not

erased properly, we would see output lines 8-13, as shown in Figure 4.4. Here we are told by the NAND device that block 1 and 32 did not get erased properly out of the 501 blocks we issued to erase. The expected byte is 0xFF but what is seen is block 1 and 32 have data values of 0xA2 and 0x01. There are a total of 14 bytes failing.

When failing bytes are seen, the root cause is determined of why the device is not functioning properly. Methods of debugging this issue are to try a different die on the silicon wafer. As mentioned before, there is a high chance that the LUN can be simply a bad logical unit. If this issue is seen across multiple LUNs, then the process is to run full-wafer testing to see if this issue can be replicated. Engineering methods of finding the root cause of the problem are developed. If the problem is replicated on other LUNs, then there are protocols for a lower level debug. This would mean to understand the firmware, datapath, and circuitry side of what can cause this internally. There can be a fabrication issue also but this is out of our hands at this point and would require deeper analysis to be reviewed for the next silicon wafer release.

4.0.2.4 Page Buffer Program Page Operation

Algorithm 2 Program Page Algorithm

```
1: procedure PROGRAM_ARRAY(start_page,end_page,start_byte,end_byte)
2:     Command_Address() ← 80h-10h to enter program page mode
3:     for (start_page=0; i <= end_page; i++) do
4:         program.sByte() ← start_byte
5:         program.eByte() ← end_byte
6:         program_array.statusReadout() ← PASS or FAIL
7:         tPROG_array_time ← delay
8:     end for
9: end procedure
```

Once an erase is completed, the next portion of the page buffer analysis is to properly write to the device. This is done by using the program algorithm shown in Algorithm 2. This operation is done via command 80-10h which is called within the program array routine. When the program algorithm function is called, the firmware calls the routine for the NAND device and takes in the data from user. In this instance, a data pattern is loaded as an argument for that particular column address. The bench script is created so it addresses each column address for a certain page register and have it properly written to the flash array.

The command analysis on bench is shown in Figure 4.5. Both pass and fail case scenarios are shown. In order to program the page properly, starting and ending

page and the number of bytes to be programmed must be issued. Lines 6-10 show a passing case and lines 11-16 show a failing case.

```
1  #Erase the blocks to be programmed so no corrupted data exists.
2  #Block Erase 60h-D0h
3  block_erase(0,500)
4  #Program Page 80h-10h
5  #Program_array calls internal firmware and we manually set the command
       80-10h to issue program algorithm.
6  program_array($start_page,$end_page,$start_byte,$end_byte,$data)
7  #Programming device up to only 1501 bytes for page 0 to 50.
8  #Pass Case:
9  program_array(0,1500,0,50,0xBB)
10   > CLI Readout: Program Page readout: 0x0
11 #Fail Case:
12   > CLI Readout:
13       >Start Page - End Page - Expected Data - Stored Data
14       >    0           2           0xBB             0xAC
15       >    4           6           0xBB             0xB1
16       >Total Failing Bytes: 6
```

Figure 4.5: Page Buffer Program Page Operation

4.0.2.5 Page Buffer Read Page Operation

Once the unique pattern has been properly programmed to the latches and the page is programmed, we verify by reading the data out of the cache registers to see if it was latched properly to the given column address (page) specified. Algorithm 3 shows the read page operation. The command sequence issued is 00-30H. This sequence will be written to the command register with the supplied column address. This column address is the same where the data was initially written to. The sequence to carry out this operation is seen in the figure on next page.

