UNIVERSITY OF CALIFORNIA
Los Angeles

Pin Assignment for 2.5D Dielet Assembly

A thesis submitted in partial satisfaction
of the requirements for the degree
Master of Science in Electrical and Computer Engineering

by

Yizhang Wu

2019

ABSTRACT OF THE THESIS

Pin Assignment for 2.5D Dielet Assembly

by

Yizhang Wu
Master of Science in Electrical and Computer Engineering
University of California, Los Angeles, 2019
Professor Puneet Gupta, Chair

In this thesis we present several Linear Programming (LP) based pin assignment algorithms for silicon interposer based 2.5D dielet integration with different purposes. In the first part of the dissertation we present three different formulations of the pin assignment problem: 1) Integer Linear Programming based (ILP formulation), 2) ILP based with LP relaxation (Relaxed-ILP formulation), and 3) LP based modeling (LP formulation). ILP based modeling employees the use of integer assignment matrices, which comply with the nature of pin assignment problem where only discrete locations are available to pins. However, due to NP-hard nature of ILP problems, complexity and run time of ILP formulation is not scalable and pin assignment can be accomplished only for designs with limited size. Relaxed-ILP formulation can reduce the run time significantly and can scale to larger designs. While solutions generated with LP relaxation are usually not valid pin assignments, we apply a rounding heuristic to legalize the solution. Finally the LP formulation with rounding heuristic can further reduce the complexity of the problem and make the run time more scalable. Since I/O power consumption for each die is a major overhead of 2.5D integration, in the second part of the thesis we present a framework for optimizing the energy-per-bit required by I/O cells to drive the interconnects between different dies. We use RC wire model to simulate power and latency of interconnects, and use 2π model to simulate wire performance in presence of wire coupling and ground bouncing. We model the energy required for each

link as a function of the required link speed and wire length and modify LP formulation accordingly. We compare our model with Cadence Innovus and show that we can achieve lower power overhead. In the third part of the thesis we propose a model for multi-floorplan pin assignment (MFPA). We perform pin assignment without the knowledge of floorplan such that the block under consideration can be reused in multiple designs and avoid potential routing hotspot or wire congestion. Also we show that such MFPA flow can effectively reduce the worst case and average interconnect length across a set of random floorplans.

The thesis of Yizhang Wu is approved.

Lei He

Sudhakar Pamarti

Puneet Gupta, Committee Chair

University of California, Los Angeles

2019

To my parents

Table of Contents

List of Figures

List of Tables

Acknowledgment

I would like to thank my advisor, Prof. Puneet Gupta for the constant guidance, support and motivations since I joined NanoCAD as an undergraduate student. Our frequenct technical discussions have been very educative and constructive for this and previous projects. This thesis would not have been possible without his constant inputs.

I would like to thank my collaborator Saptadeep Pal. Discussions, suggestions and technical help from him have been very crucial in the completion of this porject. I would also like to acknowledge the support of my labmates during the past two years: Dr. Wei-Che Wang, Saptadeep Pal, Irina Alam, Tianmu Li, Wojciech Romaszkan and Eugene Chu. Thank you Dr. Wei-Che Wang (now at Cadence) for helping me getting started with research, leading me through the previous projects for the past three years, and providing me with invaluable suggestions on different aspects.

Lastly, I would like to thank my parents for their unconditional love, encouragement and support.

CHAPTER 1

Introduction

1.1 VLSI Design Flow

Modern VLSI design flow typically consists of two different parts: front end design and back end design. Front end design usually refers to the functional design of the system, which includes the high level system specification, architecture design, RTL design and logic synthesis. System specification are implemented using high level HDL, and HDL codes are synthesized into gate-level netlist using logic synthesis tools. Back end design refers to the physical design of the system, where physical layout are designed. Traditionally, physical design is a process of partitioning, floorplanning, placement, pin assignment, routing and creating GDSII layout file. In a flat design flow, floorplanning and placement step determines the dimension of the design and location of cells, and global and detail routing connects all cells to complete the netlist [43][4][30].

Most of system-on-a-chip (SoC) device can be designed in a traditional flat flow, however, in today's multi-million gate designs, it is usually desirable to adopt a hierarchical flow where a design is divided into manageable partitions and each partition can be independently designed. Besides, enabled by today's deep sub-micron semiconductor technology, large-scale integration of reusable intellectual property (IP) blocks or macros also favors hierarchical design methodology [18]. In such a system level design flow, the dimensions and locations of the blocks are determined during floorplanning and placement phase, and the locations of pins on the block boundary are determined during the pin assignment step. Finally chip assembly is finished by routing at global system level to connect pins of various design blocks [11]. Fig. 1.1 shows a typical VLSI design flow. In this work we focus on the pin assignment

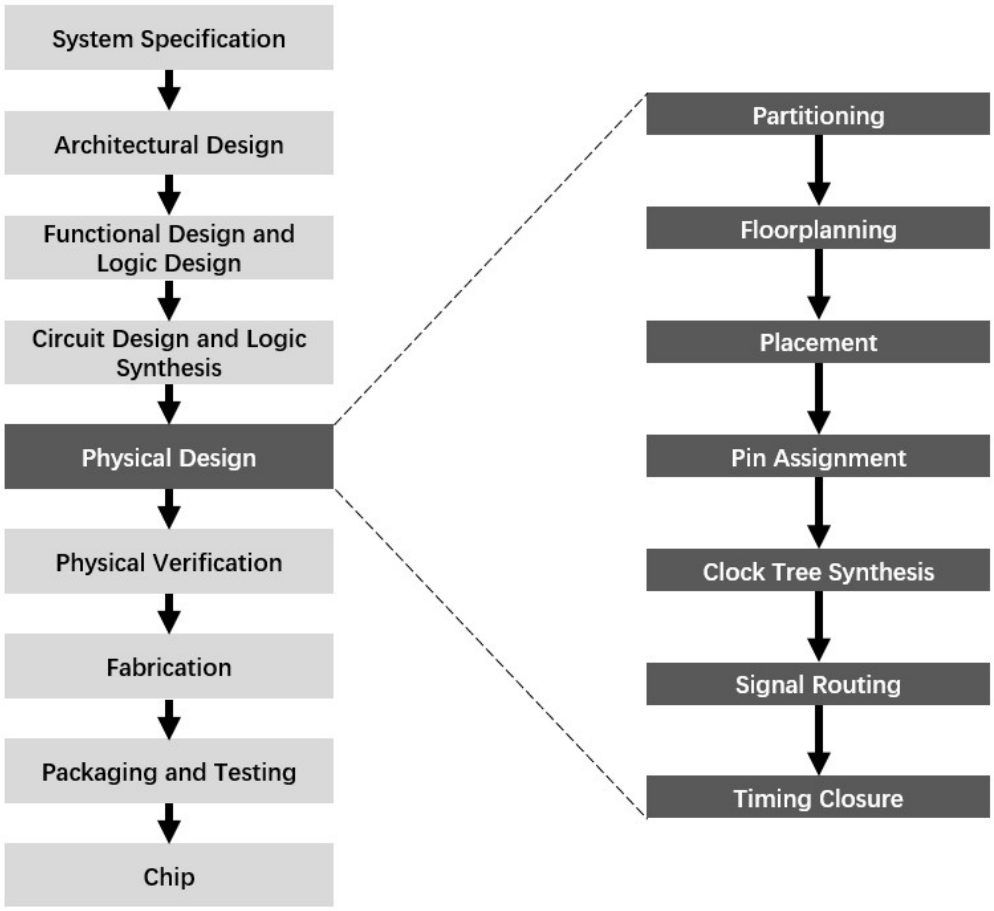


Figure 1.1: Typical VLSI design flow

step during the physical design flow.

1.2 Silicon Interposer Based 2.5D Integration

In the last few decades, Moores Law has been followed by the semiconductor industry effectively, which pushed the miniaturization of transistors and enabled integration of large scale SoCs [9]. Since the invention of the first integrated circuits, the improvement in semiconductor wafer processing and lithography technologies allowed the density of two dimensional chips to double every 18-24 months [27]. The increasing demands for high-performance and large SoCs from applications like data center, mobility and Internet of Things (IoT) has also been driving the scaling of semiconductors[24]. For instance, ITRS 2.0 roadmap [8] outlines

that the demands from datacenter and microserver markets requires a roughly 3X increase in available memory and 4X increase in the number of processor cores per socket and rack unit, respectively, over the next ten years[35].

However, to meet the increasing demands with traditional process scaling is becoming more difficult and inefficient. First of all, the Moore's Law target cadence has been slipping as meeting the desired transistor scaling rate is becoming difficult for foundries[22]. Besides, the benefits from scaling starts to diminish significantly below 28nm node. As design complexity and density of ICs increase, costs and risk associated with these designs start becoming prohibitive as more advanced lithographic nodes are required, which suffers from higher manufacturing costs and yield loss [9][37]. To address the demands for increasing system integration, interposer-based 2.5D integration is proposed as an alternative to traditional 2D monolithic integration. In 2.5D integration, large dies are partitioned into multiple chiplets and these chiplets are bonded by high-yield, high-bandwidth, chip-to-chip interconnect through silicon interposer to form a 2.5D integrated IC. Such die-partitioning approach also improves the yield [35][34]. Modern SoCs can contain analog, memory and digital dies, which typically are manufactured using different technologies. 2.5D integration also facilitates such heterogeneous integration where dies from different technologies can be placed on the same interposer to create a single chip [34][29]. A typical Si interposer based 2.5D stack is shown in Fig. 1.2. 2.5D integration has already been employed by many devices on the market. For example, AMD Radeon Fury GPU is the first 2.5D IC GPU, where the entire GPU memory system is integrated into the package using a Si interposer [23]. Xilinx has also produced 2.5D FPGA ICs containing logic dies and high speed transceiver dies from different manufacturing processes [32].

2.5D dielet integration can be summarized in following steps: placing dies on the interposer, routing inter-die connections using metal layers in Si interposer, and placing through silicon vias (TSVs) in the interposer to connect die microbump to package-level interconnects [34]. Both active-lite interposers and passive interposers are developed for 2.5D and 3D integration. Active-lite interposers can include active components such as diodes, BJTs

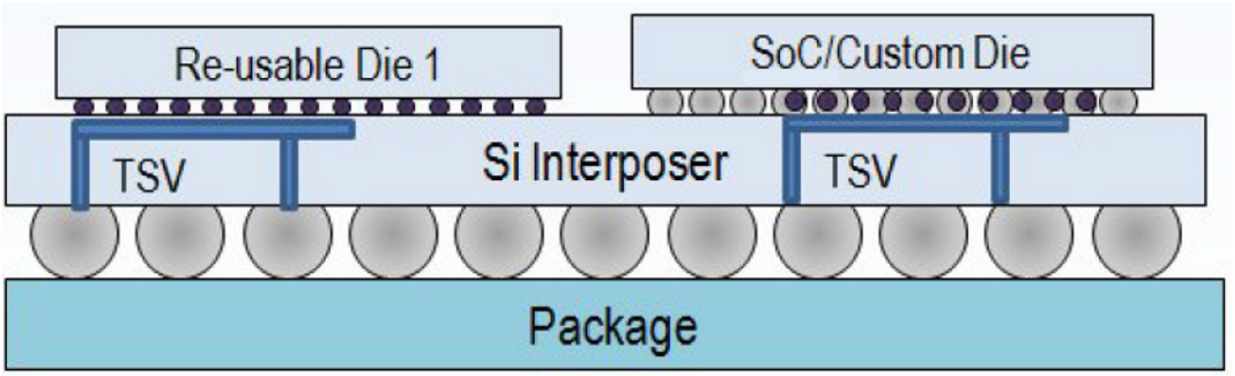


Figure 1.2: A typical Si interposer-based 2.5D design [14]

to enable faster interconnects and increase the flexibility of the design[17][35]. In this work we focus on 2.5D integration using passive interposer with only passive interconnects and TSVs since they are more sensitive to the interconnect designs. Due to the metal-only nature of passive interposer, their capability of providing high bandwidth and low latency for high performance systems can be limited by the length of interconnects [35]. Also, passive interconnects through the interposer require each die to have I/O cells with sufficient drive strength, which potentially increase the power overhead of chip-level integration. In this work we propose several algorithms for minimizing the interconnect length and power overhead through die level pin assignment.

1.3 Pin Assignment

After floorplan phase, routing at the system level connecting pins of different design blocks is a crucial part of the physical design flow as it is critical to system level timing performance. In a hierarchical design flow, since pin assignment is closely related to global routing, an off-the-shelf die's pin assignment can lead to long interconnect on silicon interposer, which is not ideal for system level timing sign off [10].

A lot of works have been done on pin assignment. Some old approaches on block by block basis includes concentric circle mapping method [21], nine zone method[26], and topological pin assignment method[7]. In concentric circle mapping method, two concentric circle are

drawn for each block where the inner circle corresponds to the boundary of the block being considered and the outer circle has all pins on other blocks mapped on it. Then a pin assignment for the current block is achieved by computing the best mapping between inner and outer circle. In nine zone method the design is divided into nine possible zones for net locations to help with pin assignment. In topological pin assignment, on each block being considered, a "radar screen" is conceptually imposed to sweep the design and determine the pin locations. These approaches process pin assignment problem on a block by block basis, thus the resulting pin assignment quality largely depends on the processing order of blocks. There are also works that perform a net-by-net approach, where pin assignment and global routing is considered simultaneously. Cong in [11] uses block boundary decomposition to model pin assignment into global connection arrangement and channel pin assignment, and algorithms are proposed to optimize each step. In [20] pin assignment is also modeled as channel pin assignment and is performed simultaneously with global routing, and block reshaping and positioning. However, all these block-by-block or net-by-net algorithms are not "global" since they fail to optimize the design in a global sense. Works also have been done to approach pin assignment globally. For instance, [24][16] use simulated annealing to find a global optimal solution. Another common approach formulates pin assignment into linear programming (LP) problem and solve with LP solvers. [16] models pin assignment as mixed integer linear programming (MILP) and use assignment matrix for each block to map pins to legal pin positions. They also propose to use simulated annealing to accelerate the process since MILP is an NP-hard problem. Work in [34] uses simulated annealing to approach FPGA flexible I/O assignment and die placement on a 2.5D interposer. A lot works also have been done on 3D integration. Similarly to pin assignment, a number of works model the through silicon via (TSV) assignment as a multi-commodity min-cost problem, and use different method such as Lagrangian relaxation to reduce the runtime [41][40][15].

In this work we develop several pin assignment modeling and algorithms specifically for 2.5D integration through a global approach, including ILP modeling, ILP with relaxation and LP modeling. While most existing works take interconnect length minimization as

the primary objective, we also propose a pin assignment model for minimizing the IO power required to drive interconnects in passive silicon interposer. In addition, all works mentioned above require the global floorplan or perform floorplaning along with pin assignment to achieve an optimal result for a specific design. In this work we develop a multi-floorplan pin assignment flow. Without optimizing for a specific floorplan, we achieve a pin assignment such that a block is portable to multiple different floorplans while reduce the average and worst case interconnect length. We also use channel routing model to avoid potential routing hotspot and wire congestion by introducing pin redundancy.

1.4 Thesis Outline

This thesis primarily focuses on the pin assignment for 2.5D dielet integration. The thesis is organized as follows:

- In Chapter 2 we propose three different formulations for pin assignment: ILP based formulation, ILP based with LP relaxation, and run time scalable LP formulation. We use different benchmarks to compare the runtime, and optimality of these formulations against Cadence Innovus[1].
- In Chapter 3 we present a framework for power optimization during the pin assignment phase. Using RC model, we model required IO power as a function of wire length and interconnect speed and modify out pin assignment formulation accordingly. We show that our model can achieve better power performance than Cadence Innovus.
- In Chapter 4 we develop a multi-floorplan pin assignment framework, where pin assignment is performed without the knowledge of floorplan. We develop a rule of thumb for pin assignment which avoids potential routing hotspot and wire congestion, and reduce the average and worst case average wire length when the block is integrated in to a set of random floorplans.
- Chapter 5 concludes the thesis.

CHAPTER 2

Modeling of Pin Assignment Problem

In this chapter we introduce the model of pin assignment problem, including problem inputs and outputs. Then we present 3 different formulations of pin assignment: 1) ILP based (ILP formulation), 2) ILP based with LP relaxation (relaxed-ILP formulation), and 3) run time scalable LP based formulation (LP formulation). For second and third problem formulation, since non-integer formulation can result in invalid pin assignment solutions, we also present our rounding heuristics which legalize the solutions.

2.1 Problem Formulation

To model pin assignment as an LP problem, we first abstract design information and design constraints into mathematical constraints. In this chapter and Chapter 3 we perform pin assignment after floorplanning phase, thus information related to the floorplan of the design such as locations of each block is known. Besides, we perform hierarchical pin assignment, where multi-instantiated cells of the same master cell will have identical pin assignment solutions. Assume there are in total t different type of cells, n blocks, m nets and k pins in the design, inputs to the problem consist of

- Set of master cells, $M = \{m_1, m_2, ..., m_t\}$
- Set of blocks, $B = \{b_1, b_2, ..., b_n\}$ and their bounding boxes defined by $bbox_i = \{Lx_i, Ly_i, Rx_i, Ry_i\}$, which are the coordinates of lower-left and upper-right corner.
- Set of pins associated with each block b_i, $Pins_i = \{pin_{i1}, pin_{i2}, ..., pin_{i|Pins_i|}\}$, where $|Pins_i|$ is the number of pins associated with block b_i

- Set of nets, $N = \{N_1, N_2, ..., N_m\}$ and netlist connections.
- Pin assignment constraints including pin location step size, $pinStepSize$, and minimum pitch between neighboring pins, $minPinPitch$.

The outputs are the coordinates for each pin.

2.2 ILP Based Formulation

Given a VLSI design, since there are a finite number of pins and available positions to place a pin, we can easily formulate pin assignment as an ILP problem. We develop our algorithm based on the work in [16]. In addition to the inputs mentioned in 2.1, we are able to obtain the following:

- $Pos_i = \{pos_{i1}, pos_{i2}, ..., pos_{i|Pos_i|}\}$, the set of available pin positions on block $b_i, i = 1, 2, ..., n$ where $|Pos_i|$ is the number of available positions. Available pin positions can be obtained from the coordinates of the block, $minPinPitch$ and $pinMoveStep$.
- $|N_i|$, $i = 1, 2, ..., m$, the number of pins associated with net N_i.

For each block b_i, we represent Pos_i as two $|Pos_i| \times 1$ vectors:

$$Pos_i\_X = (pos_{i1}\_x, pos_{i2}\_x, ..., pos_{i|Pos_i|}\_x)^T \tag{2.1}$$

$$Pos_i\_Y = (pos_{i1}\_y, pos_{i2}\_y, ..., pos_{i|Pos_i|}\_y)^T \tag{2.2}$$

where the coordinates of pos_{ij}, the j_{th} available position in block b_i, is decomposed to (pos_{ij}_x, pos_{ij}_y). Then for each block we define a $|Pos_i| \times 2$ matrix containing all available positions as

$$AvlbPos\_Mat_i = (Pos_i\_X, Pos_i\_Y) \tag{2.3}$$

Note that $AvlbPos\_Mat_i$ is a matrix whose entrees are all known from floorplan. Similarly, we can define the variables for coordinates of each pin pin_{ij} as (pin_{ij}_x, pin_{ij}_y). Then for

each block we can define two $|Pin_i| \times 1$ vectors and a $|Pin_i| \times 2$ matrix as follows:

$$Pin_i\_X = (pin_{i1}\_x, pin_{i2}\_x, ..., pin_{i|Pin_i|}\_x)^T \tag{2.4}$$

$$Pin_i\_Y = (pin_{i1}\_y, pin_{i2}\_y, ..., pin_{i|Pin_i|}\_y)^T \tag{2.5}$$

$$PinPos\_Mat_i = (Pin_i\_X, Pin_i\_Y) \tag{2.6}$$

We further define a $|Pin_i| \times |Pos_i|$ assignment matrix, A_i, for each block b_i. $A_{i\_kh} = 1$ if the k_{th} pin of b_i is assigned to the h_{th} available position, and $A_{i\_kh} = 0$ otherwise. Thus we can obtain the constraints representing the pin assignment for block b_i:

$$PinPos\_Mat_i = A_i \times AvlbPos\_Mat_i \tag{2.7}$$

$$A_{i\_kh} \quad \in \quad \{0, 1\} \tag{2.8}$$

Note that entrees of $PinPos\_Mat_i$ and A_i are variables to be solved. In the case of hierarchical pin assignment, for multi-instantiated blocks with the same master cell, $AvlbPos\_Mat$ and $PinPos\_Mat$ should have identical size.

One criteria of evaluating a 2.5D die-level pin assignment solution is the resulting weighted sum of wire length of each net after system-level routing. Since accurate wire length is not available at pin assignment stage which is before routing, in this work we use Half Perimeter Wire Length (HPWL) to estimate the length of a net. We define $(NL_{kx}, NL_{ky}, NR_{kx}, NR_{ky})$ as the bounding box that contains all pins associated with net N_k, where (NL_{kx}, NL_{ky}) and (NR_{kx}, NR_{ky}) are the coordinates of lower-left and upper-right corner, respectively. Since pins associated with a net are known from the netlist, we can model the wire length, WL_k for a net N_k as

$$WL_k = NR_{kx} - NL_{kx} + NR_{ky} - NL_{ky} \tag{2.9}$$

$$NL_{kx} \leq pin_{ij}\_x \quad \forall \quad pin_{ij} \in N_k \tag{2.10}$$

$$NL_{ky} \leq pin_{ij}\_y \quad \forall \quad pin_{ij} \in N_k \tag{2.11}$$

$$NR_{kx} \geq pin_{ij}\_x \quad \forall \quad pin_{ij} \in N_k \tag{2.12}$$

$$NR_{ky} \geq pin_{ij}\_y \quad \forall \quad pin_{ij} \in N_k \tag{2.13}$$

Notation	Meaning
$M = \{m_1, m_2, ..., m_t\}$	Set of master cells
$B = \{b_1, b_2, ..., b_n\}$	Set of blocks in the design
$Pins_i = \{pin_{i1}, pin_{i2}, ..., pin_{i\|Pins_i\|}\}$	Pins associated with block b_i
$N = \{N_1, N_2, ..., N_t\}$	Set of nets in the design
$(NL_{kx}, NL_{ky}, NR_{kx}, NR_{ky})$	Variables representing N_k bounding box coordinates
WL_k	Variables representing wire length of net N_k
$pinStepSize$	Step size between adjacent legal pin locations
$minPinPitch$	Minimum pitch between neighboring pins
$maxWL$	Maximum allowed wire length in design
$AvlbPos\_Mat_i$	Matrix containing all available pin positions in block b_i
$PinPos\_Mat_i$	Matrix of variables for coordinates of pins associated with block b_i
A_i	Assignment matrix variables for block b_i

Table 2.1: Notations in ILP modeling

Also, for designs with constrained operating frequency or performance requirements, long interconnects may violate timing constraints as the delay and slew can be unacceptable. Therefore a maximum allowed wire length constraint can be applied to our problem formulation. If we note the allowed maximum wire length within the design as $maxWL$, the constraints can be simply stated as

$$WL_k \leq maxWL \quad k \in \{1, 2, ...t\} \tag{2.14}$$

A summary of introduced notations and variables is shown in table 2.1.

Constraints of our modeled ILP come from assignment matrix A. Since each pin has to be assigned to one location and each location can only be assigned to one pin, we can add following constraints on each A_i:

$$\sum row = 1 \quad \forall \quad row \in A_i \tag{2.15}$$

$$\sum column \leq 1 \quad \forall \quad column \in A_i \tag{2.16}$$

To add constraints for hierarchical pin assignment where multi-instantiated blocks should have identical pin assignment, we define a function

$$(x_{rel}, y_{rel}) = Abs2Rel(b_i, (x_{abs}, y_{abs})) \tag{2.17}$$

which can translate a coordinates (x_{abs}, y_{abs}) into relative coordinates with respect to the origin of block b_i. If block b_j and b_k are multi-instantiated cells, we arrange $AvlbPos\_Mat_j$ and $AvlbPos\_Mat_k$ such that

$$Abs2Rel(b_j, (pos_{ji}\_x, pos_{ji}\_y)) = Abs2Rel(b_k, (pos_{ki}\_x, pos_{ki}\_y)), i = 1, ..., |Pos_j| \tag{2.18}$$

We also arrange $PinPos\_Mat_j$ and $PinPos\_Mat_k$ such that pin_{ji} and pin_{ki}, $i = 1, ..., |Pos_j|$, represent the same logical pin in two blocks. Then we can we force assignment matrix to have same solution for block with same master cell, as shown in Eq. 2.19. Note that constraints in Eq. 2.15 and 2.16 only need to be applied on one of $A_i, A_j, ..., A_x$ to reduce the number of constraints.

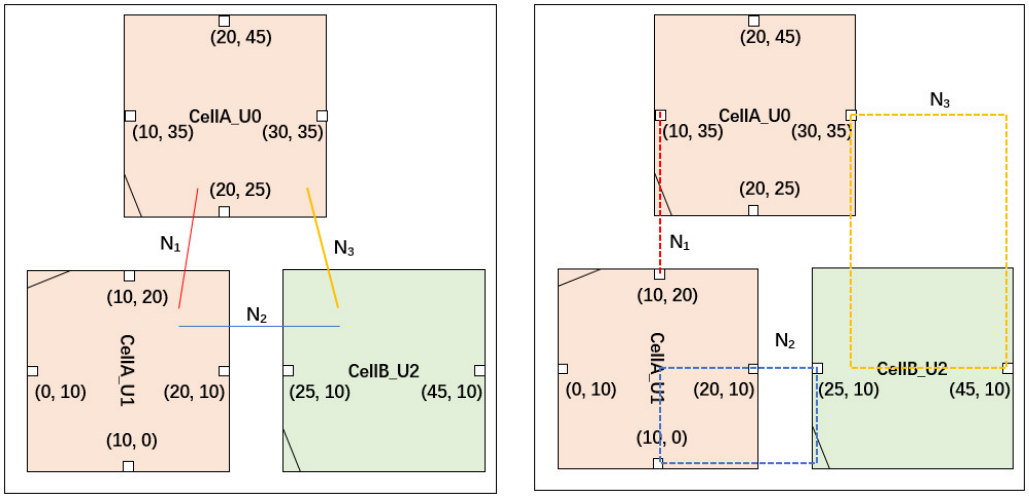
$$A_i = A_j = ... = A_x \quad if \quad b_i,\ b_j,\ ...,\ b_x \ have \ same \ master \ cell \tag{2.19}$$

The objective of our ILP can be the weighted sum of wire length, as shown in Eq. 2.20, where λ_i is the weight of net N_i.

$$\textbf{Objective} : Minimize \sum_{i=1}^{t} \lambda_i WL_i \tag{2.20}$$

We use an example floorplan shown in Fig. 2.1 to illustrate the ILP formulation of pin assignment problem given the floorplan. As shown in Fig. 2.1a, the input is a floorplan with 3 blocks: $U0$, $U1$, and $U2$, where $U0$ and $U1$ have different orientation but same master cell, and nets $N_1 = \{pin01, pin11\}$, $N_2 = \{pin12, pin21\}$, $N_3 = \{pin02, pin22\}$. The available pin positions are labeled in the figure and we can obtain three $AvlbPos\_Mat$:

$$AvlbPos\_Mat_0 = \begin{bmatrix} 10 & 20 & 30 & 20 \\ 35 & 45 & 35 & 25 \end{bmatrix}^T$$



(a) Floorplan with with 3 nets, 3 blocks and 2 master cells

(b) corresponding ILP solution for pin assignment

Figure 2.1: Example floorplan and netlist for ILP formulation

$$AvlbPos\_Mat_1 = \begin{bmatrix} 10 & 20 & 10 & 0 \\ 20 & 10 & 0 & 10 \end{bmatrix}^T$$

$$AvlbPos\_Mat_2 = \begin{bmatrix} 25 & 45 \\ 21 & 10 \end{bmatrix}^T$$

Note that since $U0$ and $U1$ have the same master cell, $AvlbPos\_Mat_0$ and $AvlbPos\_Mat_1$ are arranged to satisfy the relation in Eq. 2.18. Then variable matrices for pin position can be obtained as:

$$PinPos\_Mat_0 = \begin{bmatrix} pin_{01}\_x & pin_{01}\_y \\ pin_{02}\_x & pin_{02}\_y \end{bmatrix}$$

$$PinPos\_Mat_1 = \begin{bmatrix} pin_{11}\_x & pin_{11}\_y \\ pin_{12}\_x & pin_{12}\_y \end{bmatrix}$$

$$PinPos\_Mat_2 = \begin{bmatrix} pin_{21}\_x & pin_{21}\_y \\ pin_{22}\_x & pin_{22}\_y \end{bmatrix}$$

Since block $U0$ and block $U1$ should have identical pin assignment, we only need two

assignment matrices in this case, A_A and A_B. A_A is a 2×4 matrix and A_B is a 2×2 matrix.

$$A_A = \begin{bmatrix} A_{A\_11} & A_{A\_12} & A_{A\_13} & A_{A\_14} \\ A_{A\_21} & A_{A\_22} & A_{A\_23} & A_{A\_24} \end{bmatrix}$$

$$A_B = \begin{bmatrix} A_{B\_11} & A_{B\_12} \\ A_{B\_21} & A_{B\_22} \end{bmatrix}$$