Algorithm 3 Read Page Algorithm

```
1: procedure READ_ARRAY(start_page,end_page,start_byte,end_byte,data)
2:     Command_Address() ← 00h-30h to enter read page mode
3:     for (start_page=0; i <= end_page; i++) do
4:         read.sPage() ← start_page
5:         program.ePage() ← end_page
6:         read_array.statusReadout() ← PASS or FAIL
7:         tRD_array_time ← delay
8:     end for
9: end procedure
```

As seen in the logic above, **read_array** is issued. This writes command 00h-30h to the command registers and calls the READ firmware routine. Once the NAND device knows it is in READ mode, it is able to read that set column address by the

given start and end page. The algorithm checks if the data that is written to the latches matches what is read from the registers. If the data read is corrupted, the number of failing bytes and in which column address this is corruption is happening will be displayed. The bench sequence is shown in Fiure 4.6 below.

```
#Read_array calls internal firmware and we manually set the command 00h
    -30h to issue read algorithm.
#Read 00h-30h
read_array($start_page,$end_page,$start_byte,$end_byte,$data)
#Read device for programmed bytes only.
#Pass Case:
read_array(0,1500,0,50)
  > CLI Readout: Read Page readout: 0x0
  > CLI Readout: No Failing Bytes
#Fail Case:
  > CLI Readout:
      >Start Page - End Page - Expected Data - Stored Data
      >    0           9           0xBB             0x09
      >Total Failing Pages: 10
```

Figure 4.6: Page Buffer Read Page Operation

4.0.2.6 Ground Bounce Mitigation

Ground bounce phenomenon is researched in order to develop an understanding of the failing bytes seen on certain pages when reading the data out of the latches. This also creates a concern on how to minimize this externally via the probes used in testing. Since this is actual silicon hardware, there is no capability of making design edits to minimize this. Ground bounce behavior helps in understanding why failing bytes are seen when data is programmed and read from the internal latches. Ground bounce is a concern since this causes deviation of the results and analysis that is being researched. Shown in figure 4.7 below, running any algorithm causes this bounce. The probes are landed on various ground pockets of the NAND device.

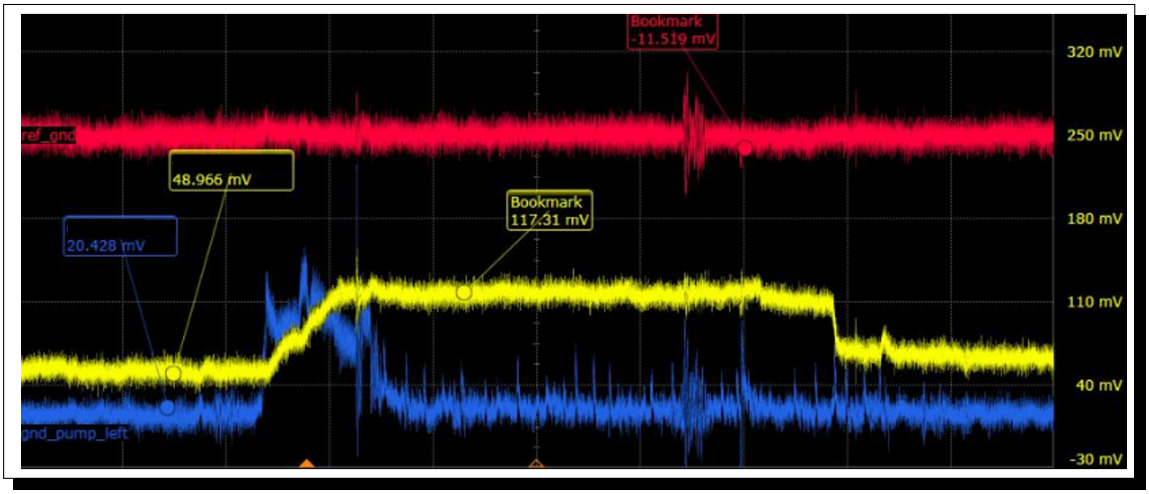


Figure 4.7: Data Latching Ground Bounce @ 0.1pF Input Cap

Ground bounce can impact obtaining high performance results, therefore it must be reduced as much as possible within the NAND device. Erase, program, and

read algorithms are run to understand how much ground bounce is occurring due to intrinsic electrical characteristics and the deviation from expected results. A metric is created of the voltage from the wave-forms of each algorithm from die-to-die. From experiments, it was seen that Program, Read, and Erase algorithms have fast ramping times. Due to their advanced logic, they introduce more of this ground bounce scenario. As referenced in Chapter 3, picoprobes that are low-parasitic probes were used as shown previously in Figure 3.5. The input capacitance can be reduced as low as 0.05pF. It was noticed that this mitigated the output ground bounce and external noise. It is seen that the ground bounce reaches up to 118mV (Yellow Waveform). Looking at the blue waveform, the ground bounce was reduced to 20mV. The red waveform shows a ground in a different region of silicon where the ground bounce is around 12mV.

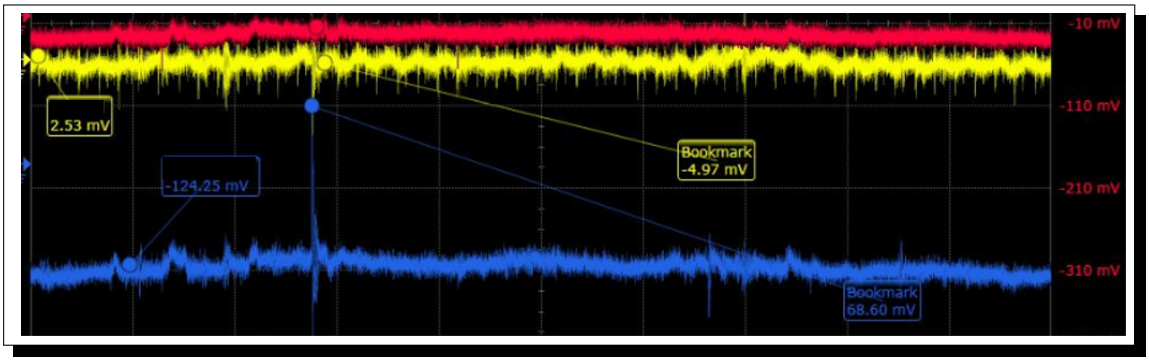


Figure 4.8: Data Latching Ground Bounce @ 0.004pF Input Cap

Mitigating and applying the correct model for the picoprobe usage during testing, the input capacitance was reduced from 0.1pF to 0.005pF. With this, it was seen a reduction in ground bounce can be achieved slightly. This is shown in Figure 4.8

above. From this analysis, a reduction in the ground bounce and external noise can be achieved to have a more stabilized output.

4.0.2.7 Temperature Analysis

Temperature impact is the next area of analysis to conduct when running Erase, Program, and Read algorithms on the NAND device. Behavior of exercising various algorithms under different temperatures to see the data retention of the NAND device gives better insight of how the data retention trend behaves and gives insight on endurance. Screen tests are run from 0°C to 90°C to collect a trend of failing and passing cases. The floating-gate transistor in this case that is used as the main building block will cause the cell voltage threshold to shift when an algorithm is run. The cell voltage threshold will shift at different temperature gradients.

Viewing the floating-gate transistor NAND Memory cell at a high level in Figure 4.9, where a voltage that is greater than the threshold voltage is applied to the gate of the transistor, it will turn on and electrons will flow from from oxide to gate.