The ILP for pin assignment of floorplan shown in Fig. 2.1a can be stated as follows:

$$Minimize: \tag{2.21}$$

$$WL_1 + WL_2 + WL_3 \tag{2.22}$$

$$Subject\ to: \tag{2.23}$$

$$WL_1 = NR_{1x} - NL_{1x} + NR_{1y} - NL_{1y} \tag{2.24}$$

$$NL_{1x} \leq pin_{01\_}x \quad NL_{1x} \leq pin_{11\_}x \tag{2.25}$$

$$NL_{1y} \leq pin_{01\_}y \quad NL_{1y} \leq pin_{11\_}y \tag{2.26}$$

$$NR_{1x} \geq pin_{01\_}x \quad NR_{1x} \geq pin_{11\_}x \tag{2.27}$$

$$NR_{1y} \geq pin_{01\_}y \quad NR_{1y} \geq pin_{11\_}y \tag{2.28}$$

$$WL_2 = NR_{2x} - NL_{2x} + NR_{2y} - NL_{2y} \tag{2.29}$$

$$NL_{2x} \leq pin_{12\_}x \quad NL_{2x} \leq pin_{21\_}x \tag{2.30}$$

$$NL_{2y} \leq pin_{12\_}y \quad NL_{2y} \leq pin_{21\_}y \tag{2.31}$$

$$NR_{2x} \geq pin_{12\_}x \quad NR_{2x} \geq pin_{21\_}x \tag{2.32}$$

$$NR_{2y} \geq pin_{12\_}y \quad NR_{2y} \geq pin_{21\_}y \tag{2.33}$$

$$WL_3 = NR_{3x} - NL_{3x} + NR_{3y} - NL_{3y} \tag{2.34}$$

$$NL_{3x} \leq pin_{02\_}x \quad NL_{3x} \leq pin_{22\_}x \tag{2.35}$$

$$NL_{3y} \leq pin_{02\_}y \quad NL_{3y} \leq pin_{22\_}y \tag{2.36}$$

$$NR_{3x} \geq pin_{02\_}x \quad NR_{3x} \geq pin_{22\_}x \tag{2.37}$$

$$NR_{3y} \geq pin_{02\_}y \quad NR_{3y} \geq pin_{22\_}y \tag{2.38}$$

$$PinPos\_Mat0 = A_A \times AvalPos\_Mat0 \tag{2.39}$$

$$PinPos\_Mat1 = A_A \times AvalPos\_Mat1 \tag{2.40}$$

$$PinPos\_Mat2 = A_B \times AvalPos\_Mat2 \tag{2.41}$$

$$A_{A\_11} + A_{A\_12} + A_{A\_13} + A_{A\_14} = 1 \tag{2.42}$$

$$A_{A\_21} + A_{A\_22} + A_{A\_23} + A_{A\_24} = 1 \tag{2.43}$$

$$A_{A\_11} + A_{A\_21} \leq 1 \tag{2.44}$$

$$A_{A\_12} + A_{A\_22} \leq 1 \tag{2.45}$$

$$A_{A\_13} + A_{A\_23} \leq 1 \tag{2.46}$$

$$A_{A\_14} + A_{A\_24} \leq 1 \tag{2.47}$$

$$A_{B\_11} + A_{B\_12} = 1 \tag{2.48}$$

$$A_{B\_21} + A_{B\_22} = 1 \tag{2.49}$$

$$A_{B\_11} + A_{B\_21} \leq 1 \tag{2.50}$$

$$A_{B\_12} + A_{B\_22} \leq 1 \tag{2.51}$$

$$A_A \in \{0,1\} \quad A_B \in \{0,1\} \tag{2.52}$$

The results of this ILP is shown in Fig. 2.1b and the values for declared variables are:

$$A_A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad A_B = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

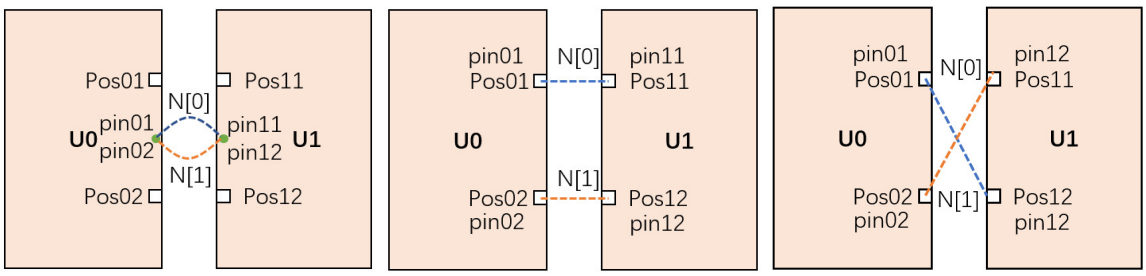
$$N_1 : \ pin01 = (10, 20) \quad pin11 = (10, 35)$$

$$N_2 : \ pin12 = (30, 35) \quad pin21 = (45, 10)$$

$$N_3 : \ pin02 = (10, 0) \quad pin22 = (25, 10)$$

2.3 ILP Based Formulation with LP Relaxation

Due to the NP-hard nature of ILP problems, complexity and run time is not scalable if design size is large. On the other hand, LP problems does not require variables to be integers and can be solved faster compared to ILP. In this section we propose a Relaxed ILP formulation



(a) Initial pin assignment from LP solution

(b) Desirable fix from LP solution

(c) Bad fix with unnecessary wire crossing

Figure 2.2: Example of illegal pin assignment from LP formulation with good and bad rounding results

which apply LP relaxation and rounding on ILP formulation presented in Section 2.2. Among all variables declared in ILP formulation, entrees of assignment matrices, A_i, are required to be either 1 or 0. Thus, instead of constraining A_i to be an integer matrix as shown in constraints 2.8, we can relax the constraints to

$$0 \leq A_{i\_kh} \leq 1 \tag{2.53}$$

while keeping all other constraints unchanged. For the example shown in Section 2.2, we only need to modify the last constraints into

$$0 \leq A_A, A_B \leq 1$$

Since solution to assignment matrix becomes non-integer values, resulting location for a given pin from the relation in Eq. 2.7 may sit between two available pin positions, which is not a legal assignment. Thus we present our rounding heuristics for fixing the illegal pin assignment. Rounding is performed block by block, and blocks with the same master cell are rounded together. Before rounding is applied, we prioritize all pins based on the estimated wire length such that pins associated with longer nets are rounded before pins associated with shorter nets. For each block, we map the LP solution to its nearest available legal pin position while maintaining $minPinPitch$ constraints.

One common scenario during rounding phase is shown in Fig. 2.2. The LP solution for

pin assignment is shown in Fig. 2.2a, where pins for two nets between block $U0$ and $U1$, $N[0]$ and $N[1]$, overlap with each other, i.e. $pin01$ and $pin02$ have same solution, and $pin11$ and $pin12$ have same solution. If the distance from $pin01, pin02$ to available pin positions $Pos01$ and $Pos02$ are the same, and similarly for $U1$, both fix shown in Fig. 2.2b and Fig. 2.2c are possible solutions from the rounding heuristic; however, solution in Fig. 2.2c can result in larger wire length and worse routing congestion due to unnecessary crossing of two nets. To reduce the number of unnecessary crossing of wires after rounding, our heuristic looks for available positions in four directions in the same sequence for each pin. In the example of Fig. 2.2, prioritizing pin step ensures that both blocks round pins associated with one net before pins associated with the other net. Without the loss of generality, assume that block $U0$ rounds $pin01$ before $pin02$ and block $U1$ rounds $pin11$ before $pin12$. During rounding $pin01$, our heuristic will check the availability of $Pos01$ before $Pos02$, and assign $pin01$ to $Pos01$, then assign $pin02$ to $Pos02$. Similarly block $U1$ will assign $pin11$ to $Pos11$, and assign $pin12$ to $Pos12$. Thus situation shown in Fig. 2.2c can be avoided. The algorithm for fixing the pin location of one block, b_i, is shown in Algorithm 1. The inputs to the heuristic include LP solutions for pin locations, $PinPos\_Mat$, and prioritized array of pins, $PinPriority$. Outputs are fixed pin location, $Fixed\_PinPos\_Mat$

2.4 Run Time Scalable LP formulation

2.4.1 LP formulation

Another limitation of ILP modeling comes from the use of assignment matrix A_i which introduces $|Pin_i| \times |Pos_i|$ variables for every master cell. Such assignment matrix needs constraints on row and column. Besides, the number of constraints introduced by the assignment equations shown in Eq. 2.7 is large and can significantly increase the complexity of the ILP. Even with LP relaxation mentioned in Section 2.3 the run time may still be not scalable to large designs. Thus, in this section we present another run time scalable LP formulation of pin assignment problem without the use of assignment matrix. Similarly, we

Algorithm 1 Fix Pin Location Algorithm

Input: $PinPos\_Mat_i$ and $PinPriority_i$

Output: $Fixed\_PinPos\_Mat_i$

$AvlbPos\_Mat_i\_cp = AvlbPos\_Mat_i$
for pin_j in $PinPriority_i$ **do**
    tmp_PinPos_{ij} = nearest Point in $AvlbPos\_Mat_i$
    for $i \leq |AvlbPos_i|$ **do**
        if $tmp\_PinPosij + (i * pinStepSize, 0)$ in $AvlbPos\_Mat_i\_cp$ **then**
            $Fixed\_PinPos\_Mat_{ij} = tmp\_PinPosij + (0, i * pinStepSize)$
        else if $tmp\_PinPosij - (i * pinStepSize, 0)$ in $AvlbPos\_Mat_i\_cp$ **then**
            $Fixed\_PinPos\_Mat_{ij} = tmp\_PinPosij + (0, i * pinStepSize)$
        else if $tmp\_PinPosij + (0, i * pinStepSize)$ in $AvlbPos\_Mat_i\_cp$ **then**
            $Fixed\_PinPos\_Mat_{ij} = tmp\_PinPosij + (0, i * pinStepSize)$
        else $tmp\_PinPosij - (0, i * pinStepSize)$ in $AvlbPos\_Mat_i\_cp$
            $Fixed\_PinPos\_Mat_{ij} = tmp\_PinPosij - (0, i * pinStepSize)$
    $Fixed\_PinPos\_Mat_{ij}$ = nearest Point in $AvlbPos\_Mat_i\_cp$
    $AvlbPos\_Mat_i\_cp$.erase($Fixed\_PinPos\_Mat_{ij}$)
    $j = j + 1$

apply rounding heuristics after solving LP to obtain the final valid pin assignment.

In relaxed ILP formulation discussed in Section 2.3, it is common for LP to give an invalid solution similar to Fig. 2.2a. For instance, a 2×2 assignment matrix with value

$$A = \begin{bmatrix} 0.5 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

satisfies all constraints on A but results in two pins overlapping on each other. As the intention of applying constraints shown in Eq. 2.15 and Eq. 2.16 is to maintain the validity of pin assignment, these constraints become ineffective after LP relaxation. Therefore one insight we can obtain is to remove these constraints completely to reduce the complexity of the problem. Furthermore, since solutions for $PinPos$ after relaxation no longer correspond to positions in $AvlbPos\_Mat$ as A is not an integer matrix, we can free variables in $PinPos\_Mat$ from the relation shown in Eq. 2.7. Given a block b_i, we note its bounding box using the coordinates of its lower-left corner and upper-right corner as $\{BL_{ix}, BL_{iy}, BR_{ix}, BR_{iy}\}$. Then the constraints on a pin, pin_{ij}, associated with b_i are

$$BL_{ix} \leq pin_{ij}\_x \leq BR_{ix} \tag{2.54}$$

$$BL_{iy} \leq pin_{ij}\_y \leq BR_{iy} \tag{2.55}$$

which limit the location of pin_{ij} to be inside the bounding box of its associated cell. To maintain the minimum pin pitch rule, one naive way is to add constraints between every two pins in a block:

$$|pin_{ij}\_x - pin_{ik}\_x| + |pin_{ij}\_y - pin_{ik}\_y| \geq minPinPitch \quad \forall \quad pin_{ij}, pin_{ik} \in b_i, j \neq k \tag{2.56}$$

In the example shown in Fig. 2.1, constraints from Eq. 2.39 to Eq.2.52 can be replaced by following constraints:

$$10 \leq pin_{01}\_x \leq 30 \tag{2.57}$$

$$25 \leq pin_{01}\_y \leq 45 \tag{2.58}$$

$$10 \leq pin_{02}\_x \leq 30 \tag{2.59}$$

$$25 \leq pin_{02}\_y \leq 45 \tag{2.60}$$

$$|pin_{01}\_x - pin_{02}\_x| + |pin_{01}\_y - pin_{02}\_y| \geq minPinPitch \tag{2.61}$$

$$0 \leq pin_{11}\_x \leq 20 \tag{2.62}$$

$$0 \leq pin_{11}\_y \leq 20 \tag{2.63}$$

$$0 \leq pin_{12}\_x \leq 20 \tag{2.64}$$

$$0 \leq pin_{12}\_y \leq 20 \tag{2.65}$$

$$|pin_{11}\_x - pin_{12}\_x| + |pin_{11}\_y - pin_{12}\_y| \geq minPinPitch \tag{2.66}$$

$$25 \leq pin_{21}\_x \leq 45 \tag{2.67}$$

$$0 \leq pin_{21}\_y \leq 20 \tag{2.68}$$

$$25 \leq pin_{22}\_x \leq 45 \tag{2.69}$$

$$0 \leq pin_{22}\_y \leq 20 \tag{2.70}$$

$$|pin_{21}\_x - pin_{22}\_x| + |pin_{21}\_y - pin_{22}\_y| \geq minPinPitch \tag{2.71}$$

2.4.2 Bus Bundling

Without the use of assignment matrix, LP formulation mentioned in Section2.4.1 can reduce the complexity of the problem. However, minimum pin pitch constraints shown in Eq.2.56 can become the bottleneck, since the number of constraints needed if we add such constraints to every pair of pins is at the order of $O(|Pin_i|^2)$, which is not scalable to large designs. On the other hand, it is not necessary to apply minimum pitch constraints since rounding phase after solving LP will legalize all pin locations. However, without any constraints on pin pitch can cause pin snapping on each other as shown in Fig.2.2a, where same optimal solution is reached for every pin associated with nets between two blocks. If the number of overlapping pins is to large, resulting pin assignment after rounding can deviate significantly from LP solution and wire length estimations in LP formulation may no longer be accurate, and final pin assignment can be far from optimal.

To compromise between run time and quality of pin assignment, we propose a bus

bundling method to reduce the number of minimum pin pitch constraints needed. In most chip designs, it is common that nets from a bus in block b_i are all connected to the block b_j. For instance, address nets and data nets of a CPU can connect to the same memory block. Therefore it is desirable to place pins in a bus together. Assume we have a block with pins pin_{a1}, pin_{a2} from bus_a and $pin_{b1}, pin_{b2}, pin_{b3}$ from bus_b. In LP formulation, we constraint that pins from the same bus should have same solution, and that pitch between two buses is at least $\lceil \frac{1}{2}(|bus_a| + |bus_b|) \rceil \times minPinPitch$ such that enough space is reserved for pins during the rounding phase, where $|bus_a|$ and $|bus_b|$ are the sizes of two buses. Instead of adding minimum pin pitch constraints between every two pins, bus bundling constraints can be stated as

$$pin_{a1}\_x = pin_{a2}\_x \tag{2.72}$$

$$pin_{a1}\_y = pin_{a2}\_y \tag{2.73}$$

$$pin_{b1}\_x = pin_{b2}\_x \tag{2.74}$$

$$pin_{b1}\_y = pin_{b2}\_y \tag{2.75}$$

$$pin_{b1}\_x = pin_{b3}\_x \tag{2.76}$$

$$pin_{b1}\_y = pin_{b3}\_y \tag{2.77}$$

$$|pin_{a1}\_x - pin_{b1}\_x| + |pin_{a1}\_y - pin_{b1}\_y| \geq 3 \times minPinPitch \tag{2.78}$$

If we note the number of buses within a block as $|BUS|$, the number of bus bundling constraints is at the order of $O(|BUS|^2)$. Since $|BUS|$ can be much smaller than number of pins in a block, considerable reduction in number of constraints and run time can be achieved. For instance, if a block has 100 pins and in total 10 buses, the number of constraints for adding minimum pin pitch between every two pins is $100 \times 99 = 9900$, while only $10 \times 9 + 100 - 10 = 180$ for bus bundling constraints, 10×9 for bus pitch and $100 - 10$ for equality constraints among pins within the same bus.

2.5 Comparison of Different Formulation

In this section we present a comparison of ILP formulation, relaxed ILP formulation, and LP formulation described in section 2.2, 2.3 and 2.4.1. For LP formulation, we do not include minimum pin pitch constraints in Eq. 2.56 and bus bundling constraints described in Section 2.4.2. The experiment flow is shown in Fig. 2.3. We use OpenAccess [2] for the design database and IBM ILOG CPLEX Optimizer [3] for solving ILP and LP. The program is implemented in C++. With initial floorplan, we perform pin assignment and then use Cadence Innovus [1] to perform chip-level routing using only metal 6 and metal 7. A list of benchmarks we use is shown in Table 2.2. The program terminates either when pin assignment is finished, or run time is larger than 24 hours(TO). We compare both average wire length, max wire length and run time of three formulations and also results from Cadence Innovus. The results for the objective minimizing average wire length are shown in Table 2.3. Fig. 2.4 shows the resulting average Manhattan wire length normalized to Innovus results. We can see our formulation can achieve large improvement over some benchmarks where Innovus fails to produce an optimal solution. Fig. 2.5 shows the normalized results after routing. Still we can see for some benchmarks our formulation is close to Innovus and for others we can have large wire length reduction. In addition, we can see that ILP formulation can achieve the best results among all formulations since the results from ILP is exact and no rounding is needed. The trade-off for ILP formulation is the unaffordable run time for large designs, which fails to finish pin assignment for $des3$, $rockettile\_x2$ and $rockettile\_x4$. Also, relaxed ILP formulation outperforms LP formulation, but the run time is longer than LP formulation and can be 100X for some benchmarks. Overall we can see a trade-off between optimality and run time among these three formulations.

In addition, since we can easily impose maximum wire length constraints into our formulation, we can observe the effects of $maxWL$ on both final average and maximum wire length after pin assignment. We apply maximum wire length constraints in Eq. 2.14 on LP formulation for faster run time and test on aes_top benchmark. Since we use HPWL method to estimate wire length, the total net length after routing can be different from

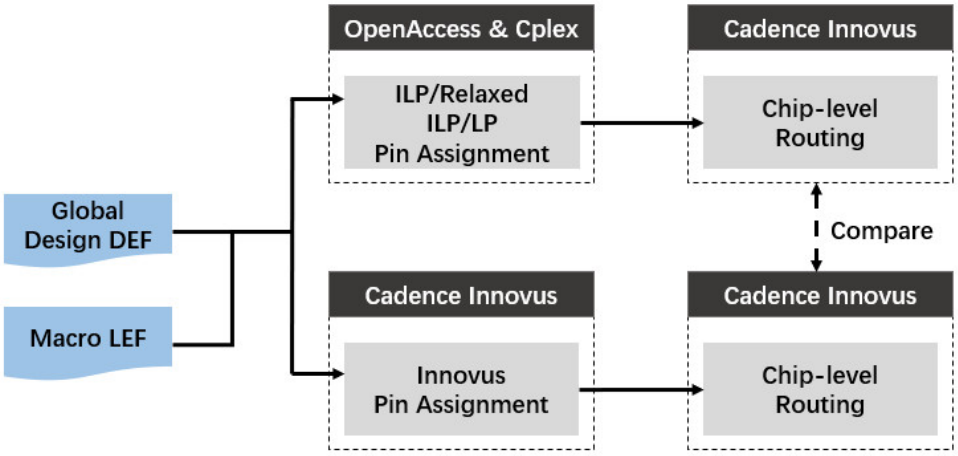


Figure 2.3: Experiment flow for comparing different formulation

Benchmark	**Die Size [um]x[um]**	**Blocks**	**Total Pins**
sbox_x4	145.7 x 145.0	4	64
aes_top	389.3 x 282.2	17	548
des3	649.3 x 626.6	3	564
rockettile_x2	4618.1 x 4194.1	2	1060
rockettile_x4	5670.93 x 5668.04	4	2120

Table 2.2: Benchmarks used for experiment

	Innovus/ILP/Relaxed ILP/LP		
Benchmark	Average WL [um]	Max WL[um]	Runtime [s]
sbox_x4	34.0/8.0/8.3/8.8	74.0/16.97/17.25/19.825	NA/2.5/0.3/0.1
aes_top	163.8/169.8/172.0/172.4	506.9/569.3/581.5/582.9	NA/69570.6/48.6/1.1
des3	327.7/NA/310.6/368.8	656.8/NA/686.6/786.6	NA/TO/19.9/1.5
rockettile_x2	2339.1/NA/303.2/409.3	656.8/NA/636.4/877.0	NA/TO/407.3/4.2
rockettile_x4	4244.4/NA/4258.1/4345.1	4797.1/NA/5944.5/6259.5	NA/TO/238.5/10.6

Table 2.3: Comparison between ILP, relaxed ILP and LP formulation

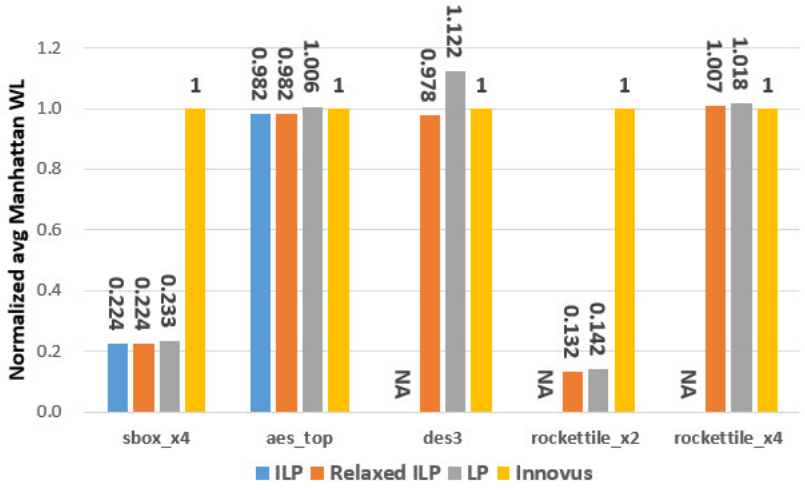


Figure 2.4: Normalized average net Manhattan length results for ILP, relaxed ILP, LP formulation and Innovus

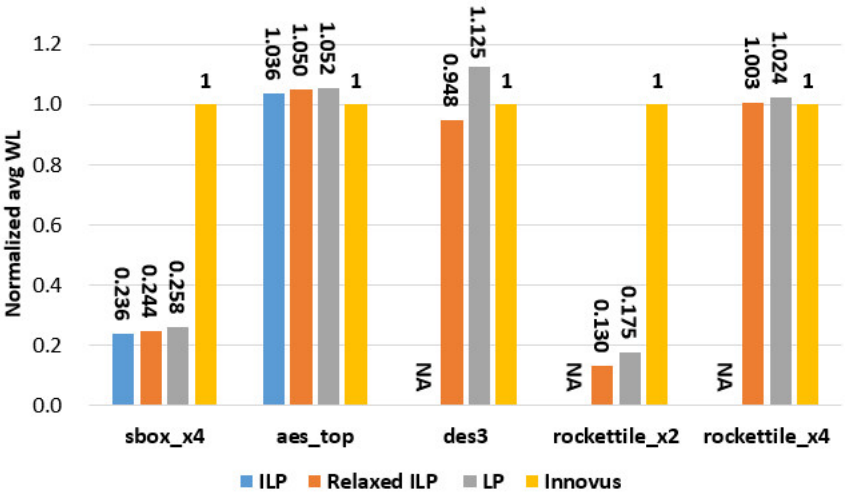


Figure 2.5: Normalized average wire length after routing for ILP, relaxed ILP, LP formulation and Innovus

estimated HPWL if more than two pins are associated with such net. However, HPWL is still an accurate estimate for the maximum wire length between two pins. We first run our LP without maximum wire length constraint and obtain the default estimated maximum HPWL, $HPWL\_max\_default$, in the design. We also obtain the default maximum and average wire length, $WL\_max\_default$ and $WL\_avg\_default$, after routing when $maxWL = HPWL\_max\_default$. Then we sweep down the value of $maxWL$ from $HPWL\_max\_default$ until the LP is not solvable. For benchmark aes_top, $HPWL\_max\_default = 415um$ and the solver fails to solve the LP when $maxWL = 385$. We sweep $maxWL$ from $415um$ to $385um$ in step of $5um$, then perform pin assignment, routing and obtain maximum and average wire length for each value of $maxWL$. We normalize each $maxWL$ as a ratio difference from $HPWL\_max\_default$. Intuitively, strict maximum wire length constraint can reduce maximum wire length but relax on average wire length. Thus we perform routing for each $maxWL$ and obtain the new maximum, average wire length as a reduction, increase ratio of $WL\_max\_default$ and $WL\_avg\_default$, respectively. The result is shown in Fig. 2.6. As expected, maximum wire length decreases and average wire length increases as we apply more strict maximum wire length constraint. Still, the value of $maxWL$ is more effective on reducing maximum wire length than degrading overall wire length. Thus our proposed model can effectively enforce the maximum wire length constraint, which is often associated with the timing constraints of certain critical paths, without significantly sacrificing overall performance of the system.

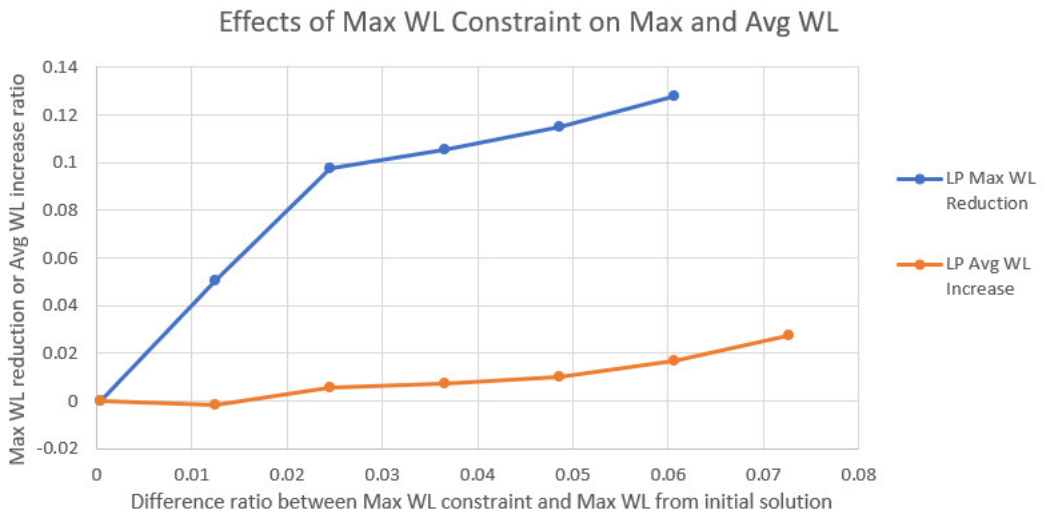


Figure 2.6: Effects of maximum wire length constraints on maximum and average wire length of routing results

CHAPTER 3

Interconnect Modeling and Power Optimization

In addition to interconnect length, another important metric for 2.5D integration is the power overhead. Due to the passive nature of silicon interposer, links between blocks cannot be buffered like links in SoCs. Thus, appropriate I/O cells with enough driving strength should be provided for interconnects. Large I/O cell can drive longer interconnect but also consumes more power. Therefore in this work, the objective for pin assignment is not solely minimizing wire length, but also minimizing the power required to drive all links. In this chapter we present our modeling of interconnects for extracting the relation between driver power, interconnect speed and wire length. For better run time, we modify our proposed LP formulation in Section 2.4.1 to minimize driver power.

3.1 Interconnect Modeling

The RC model first proposed in [5] has been widely used for resistive interconnects in integrated circuits since it is simple and can characterize wire delay with a reasonable accuracy. In this work we also use RC model to simulate the wire performance and estimate I/O power consumption. We obtain the RC model for each $10\mu m$ wire segment and lump them into the final RC network. We use the dimension and RC of metal layer 6 in NangateOpenCellLibrary_PDK 45nm technology library to calculate the wire RC. Following the definition of $RPERSQRU$, $CPERSQRDIST$ and wire width, $WIDTH$ in technology LEF file, for each wire segment, resistance, $R_{segment}$, and C, $C_{segment}$ can be calculated as

$$R_{segment} = RPERSQRU/WIDTH \times 10\mu m \tag{3.1}$$

$$C_{segment} = (CPERSQRDIST \times WIDTH + 2 \times CEDGE) \times 10\mu m \tag{3.2}$$

We lump the fringing capacitance at two ends of the wire into two terminal capacitance, C_{term}, which can be calculated as

$$C_{term} = WIDTH \times CEDGE \tag{3.3}$$

Similar to SoCs, silicon interposer based 2.5D integration suffers from electrostatic discharge (ESD) issues. To protect chips from damages caused by ESD stress, dedicated on-chip ESD protection circuits are widely used. On-chip ESD protection circuits are tested and rated using different ESD test models which simulate the ESD event according to its origins [36]. Several common models used are

1. Human body model (HBM) simulates the ESD event stimulated by a charged human body contacting and discharge through an electronic device.

2. Machine model (MM) simulates the ESD event when a charged machinery touch the device and discharge through the device during testing.

3. Charged-device model (CDM) simulates the ESD event when a charged device discharges though its grounded pins. The floating device can be charged up due to self induction during the manufacturing and assembly. When some pins of the charged IC touches an external ground, stored charge will discharge through grounded pins to the external ground [19] [39].