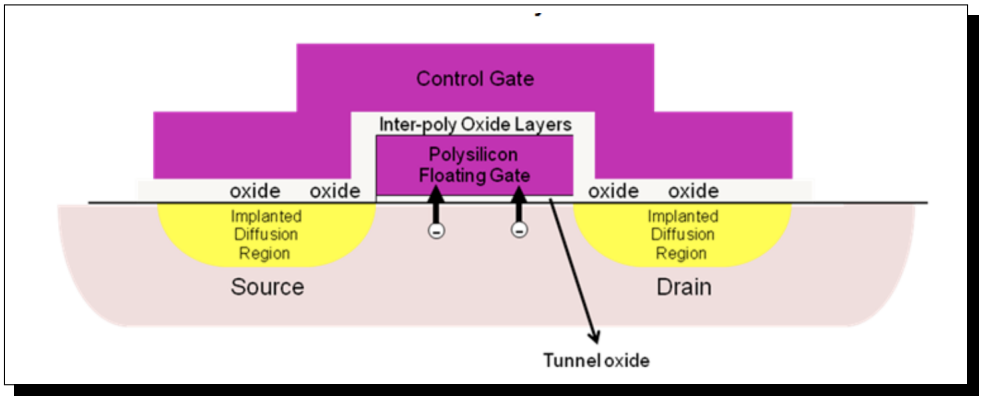


Figure 4.9: NAND Memory Cell [6]

To erase a cell, the substrate well is raised to a high voltage forcing electrons back through the oxide layer from the floating gate into the substrate. To read a NAND cell, a voltage above Vt is applied to the top gate and the current flowing in the transistor is sensed by a sense amplifier, which gives information about the amount of charge stored in the cell [4].

NAND Flash Array Temperature Characterization: Read Operation			
Temp	Fail Bytes	Expected Bytes	Read Bytes
90°C	4	0xAA	0xAB
75°C	3	0xAA	0x1F
25°C	2	0xAA	0xAC
0°C	19	0xAA	0xA0

Figure 4.10: Temperature Failure Analysis

Therefore this analysis is conducted to understand how much of a temperature shift can cause certain bytes to flip and fail during a Erase, Program, Read operation. Theoretically and mathematically, it is known to expect higher failures at colder temperatures. Development of deviation of failing bytes is shown above in Figure 4.10. To minimize corrupted data, all blocks are erased and 0xAA is loaded to the latches.

Expected data programmed to latches is 0xAA. It is seen that there are failing bytes as temperature is decreased which decreases data retention. This is expected since the floating-gate transistor cell voltage shifts. Typical ranges used to implement

analysis is at 90°C, 25°C, and 0°C. The amount of failing bytes seen at 0°C is around 19 bytes. This capture gives insight on understanding the failure analysis of why the bits are shifted and to further analyze the logical and circuitry to limit read fails.

4.0.2.8 Silicon Sensor

Algorithm 4 Defective Device Check

```
1:  procedure DIE_CHECK(Defective_ID,Start_LUN,End_LUN)
2:      Defective_ID ← False
3:      for (Start_LUN=0; i <= End_LUN; i++) do
4:          ShortCheck.Die()← Open and Short Check
5:          LUN.statusReadout() ← PASS or FAIL
6:          if (LUN.statsReadout() == 0x1) then
7:              (Defective_ID) ← Fail.
8:          else
9:              (Defective_ID) ← Pass.
10:         end if
11:     end for
12: end procedure
```

Silicon sensor characterization was developed to verify if the silicon wafer has any underlying defects. Even though the parameter check (Explained in Chapter 2) shows that the LUN is electrically connected, the LUN itself could be physically damaged, causing not to properly implement Erase, Read, and Program operations due to

shorts or opens.

In Algorithm 4, a firmware routine is created to test at wafer level to develop an understanding of the resistance of the material to determine if there are any open or shorts which gives insight on the fail signatures. The material can be connected properly but after some time the material can degrade and cause the endurance to drop after multiple cycles of operations have been issued. There could also be material issues that cause the device to fail and return failing bytes.

On startup, the device is assumed to have no failures. Each LUN is iterated and checked for a short and open resistance check. The voltage will be computed and compared to a reference setting supplied. A status readout will be issued if the compared voltage is matched with what is expected. If there is a failure and misalignment, the status readout will list 0x1 (Fail) otherwise 0x00 for a passing condition that the LUN is not defective. The ID will be updated and each logical unit will be checked. This gives insight that the failure seen during the read operation can be also occur due to a defective device.

4.0.3 Automation Development

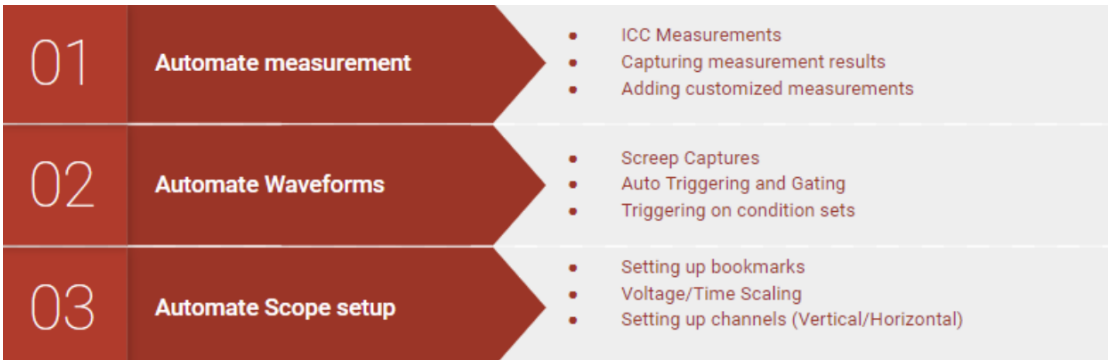


Figure 4.11: Automation Flow

Automation was another aspect of this work that was done in parallel to get better performance results. This was to introduce a higher throughput by automating oscilloscope data capture. The purpose of automating scope data collection and measurement is to become more efficient, accurate, and reduce the time of conducting research via using the existing Keysight hardware and API with customized scripts to have direct communication with the Keysight Oscilloscope and the semiconductor testers.

In the process above in Chapter 2 and 3, analyzing data and running various tests can result as a time constraint. Once the data is captured, it will take time to understand and analyze that data to do understand the underlying engineering problem. The motive of this is to automate the process via customized environment using the Keysight Oscilloscope using built-in API functions. This allows us to capture waveforms and extract data at a much faster rate.

4.0.3.1 Automation Hardware Connectivity

Figure 4.12: Hardware Connectivity

The setup requires a USB A/B 2.0 cable connecting from the MAGEV Tester (PC-CLI Communication) to the Keysight Oscilloscope. USB interface connectivity is first set up using the USB A/B from PC to Keysight Oscilloscope. This interface is called within a function from script. This has a capability of USB 2.0 and USB 3.0 super-speed port [2].

The required software must be installed on the tester PC. The IO Libraries suite 2019 allows user to run built-in APIs with the Keysight Oscilloscope, allowing to set up a direct connection. The image is shown above in Figure 4.12. Once a connection has been made via USB A/B, a connection string VISA Address will appear with SICL address. This is important for the USB interface connection function that is called within our custom Perl script. The set-up script below establishs a communication between host PC and oscilloscope shown in next section.

4.0.3.2 Automation Software Connectivity

Once there is a working communication link between Host PC and Oscilloscope, Lab-Visa-3.20 must be installed on the computer. This is a Perl interface that allows communication with the Keysight Oscilloscope, allowing to call firmware and built in APIs by Keysight. The download the files listed below in command prompt [2].

Automation Packaged Files

- Lab::VISAc in VISA.pm
- Changes
- MANIFEST
- META.json
- Makefile.PL
- README

Automation Compile

- Perl Makefile.PL
- Make
- Make install
- Perl -Mblin automation_Link.pl

This automation package bundle must be on the host PC. The automation compile files are to build the system files and allow a confirmation if there has been a connection established between Host PC and Keysight Oscilloscope by running the script shown on next page in Figure 4.13. The return should be SUCCESS if there is a connection, else NO SUCCESS. The goal for this automation is to develop a foundation for auto analysis. Once this connectivity has been established, the user will create a custom based script to their needs using the Keysight APIs. One example of this is to auto calibrate all the channels by calling API call **SYST:CAL:ALL:SEL** rather than manually calibrating the channels. The ideal goal for the user is to generate a script