One of the most common ESD protection solution is to use dual diode concept shown in Fig. 3.1, where a positive stress on IO pad can discharge to VDD rail through the diode on the top, and a negative stress can discharge to ground through the diode at the bottom [6]. ESD protection circuits design is a major challenge in high frequency ICs due to the parasitic capacitance. Generally, to sustain a higher ESD stress level, stronger circuits are required, which introduce more parasitic capacitance [13]. As mentioned in [14], $100V - 500V$ HBM protection level is recommended for today's 2.5D/3D ICs for factory ESD control. An

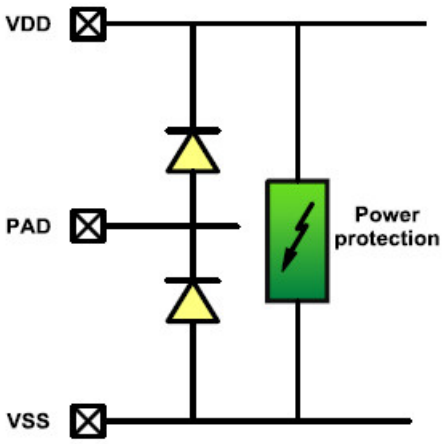


Figure 3.1: Dual diode ESD protection scheme[6]

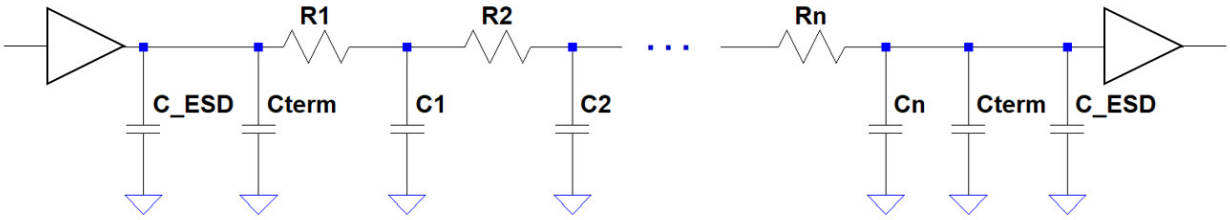


Figure 3.2: Wire model for simulating interconnect speed and power

optimized layout for ESD protection diodes is proposed in [38] and for $300V$ and $800V$ HBM level, the parasitic capacitance are $2.84fF$ and $8.01fF$, respectively. Therefore, for the interconnect model used in our work, we attach a $C_{ESD} = 5fF$ capacitor at each terminal to approximate the ESD protection overhead.

We model interconnect drivers, which are the I/O buffers of each macro block, and receivers as four back-to-back inverters and each inverter has fanout of 2. Therefore the complete interconnect model we use for simulating wire speed and driver power is shown in Fig. 3.2. We use energy-per-bit to evaluate the power efficiency of interconnects. SPICE simulations are performed to determine the energy required for I/O cells with different transistor size.

To optimize the power required for drivers after pin assignment, we simulate with our interconnect model to obtain a power look up table which specifies the power required to

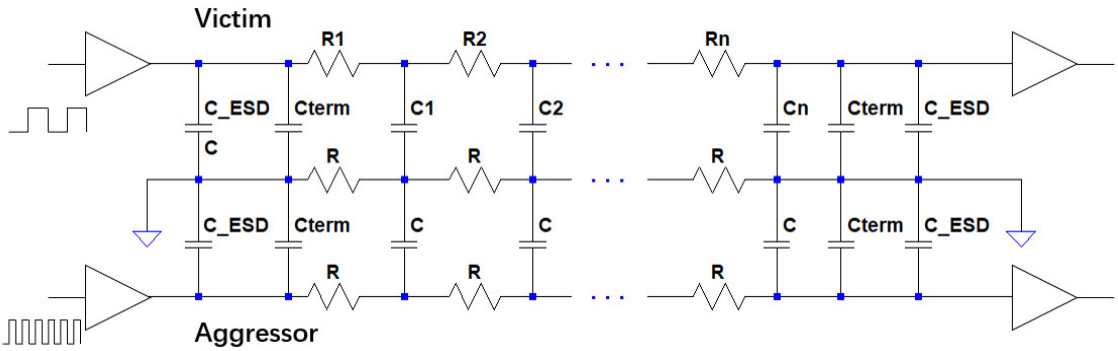


Figure 3.3: Wire model in presence of noise and crosstalk

driver a link with certain length and frequency requirement. We simulate different wire length from $10um$ to $7000um$. For each wire length, we sweep the driver size to obtain the power required for the link to run at different frequencies. Fig. 3.5 shows the relation between IO power and interconnect length and speed. The non-linear relation indicates that minimizing average wire length as in Chapter 2 cannot necessarily minimize power consumption if different interconnects are running at different frequency.

In today's sub-micron circuits design, the coupling capacitance between neighboring wires can be an important component in determining wire performance due to the small wire spacing. The coupling effects can cause extra signal delays or even logic malfunctions [12]. Therefore, in addition to speed and power simulation, we also simulate interconnect performance in presence of noise and wire crosstalk. Using the 2π model proposed in [12], we include the noise from an adjacent switching wire and also an non-ideal ground line. The model is shown in Fig. 3.3. For each entree in our obtained power look up table, we apply a switching signal with corresponding frequency at the victim wire, and a $10X$ faster signal at the aggressor wire. Then we check if the eye diagram has at least 80% opening. If not we increase the driver size until enough eye opening is obtained. An example eye diagram for $5000um$ wire running at $2.5GHZ$ with $25GHz$ noise is shown in Fig. 3.4

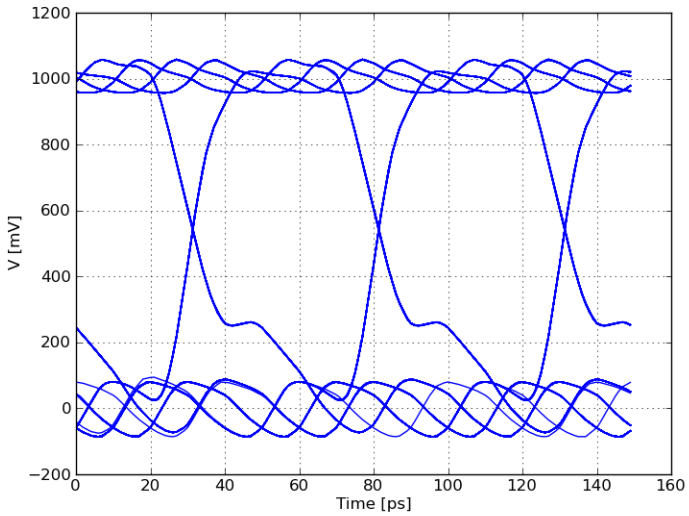


Figure 3.4: Eye diagram for a 5000um interconnect running at 2.5GHz

3.2 Power Optimization and LP formulation

As we have shown from the simulation results in Section 3.1, given the length of an interconnect, the power required to drive the link has a non-linear relation with the link speed. Thus, the objective of minimizing overall average wire length is not equivalent to minimizing total power consumption if different nets have different operating frequency. Therefore in this section we modify our LP formulation to optimize the integration power overhead.

The primary objective in our proposed LP pin assignment formulation in Chapter 2 is to minimize the average wire length. However, all the constraints we modeled are not specific to minimizing wire length, instead they are general constraints to ensure a valid pin assignment is achieved. Thus, to optimize the power consumption, we can keep all the constraints in Section 2.4.1 unchanged but only modify the objective function by applying proper weight to each net, $Wgt_i, i = 1, 2, ...t$, according to the maximum delay a net has to meet. We perform curve fitting using second order polynomial on the results shown in Fig. 3.5, where the independent variable is the inverse of required maximum delay, or target frequency, and dependent variable is the required IO power. Examples of fitted curves for wire length ranging from $10000\mu m$ to $20000\mu m$ are shown in fig. 3.5. Since different wire length exhibit

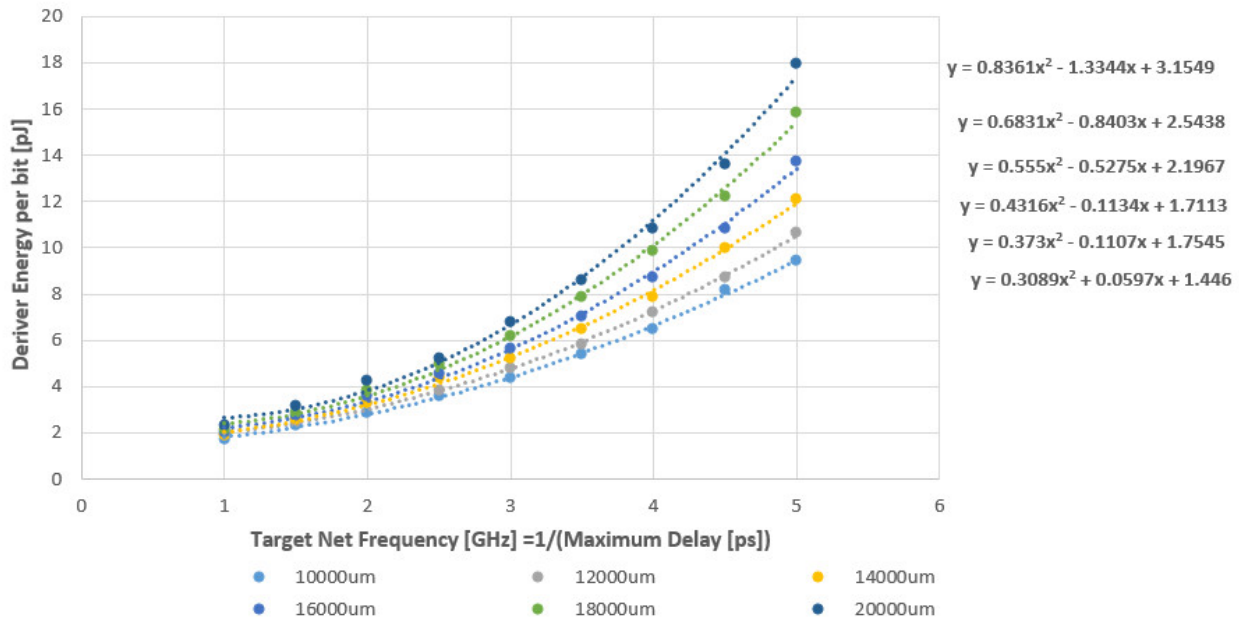


Figure 3.5: Fitted curves for interconnects with different length

different curves, and we have knowledge on the estimated wire length in the design only after pin assignment is done, in order to choose the proper function for weight calculation, we use the average distance between centers of two blocks as a premature estimation of average wire length before pin assignment is performed. Then we use the fitted function associated with the wire length which is closest to our premature estimation to calculate Wgt_i. The weight can be calculated as

$$Wgt_i = a \times f_i^2 + b \times f_i + c \tag{3.4}$$

where f_i is the target operating frequency that net i is to meet, and a, b, c are the coefficients of fitted function. Then we can modify the objective function of ILP/LP formulations to

$$Minimize: \sum_{i=1}^{t} Wgt_i \times WL_i \tag{3.5}$$

3.3 Experiment Setup and Results

We use the same benchmarks as the experiments in Chapter 2. In reality the required speed of certain nets depends on the type of interfaces or IP blocks they are associated with. For

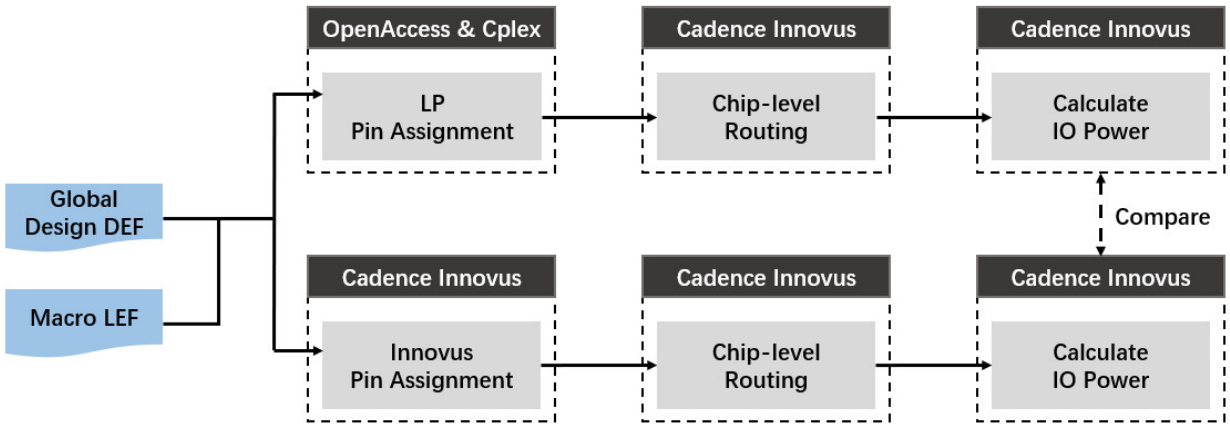


Figure 3.6: Experiment flow for power optimization

instance, a HBM (high bandwidth memory) interface are likely to be running at a lower frequency than a GDDR interface. Therefore for each block, we randomly group pins into buses to model different interfaces a chip has, and apply bus bundling constraints from Section 2.4.2. Then we randomly assign operating frequencies to each bus. We use the LP formulation in Section 2.4.1 in this part to apply bus bundling constraints and compare our results with Cadence Innovus. The experiment flow is shown in Fig. 3.6. Similarly to the setup in Section 2.5, given the initial floorplan, we perform pin assignment using both Innovus and our LP formulation. In Innovus we also apply same bus bundling constraints using *addPinGroup* command, and apply same priority, Wgt_i, for each pin. After the pin assignment is finished, we perform routing using Innovus for both pin assignment solutions. Finally we calculate and compare the total IO power required using the power look up table obtained in Section 3.1.

The results are shown in Table 3.1 and we can see improvement for all benchmarks from Innovus results. The small power reduction on *aes_top* may due to the large fanout of nets. Since in this work we use Manhattan distance to estimate interconnect length, estimation may not be accurate in case of multi-pin nets. Also, size of the benchmark is comparable to the resolution in obtained power look up table. Thus the small length reduction on nets may not reflect a step in the power look up table.

Benchmarks	**Innovus Energy/Bit**	**LP Energy/bit**	**Reduction**	**LP Runtime**
sbox_x4	145.99pJ	144.21pJ	1.22%	0.24s
aes_top	1042.76pJ	1042.62pJ	0.01%	1.27s
des3	1278.87pJ	1243.89pJ	2.73%	1.38s
rockettile_x2	1660.4385pJ	1315.70pJ	20.76%	771.32s
rockettile_x4	5882.06pJ	5652.64pJ	3.90%	587.76s

Table 3.1: Experiment results for energy optimization

CHAPTER 4

Multi-Floorplan Pin Assignment

As mentioned in Section 1.3, most of works on pin assignment focus on the design phase after floorplan or approach pin assignment and floorplan simultaneously. Therefore each pin assignment solution is customized for a specific design. However, modern SoCs usually contain hard IP blocks whose physical design are already finished by the IP vendor and cannot be modified by the designer [25]. Thus during the design of a hard IP block, a proper pin assignment is needed such that the IP block can be integrated into different systems with acceptable costs. In this chapter we propose a framework for multi-floorplan pin assignment, where we perform pin assignment to a block without the knowledge of floorplan. We also try to avoid potential routing hotspot and wire congestion by introducing pin redundancy during pin assignment, where multiple physical pins are assigned to a single logical pin.

4.1 LP Formulation

In this section we present our model and formulation for pin assignment without the knowledge of specific floorplan. Although floorplan is not available, we assume that we have the following knowledge or assumptions for the design:

1. System level netlist: During the development of an IP block, the possible system level netlist associated with the block is available since designers know the type of macros each interface should be connected to.

2. Estimation of sizes of other blocks: Since we know what types of blocks can be connected to the block under consideration, we can also estimate the size of those macros.

The number of pins connected to an block could be an inaccurate inference for the block size since some blocks, for example a memory block can have a large physical size but relatively small number of pins.

A rule of thumb for multi-floorplan pin assignment is to group pins corresponding to same interface together, and spread pins around the perimeter of the block. Therefore in the first step we modify the LP formulation in Section 2.4.1 with bus bundling constraints to assign a location for each bus. Then we apply rounding heuristics to allocate pins within each bus around assigned bus location.

To spread buses around the perimeter of the block under consideration, we modify the bus bundling constraints in Section 2.4.2 such that the pitch between buses is determined not only by the number of pins in each bus, but also related to the the total number of available positions and area of the connected block. Assume for a block b_k, we have buses $bus_0, bus_1, ..., bus_n$, where pins in a bus are connected to the same block. $|bus_i|$ is the number of pins in bus_i. For each bus, we assume that the area of the connected block is $A_i, i = 0, 1, ..., n$. We define the weight for each bus to be

$$BusWgt_i = \frac{A_i \times |bus_i|}{\sum_{j=1}^{n} A_j \times |bus_j|} \tag{4.1}$$

We allocate a number of available pin positions to each bus based on its weight and apply bus pitch constraints accordingly. Following the notation in Chapter 2 where $|Pos_k|$ is the number of available pin positions on block b_k, the number of positions that need to be reserved for bus_i can be calculated as

$$bus\_rsrvPos_i = max(\lfloor BusWgt_i \times |Pos_k| \rfloor,\ |bus_i|) \tag{4.2}$$

where $|bus_i|$ is the minimum number of positions needed to assign pins in bus_i. Since the total number of positions assigned to all the buses can be larger than $|Pos_k|$ if one or more of $bus\_rsrvPins_i$ are $|bus_i|$, we keep the buses with minimum number of locations reserved unchanged while truncating the number of reserved locations for other buses until the total number of reserved locations is less than or equal to $|Pos_k|$. Following the example in Section

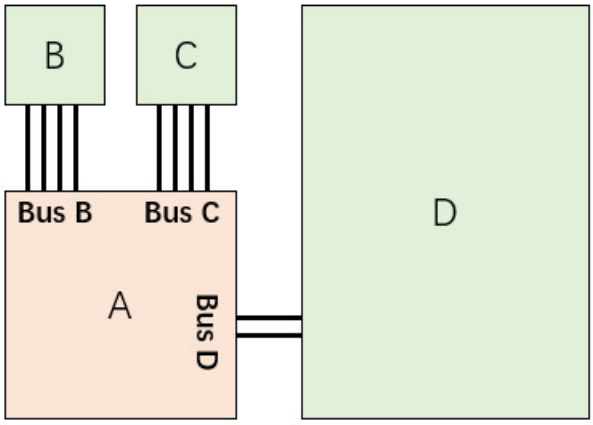


Figure 4.1: Effects of block sizes on pin assignment

2.4.2, assume we have a block with pins pin_{a1}, pin_{a2} from bus_a and $pin_{b1}, pin_{b2}, pin_{b3}$ from bus_b, we can add bus bundling constraints as

$$pin_{a1}\_x = pin_{a2}\_x \tag{4.3}$$

$$pin_{a1}\_y = pin_{a2}\_y \tag{4.4}$$

$$pin_{b1}\_x = pin_{b2}\_x \tag{4.5}$$

$$pin_{b1}\_y = pin_{b2}\_y \tag{4.6}$$

$$pin_{b1}\_x = pin_{b3}\_x \tag{4.7}$$

$$pin_{b1}\_y = pin_{b3}\_y \tag{4.8}$$

$$|pin_{a1}\_x - pin_{b1}\_x| + |pin_{a1}\_y - pin_{b1}\_y| \geq \lceil \frac{bus\_rsrvPos_a + bus\_rsrvPos_b}{2} \rceil minPinPitch \tag{4.9}$$

Area of the block connected to a bus is not trivial to pin assignment since it can affect the resulting floorplan, which in turn determines the quality of pin assignment. For example, as shown in Fig. 4.1 where pins of block A are being assigned, since the size of block D is much larger than other blocks, block D may "cover" one entire side of block A after the floorplan. Thus it is desirable to reserve more space for bus D even though bus B and C have more pins.

Because pin assignment is performed without the knowledge of floorplan, we cannot

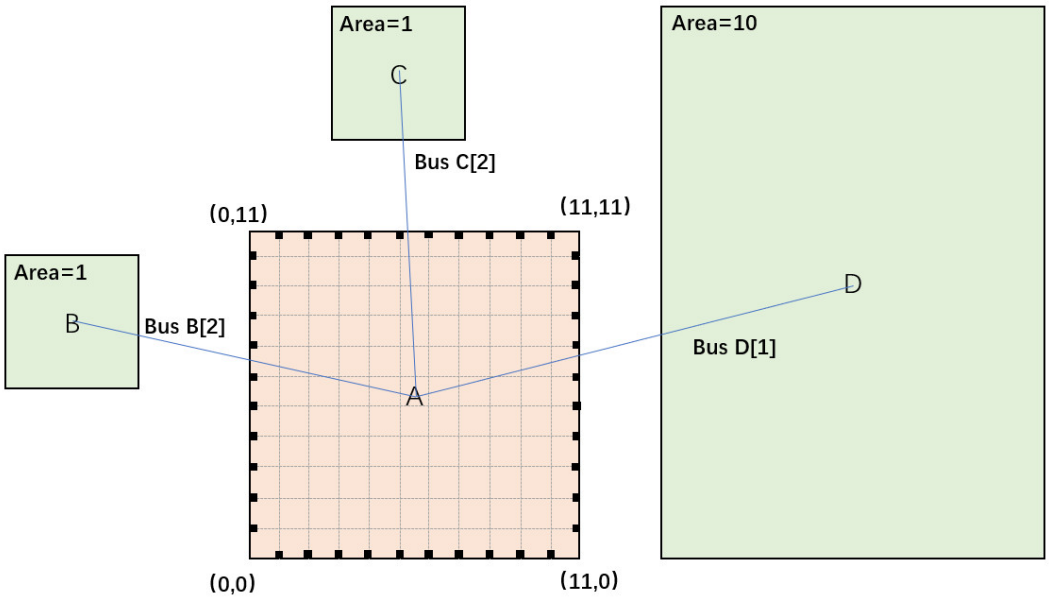


Figure 4.2: Multi-floorplan pin assignment for a block connected to three other blocks

estimate the resulting interconnect wire length and we cannot use the same objective function as in Chapter 2. Besides, since we have reserved space for each bus during construction of bus bundling constraints, additional objective to spread the bus around the perimeter of the block is not needed. Therefore we can formulate this multi-floorplan pin assignment into a LP without an objective function. We only need a valid solution from the LP solver.

Fig. 4.2 shows an example for multi-floorplan pin assignment for block A. Block A has three buses bus_B, bus_C, bus_D with sizes 2, 2, and 1, respectively, and pins associated with each bus are $pin_{b1}, pin_{b2}, pin_{c1}, pin_{c2}$, and pin_{d1}. The locations of blocks in Fig. 4.2 does not represent an actual floorplan but only illustrate the system level netlist. The area for block B, C and D are 1, 1 and 10. There are 40 pin positions on the perimeter of block A marked by black squares, with $minPinPitch = 1\mu m$. Using Eq. 4.1, we can get that

$$BusWgt_B = \frac{1}{7}, \quad BusWgt_C = \frac{1}{7}, \quad BusWgt_D = \frac{5}{7}$$

Then the number of reserved pin locations for each bus can be obtained as

$$bus\_rsrvPos_B = max(\lfloor \frac{1}{7} \times 40 \rfloor,\ 2) = 5$$

$$bus\_rsrvPos_C = max(\lfloor \frac{1}{7} \times 40 \rfloor,\ 2) = 5$$

$$bus\_rsrvPos_D = max(\lfloor \frac{5}{7} \times 40 \rfloor,\ 1) = 28$$

The total number of reserved locations is $5 + 5 + 28 = 38 < 40$ so none of the buses should be truncated. Therefore the complete LP pin assignment for block A can be formulated as follows:

$$Minimize: \tag{4.10}$$

$$0 \tag{4.11}$$

$$Subject\ to: \tag{4.12}$$

$$0 \leq pin_{b1}\_x \leq 11, \quad 0 \leq pin_{b1}\_y \leq 11 \tag{4.13}$$

$$0 \leq pin_{b2}\_x \leq 11, \quad 0 \leq pin_{b2}\_y \leq 11 \tag{4.14}$$

$$0 \leq pin_{c1}\_x \leq 11, \quad 0 \leq pin_{c1}\_y \leq 11 \tag{4.15}$$

$$0 \leq pin_{c2}\_x \leq 11, \quad 0 \leq pin_{c2}\_y \leq 11 \tag{4.16}$$

$$0 \leq pin_{d1}\_x \leq 11, \quad 0 \leq pin_{d1}\_y \leq 11 \tag{4.17}$$

$$pin_{b2}\_x = pin_{b1}\_x, \quad pin_{b2}\_y = pin_{b1}\_y \tag{4.18}$$

$$pin_{c2}\_x = pin_{c1}\_x, \quad pin_{c2}\_y = pin_{c1}\_y \tag{4.19}$$

$$|pin_{b1}\_x - pin_{c1}\_x| + |pin_{b1}\_y - pin_{c1}\_y| \geq 5 \times 1 \tag{4.20}$$

$$|pin_{b1}\_x - pin_{d1}\_x| + |pin_{b1}\_y - pin_{d1}\_y| \geq 17 \times 1 \tag{4.21}$$

$$|pin_{c1}\_x - pin_{d1}\_x| + |pin_{c1}\_y - pin_{d1}\_y| \geq 17 \times 1 \tag{4.22}$$

A valid solution for the LP is shown in Fig. 4.3, where bus locations are marked by colored squares.

4.2 Rounding Heuristic

LP formulation will assign all pins in a bus to one location on the perimeter based on bus size and area of corresponding block. In this section we present our heuristic to round the LP solution and obtain a valid pin assignment. In the first step we assign a slot of pin positions to each bus and slot size is roughly equal to $bus\_rsrvPos$. In the second step we spread pins

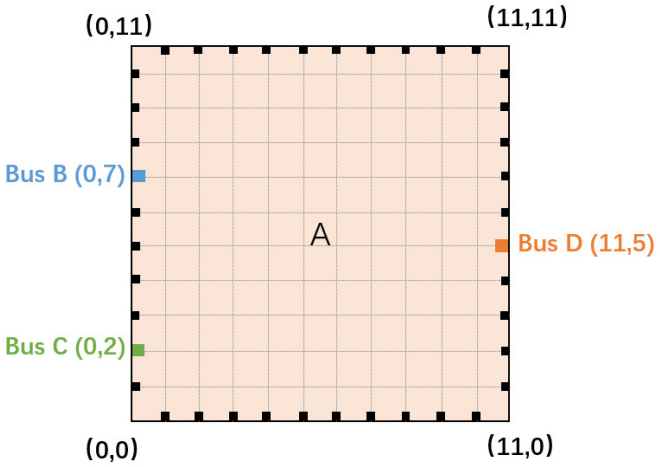


Figure 4.3: A valid LP formulation solution for the problem shown in Fig. 4.2

in a bus evenly over the corresponding slot.

We note the coordinate assigned to a bus as (bus_i_x, bus_i_y). The algorithm for assigning a slot for a bus in block b_k is shown in Algorithm 2. It takes the bus locations as inputs and outputs a vector of positions, $slot_i$, for each bus. We calculate a maximum distance, $maxDis_i$, from the bus location that a position can be allocated into a bus's slot. The equation for calculating $maxDis_i$ is shown in line 3 in Algorithm 2. After we add all pin positions within $maxDis_i$ to $slot_i$, we sort $slot_i$ such that the positions around the block perimeter are added into $slot_i$ clockwise. The algorithm for spreading pins in buses along their slots for block b_k is shown in Algorithm 3, which takes the slots for all buses as inputs and update pin locations. The complete algorithm for rounding the LP solution is shown in Algorithm 4.

The rounding procedure for the example shown in Fig. 4.3 is illustrated in Fig. 4.4. Assigned slot for each bus marked with color is shown in left figure and the final pin assignment is shown in the right figure.

Algorithm 2 Assign bus slot and quantize each slot

function AssignBusSlots(b_k, (bus_i_x, bus_i_y), $i = 1, 2, ..., n$)
  for each bus_i in b_k **do**
    $maxDis_i = \frac{1}{2} bus\_rsrvPins_i \times MinPinPitch$
    for each pos in Pos_k **do**
      if $|pos\_x - bus_i\_x| + |pos\_y - bus_i\_y| \leq maxDis$ **then**
        $slot_i.pushback(pos)$
    sort($slot_i$)
  return $slot_i, i = 1, 2, ..., n$

Algorithm 3 Spread pins in a bus along the slot

function SpreadPinsInSlot(b_k, $slot_i$, $i = 1, 2, ..., n$)
  for each bus_i in b_k **do**
    posPerPin = $\lfloor \frac{|slot_i|}{|bus_i|} \rfloor$
    $j = 0$
    for each pin in bus_i **do**
      $(pin\_x, pin\_y) = slot_i[j]$
      $j+ = posPerPin$
  return pin_{iq}_x, pin_{iq}_y, $1 = 1, 2, ..., |bus_i|$, $i = 1, 2, ..., n$

Algorithm 4 Complete algorithm for rounding LP solution

Input: Block b_k, (bus_i_x, bus_i_y), $i = 1, 2..., n$

Output: pin_{iq}_x, pin_{iq}_y, $1 = 1, 2, ..., |bus_i|$, $i = 1, 2, ..., n$

for each block b_k **do**
  $slot_i, i = 1, 2, ..., n$ = AssignBusSlots(b_k, (bus_i_x, bus_i_y), $i = 1, 2..., n$)
  SpreadPinsInSlot(b_k, $slot_i, i = 1, 2, ..., n$)

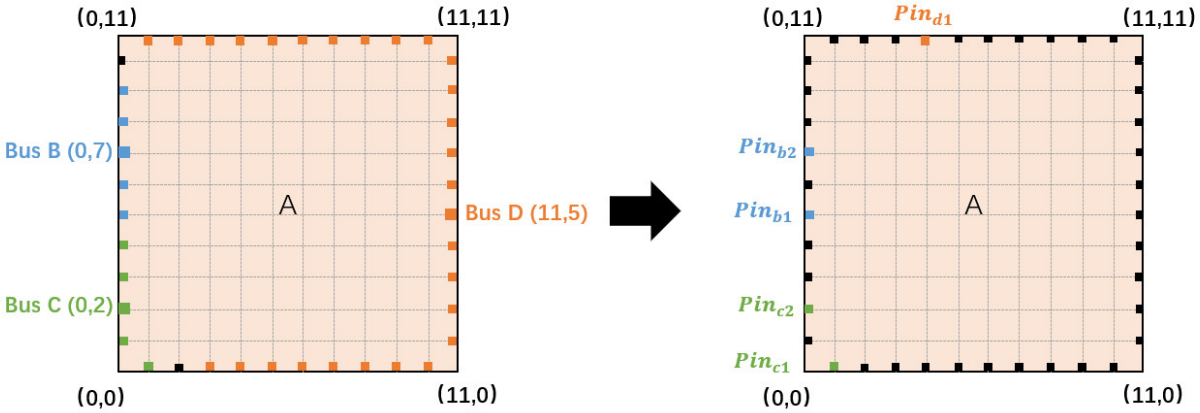


Figure 4.4: Rounding results for example shown in Fig. 4.3.