that calls these APIs to increase efficiency.

```
#!/usr/bin/perl

use Lab::VISA;

my($status,$sesn) = Lab::VISA::viOpenDefaultRM();

printf "status: %x (%s)\n", $status, (($status == $Lab::VISA::VI_SUCESS)) ? "success" : "no sucess");
print "sesn: $seen\n";

my($status, $findList, $retcnt, $instrDesc) = Lab::VISA::viFindRsrs($sesn, "ASRL1::INSTR");

printf "status: %x (%s)\n", $status, (($status == $Lab::VISA::VI_SUCESS)) ? "success" : "no sucess");
print "findList: $findList\n";
print "retcnt: $retcnt\n";
print "instrDesc: $instrDesc\n";
```

Figure 4.13: Scope Connectivity [2]

Chapter 5

Conclusions and Future Work

In this thesis, 3D NAND technology was explored by developing experimental analysis to overcome the engineering challenges for NAND floating gate cell technology. Improving read and write performance for customer operations through various lab orientated experiments was implemented on silicon. Optimizing performance and increasing a higher die yield was conducted and direct simulation-to-silicon correlation was developed to relate the mathematical and simulation model to the silicon analysis.

Using floating gate technology for NAND Flash solid-state drives allows for better data retention with QLC capabilities. A benchmark was created for the analyzed data when implementing firmware logic and customer device capabilities to define high speed NAND interface technology. Developing methods of increasing data retention and vulnerabilities that can cause data corruption were analyzed by various experiments using silicon hardware.

Investigating the byte mismatch and developing misalignment checks were cre-

ated by analyzing the ground bounce, probing analysis, temperature analysis, and defect sensor analysis. This analysis implemented shows the root cause solution to reduce the number of failing bytes seen for the page buffer data latching (Read, Program, and Erase). With this analysis, the failing signature is reduced when programming and reading data from the internal latches. The combined process yields a lower margin of byte failure.

Implementing various test experiments on silicon wafers enhanced our understanding of the semiconductor device physics and how a variety of factors, and their interaction, present engineering challenges to exploitation of NAND floating gate technology. Complimenting this research, automation of data collection and data analysis was also developed that greatly improves the outcome testing. Future work regarding this work is to improve the automation of data analysis. The current implementation has created a communication channel between the silicon wafer and command line interface, allowing full capability of calling Keysight APIs to run analysis purely dependent on a firmware routine. This decreases the time spent on manually analyzing the data and employs automation to analyze the data. Currently, implementing auto-gating functionality to automate the process of timing analysis is being developed. The future steps are to develop an understanding of auto-gating functionality and incorporate auto-triggering on condition sets and compute the analysis.

Bibliography

[1] *Picroprobe - GGB Industries INC. [Online]*, 2019. Avaliable: http://www.ggb.com/18c.html.

[2] A. Httel. *LAB::VISA - Perl interface to the National Instruments VISA library. [Online]*, 2019. Avaliable: https://metacpan.org/pod/Lab::VISA.

[3] Micron Technology Inc. *NAND Flash Memory*, 2004. 2,4, 8Gb: x8/x16 Multiplexed NAND Flash Memory Features datasheet. Rev 1.

[4] Micron Technology Inc. *NAND Flash Memory*, 2004. 4Gb, 8GB, and 16Gb x8 NAND Flash Memory Features datasheet. Rev B.

[5] Phison Wstern Digital SK Hynix Intel, Micron and Sony. *ONFi OPEN NAND FLASH INTERFACE Specification*, 2005. [Revised December 2017].

[6] E. Tiomkin. *Industrial Temperature and NAND Flash in SSD Products. EE Web [Online]*, 2012. Avaliable: https://www.eeweb.com/profile/eli-tiomkin/articles/industrial-temperature-and-nand-flash-in-ssd-products.