4.3 Pin Redundancy

Since pin assignment is performed without the knowledge of floorplan, we can reduce the probability of wire crossing by bundling pins connected to same macro together and spread pins around the perimeter of the block under consideration. Still, without performing pin assignment globally, in some cases resulting pin assignment can cause potential routing hotspot, which eventually increase the interconnect length. For example, consider an interface with multiple nets between two blocks, $b1$ and b_2, as shown in Fig. 4.5. Multi-floorplan pin assignment has been applied on both block $b1$ and b_2, and system-level floorplan and placement has been performed. Since pin assignment is performed locally on each block, even though placement tool can place these two blocks optimally such that two interfaces face each other, one interfaces can have a reversed ordering with respect to the other, which can cause wire congestion or even unroutable design.

In 2.5D integration, due to the passive nature of silicon interposer, a signal cannot be buffered once it leaves the block and the interconnect length is limited. Pin redundancy allows such net to be routed internally and buffered through the source macro and the net would exit through the physical pin that is nearest to the destination. For the multi-floorplan pin assignment flow we proposed, in terms of routability of the final floorplan, the worst case scenario is two buses facing each other but with reversed ordering as shown in Fig. 4.5. In

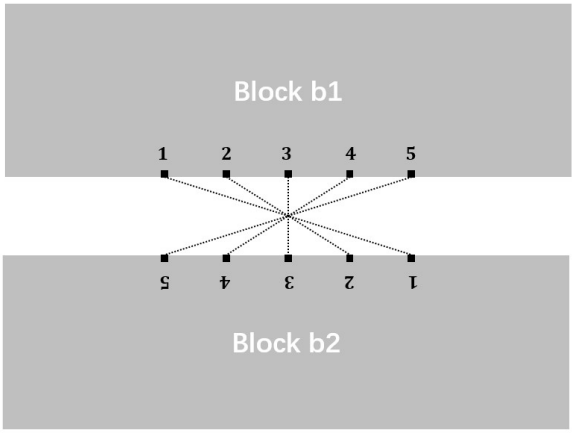


Figure 4.5: Pin assignment performed locally can cause wire congestion or routing hotspot in the final floorplan

this section we present a multi-pin placement method where we add extra physical copy of a logical pin to alleviate the worst case chip-level routing scenarios and potential routing congestion.

The scenario depicted in Fig. 4.5 can be modeled as a channel routing problem (CRP). In Manhattan model of channel routing, a channel consists of 2-layer of rectangular grid of columns and rows (tracks) and pins are located at the top and bottom tracks. The objective of such channel routing problem is to connect pins of each net while minimizing the number of tracks required. Wires can be routed in either layers of tracks between top and bottom tracks and in either layer of columns [31]. The density of the channel routing problem, d_{max}, is defined as the maximum number of nets that are split by any vertical cut across the channel, where one pin is located on one side of the cut and the other pin is located on the other side. If we note the number of nets in the problem as n_{net}, the worst case density of the problem can be $d_{max} = n_{net}$. It is also obvious that a trivial lower bound for the number of tracks needed to route all the nets is d_{max} [28]. Also, if we denote l as the length of the longest path of the vertical constraint graph of the channel, a well-know loose lower bound on the number of tracks is $max\{d_{max}, l\}$[42]. An upper bound for the number of tracks needed for a channel density d_{max} is $2d_{max} + O(f)$, where f is the flux of CRP. However, in practice $2d_{max} + O(1)$ tracks is usually needed since f is small and bounded by

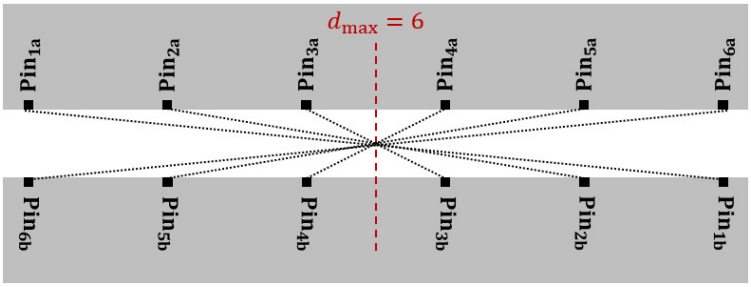
a small constant in practical problems [31]. In the 2.5D integration where routing between different dies is performed on the substrate, the number of pins associated with a channel between two blocks can be significantly larger than a small constant. Thus, if we note t as the number of tracks needed, we can roughly say that it is bounded by

$$t \leq 2 \times d_{max} \tag{4.23}$$

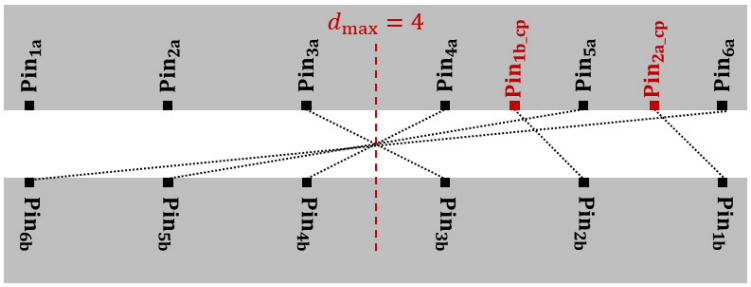
In a typical design of 2.5D integration interposer, 2500 die-to-die interconnects can be routed in a single layer at a signal bus with width of 25mm. Also, a gap between different dies with width at least 100μm is usually maintained in the chip level assembly [33]. Thus in this section we can make the assumption that around 10 tracks are available for routing in the channel between two blocks. Since in the worst case $d_{max} = n_{net}$, following the upper bound of in Eq. 4.23, we can get $n_{net} \geq 5$, which means in the worst scenario at least 5 nets can be routed in the channel between two blocks. In other words, $d_{max} \leq \frac{t}{2}$ should be satisfied in order to make the channel routable in the scenario shown in Fig. 4.5. Thus, in order to allow more nets to be routed in the channel, we introduce pin redundancy to decrease the maximum density in the channel. For example, consider a channel shown in Fig. 4.6a which consists of 6 nets and 6 pins on each side of the channel. The vertical cut shown in red has the maximum density 6, since all 6 nets have one pin on the left side and the other on the right side of the cut. To reduce the channel density, we can duplicate a pin and place it onto the other side of the vertical cut. For instance, in Fig. 4.6b we make extra physical copies of Pin_{1a} and Pin_{2a} and place on the right side of the cut, which are labeled as Pin_{1a_cp} and Pin_{2a_cp}. The maximum density of the channel becomes 4 as net_1, net_2 are located on one side of the cut and different copies of the same pin are internally connected through their macro block. Therefore, one observation we have is that we can keep making redundancy for each pin to the other side of the cut until $d_{max} \leq \frac{t}{2}$ is satisfied. d_{max} can be estimated as

$$d_{max} = d_{max\_original} - \#\ pin\ redundancy \tag{4.24}$$

where $d_{max\_original}$ is the channel density before any pin redundancies are made. In the example shown in Fig. 4.6a, 1 pin needs to be duplicated in order to make the channel



(a)



(b)

Figure 4.6: (a) A channel with 6 nets and maximum density 6. (b) After making physical copies for two pins the density is reduced to 4.

routable. Thus, in addition to the rule of thumbs for multi-floor pin assignment proposed in Section 4.1 and Section 4.2, after we spread pins along a bus slot, we make physical copy for each pin and place in reversed order until d_{max} is less than $\frac{t}{2}$. If we note $t_{channel}$ as the number of tracks available in the channel and pin_{iq_cp} as the duplicate of pin_{iq}, we modify Algorithm 3 and the complete algorithm for spreading and duplicating pins is shown in Algorithm 5.

4.4 Experiment Flow and Results

In this section we present two experiment flows to evaluate the quality of multi-floorplan pin assignment we proposed. In the first flow we compare multi-floorplan results with LP formulation presented in Section 2.4.1 for a specific floorplan. In the second flow we generate 20 random floorplans for each benchmark, and apply one single multi-floorplan pin assignment and LP assignment on all floorplans to compare the average and the worst case quality.

Algorithm 5 Spread pins in a bus and add pin redundancy along the slot

function SPREADPINSANDPINREDUNDANCY(b_k, $slot_i$)
  for each bus_i in b_k **do**
    $posPerPin = \lfloor \frac{|slot_i|}{|bus_i|} \rfloor$
    $j = 0$
    for each pin in bus_i **do**
      $(pin\_x, pin\_y) = slot_i[j]$
      $j+ = posPerPin$
    $RedundancyNeeded = |bus_i| - \frac{1}{2}t_{channel}$
    $new\_slot =$ all unoccupied pos in $quantized\_slot_i$
    $j = |new\_slot| - 1$
    $newPosPerPin = \lfloor \frac{|new\_slot|}{|bus_i|} \rfloor$
    $RedundancyCount = 0$
    for each pin in bus_i **do**
      if $d_{max\_original} - RedundancyCount \leq \frac{1}{2}t_{channel}$ OR $j < 0$ **then**
        break
      copy pin to $new\_slot[j]$
      $j- = newPosPerPin$
  return $pin_{iq}\_x, pin_{iq}\_y, pin_{iq\_cp}\_x, pin_{iq\_cp}\_y$, $i = 1, 2, ..., |bus_i|$, $i = 1, 2, ..., n$

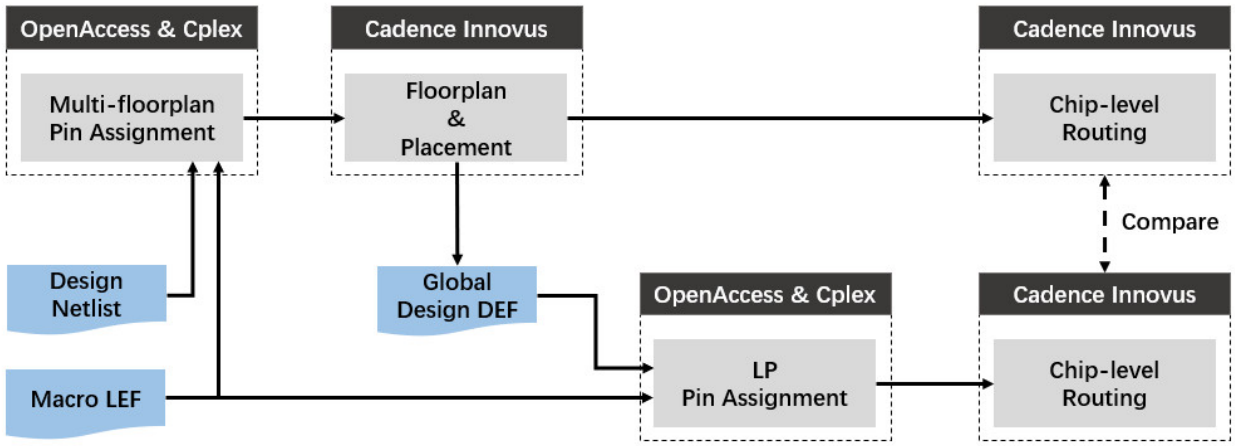


Figure 4.7: Experiment flow for comparing multi-floorplan and LP pin assignment for a given floorplan.

In the first experiment flow, we compare the final interconnect length after routing with the floorplan-specific LP formulation in Section 2.4.1. For each benchmark, we perform multi-floor pin assignment based on the system-level netlist, then perform floorplan and routing using Cadnece Innovus. We then perform LP pin assignment specific to the floorplan generated by Innvous, and perform routing again. The routing results of these two pin assignments are compared. The experiment flow is shown in Fig. 4.7.

Since the multi-floorplan pin assignment is performed without the knowledge of the floorplan while LP formulation assigns pins globally for a specific floorplan, it is expected that multi-floorplan assignment will result in longer average interconnect length than LP assignment. We also compare the effectiveness of pin redundancy by making different assumptions about the die spacing and the number of routing tracks available in a channel between two blocks. We compare the results of assuming unlimited tracks, 20 tracks, 10 tracks and "0 tracks". No pin redundancy is made for unlimited tracks and all pins are duplicated for 0 track. We normalize all results to the average wire length of LP formulation and the comparison is shown in Fig. 4.8. We can notice that pin redundancy can effectively decrease the resulting average interconnect length. *sbox_x*4 does not show improvement from unlimited track assumption to 10 track assumption because the benchmark size is small and no pin

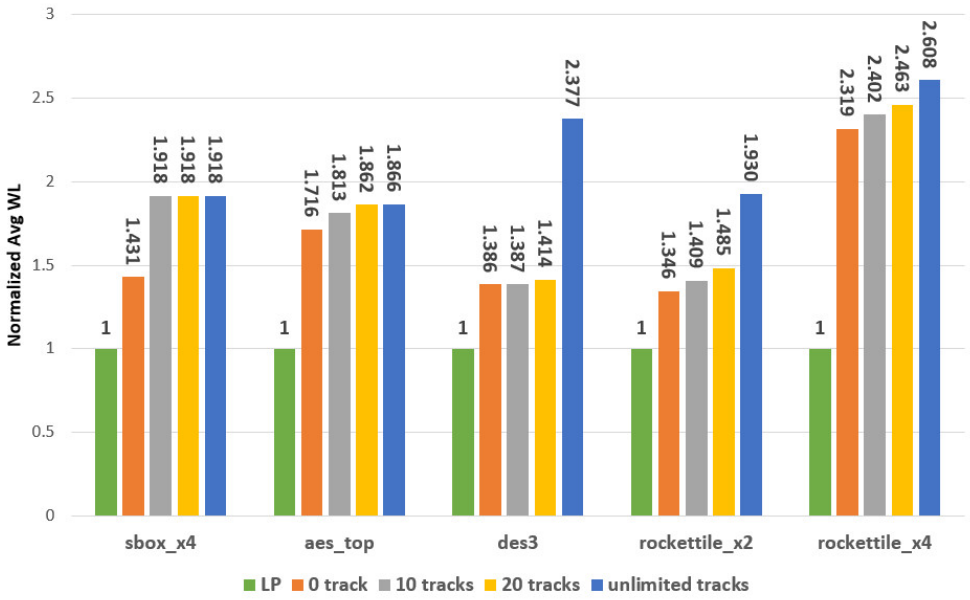


Figure 4.8: Comparison between multi-floorplan and LP pin assignment with different available channel routing track assumptions.

redundancy is needed. Comparing with LP results shows a "lower bound" on the performance of our proposed multi-floorplan pin assignment on a specific floorplan. We can see that for *rockettile_x4* the integration overhead is at most 2.6X without any pin redundancy and 2.4X for 10-track assumption. For other benchmarks overhead are all below 2X with pin redundancy.

Since the purpose for multi-floorplan pin assignment is to propose a rule of thumb for pin assignment without the knowledge of floorplan such that the same block can be reused in different designs, in the second flow we compare the average and worst case performance of a multi-floorplan assignment and LP assignment across a set of random floorplans. For each benchmark, we randomly generate 20 floorplans, $fp_0, fp_1, ...fp_{19}$. To generate random floorplans for a benchmark, we first construct different placement schemes for all blocks. For example, *des*3 benchmark has 3 blocks, $b1$, $b2$, and $b3$, and we construct 4 different placement schemes as shown in Fig. 4.9, where block $b1$, $b2$, and $b3$ can be placed into slot *slot_a*, *slot_b* and *slot_c* in any sequence. Each block can have 4 orientations, *north*, *south*, *west*, and *east*. For each floorplan, we randomly choose a placement scheme, placement sequence, and

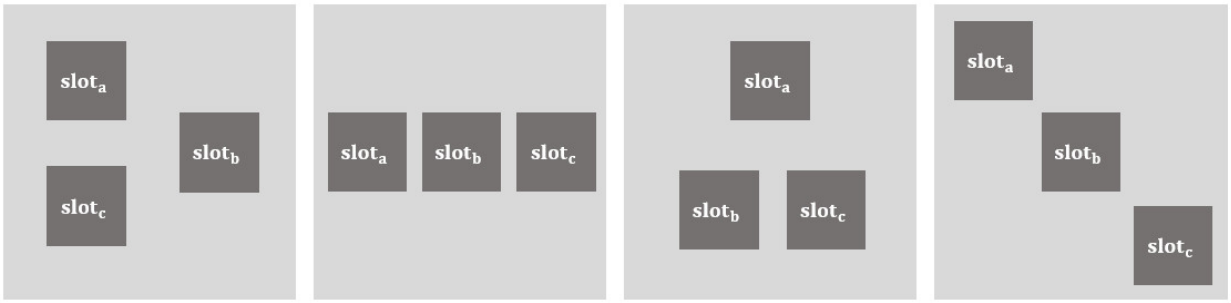


Figure 4.9: Four placement schemes for des3 benchmark used to generate random floorplans.

an orientation for each block.

After random floorplans are generated, a LP pin assignment, PA_{LP}, is generated using fp_0, and multi-floorplan pin assignment, PA_{multi} is generated based on the netlist. We apply PA_{LP} and PA_{multi} to all floorplans and perform routing to get the average wire lengths, $\{avgWL_{LP}\_1, avgWL_{LP}\_2, ..., avgWL_{LP}\_19\}$ and $\{avgWL_{multi}\_1, avgWL_{multi}\_2, ..., avgWL_{multi}\_19\}$. Finally the average and worst case of all $avgWL_{LP}\_i$ and average of all $avgWL_{multi}\_i$ are compared. The experiment flow is shown in Fig. 4.10 and we repeat the same flow for each channel width assumption. The normalized results for worst case average wire length of all floorplans are shown in Fig. 4.11. We can see that multi-floorplan pin assignment can effectively reduce the worst case wire length across the 19 floorplans. A maximum 26% reduction is achieved without any pin redundancy and 48% reduction can be achieved with a 10-track assumption. Two different pin assignment would have similar average performance across a set of random floorplans. A comparison of the average of average wire length of all floorplans are shown in Fig. 4.12. Still without pin redundancy we can see a small reduction for most of benchmarks, and a maximum of 40% reduction with 0-track assumption.

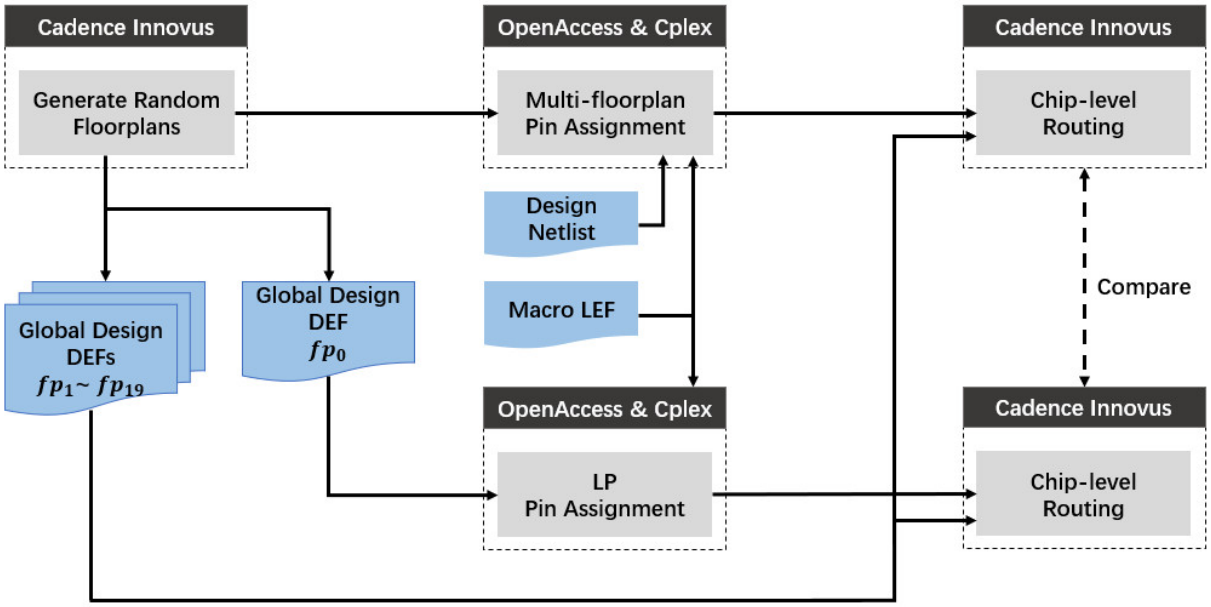


Figure 4.10: Experiment flow for comparing multi-floorplan and LP pin assignment using random floorplans.

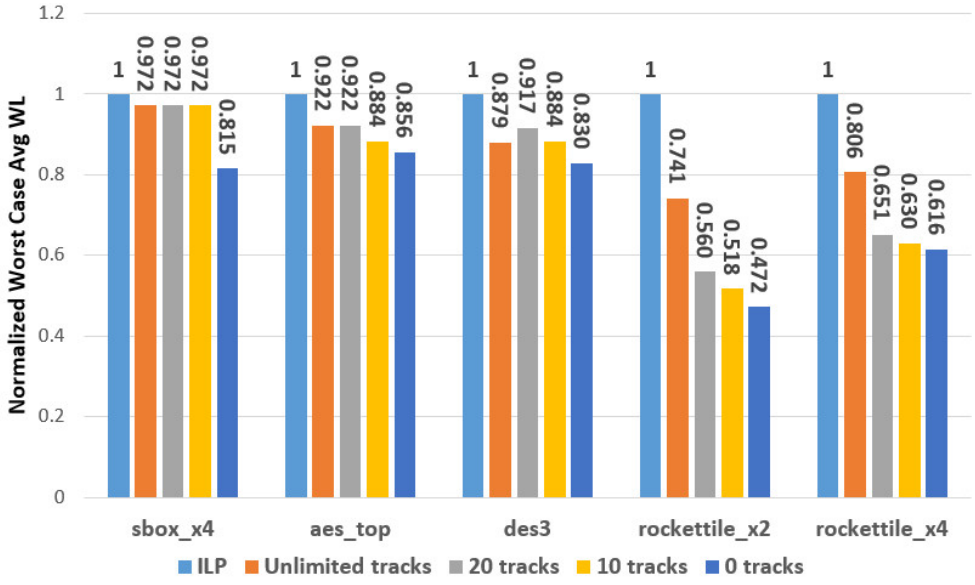


Figure 4.11: Comparison between multi-floorplan and LP pin assignment on the worst case performance across random floorplans.

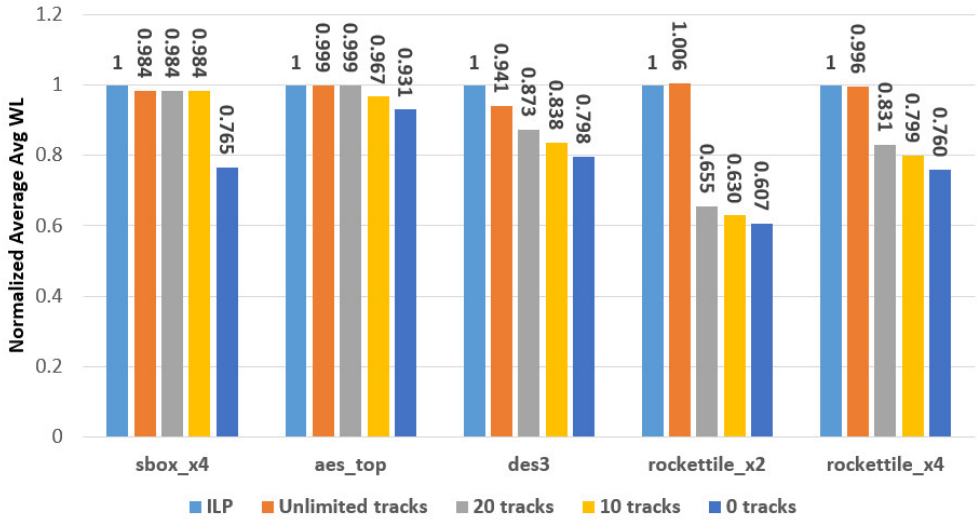


Figure 4.12: Comparison between multi-floorplan and LP pin assignment on average performance across random floorplans.

CHAPTER 5

Conclusion

Silicon interposer based 2.5D integration has become an alternative to traditional 2D monolithic integration as a result of increasing cost and decreasing efficiency for traditional process scaling. It allows high yield and high band width through die partitioning, and also facilitates heterogeneous integration. In a 2.5D process, dies are placed on the interposer and connected by interconnects in the interposer. In this work focus on minimizing the integration overhead through die-level pin assignment.

In Chapter 2 we develop three ILP/LP based formulations for hierarchical pin assignment where multi-instantiated blocks of the same cell have same pin assignment. We first present a ILP based formulation which uses an integer assignment matrix to map each pin to a legal position. Constraints are applied on the assignment matrix to ensure a valid pin assignment solution is generated. Since ILP is a NP-hard problem, the run time for directly solving ILP is not scalable for large designs. Thus we propose a relaxation method for ILP and present relaxed-ILP formulation. We relax the integer assignment matrix into a non-integer matrix and other constraints on the matrix are preserved. As the pin assignment results can be invalid if assignment matrix is no longer integer, we propose a rounding heuristic to round the LP solution into a valid pin assignment. Finally we present a runtime-scalable LP formulation by further reducing the complexity of the LP. We simplify the modeling by removing assignment matrix and only constraint pins to be inside the block bounding box. Rounding heuristic is then applied to legalize the solution. We also present bus bundle constraints to place pins from the same bus together. We show that our formulation can achieve large improvement over some benchmarks where Cadence Innovus fails to obtain the optimal solution. In addition, these three formulations show a trade-off between optimality and

run-time, where ILP formulation can achieve the most optimal results and LP formulations is most run-time efficient.

In Chapter 3 we present a framework of pin assignment for I/O power optimization. In a passive silicon interposer, a signal cannot be buffered once it leaves its source block. Thus the I/O cell of a die must have enough strength to drive the interconnect to meet system level timing requirements. The I/O power consumption is also a major overhead for 2.5D integration. We use a RC wire model to simulate the I/O power required as a function of interconnect length and speed and obtain a power look up table to evaluate power consumption after routing. We show that an non-linear relation between power and link speed exists thus minimizing wire length is not equivalent to minimizing I/O power. LP formulation is modified to optimize power consumption. We add random running frequencies to nets and perform pin assignment. Our results show that we can achieve a better power performance compared to Cadence Innovus.

In Chapter 4 we present a multi-floorplan pin assignment flow where pin assignment is performed without the knowledge of floorplan. For a block to be integrated into a design, floorplann-specific pin assignment can optimize the integration overhead for a given floorplan, however the pin assignment is not optimal for other floorplans and the block needs to be redesigned. On the other hand, multi-floorplan pin assignment assigns pins based on the possible system level netlist such that the block can be reused in different designs. We also propose a method to introduce pin redundancy into the design to avoid possible routing hotspot and further reduce interconnect length. We show that the overhead for a floorplan-unaware pin assignment is at most 2.6X that of the floorplan-specific pin assignment. In addition, compared to a pin assignment solution for a specific floorplan, we show that our flow achieves a better average and worst case performance for a set of random floorplans. In other words, our proposed multi-floorplan pin assignment can reduce the integration overhead if a single block design needs to be reused in multiple unknown designs.

In conclusion, in this work we present different flexible formulations for 2.5D die level pin assignment which are more optimal than Cadence Innovus. Also a multi-floorplan pin

assignment flow is proposed to allow the reuse of blocks. In the future we can model buffer bank allocation into pin assignment formulation, which allows signal to be buffered through a block before reaching to its destination. We can also develop a flow to integrate pin assignment and floorplan or routing for 2.5D or 3D integration.

Bibliography

[1] Cadence Innovus Implementation System https://www.cadence.com/content/cadence-www/global/en_US/home/tools/digital-design-and-signoff/hierarchical-design-and-floorplanning/innovus-implementation-system.html.

[2] OpenAccess API http://www.si2.org/.

[3] IBM CPLEX Optimizer https://www.ibm.com/support/knowledgecenter/SSSA5P_12.7.1/ilog.odms.cplex.help/CPLEX/homepages/CPLEX.html.

[4] Vlsi design flow an overview, May 2017. https://anysilicon.com/vlsi-design-flow-overview/.

[5] R. J. Antinone and G. W. Brown. The modeling of resistive interconnects for integrated circuits. *IEEE Journal of Solid-State Circuits*, 18(2):200–203, April 1983.

[6] I. Backers, B. Sorgeloos, B. Van Camp, O. Marichal, and B. Keppens. Low capacitive dual bipolar esd protection. In *2017 39th Electrical Overstress/Electrostatic Discharge Symposium (EOS/ESD)*, pages 1–7, Sep. 2017.

[7] H. N. Brady. An approach to topological pin assignment. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 3(3):250–255, July 1984.

[8] J. Carballo, W. J. Chan, P. A. Gargini, A. B. Kahng, and S. Nath. Itrs 2.0: Toward a re-framing of the semiconductor technology roadmap. In *2014 IEEE 32nd International Conference on Computer Design (ICCD)*, pages 139–146, Oct 2014.

[9] R. Chaware, K. Nagarajan, and S. Ramalingam. Assembly and reliability challenges in 3d integration of 28nm fpga die on a large high density 65nm passive interposer. In *2012 IEEE 62nd Electronic Components and Technology Conference*, pages 279–283, May 2012.

[10] J. Cong. Pin assignment with global routing. In *1989 IEEE International Conference on Computer-Aided Design. Digest of Technical Papers*, pages 302–305, Nov 1989.

[11] J. Cong. Pin assignment with global routing for general cell designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(11):1401–1412, Nov 1991.

[12] J. Cong, D. Z. Pan, and P. V. Srinivas. Improved crosstalk modeling for noise constrained interconnect optimization. In *Proceedings of the ASP-DAC 2001. Asia and South Pacific Design Automation Conference 2001 (Cat. No.01EX455)*, pages 373–378, Feb 2001.

[13] H. Feng, K. Gong, and A. Z. Wang. A comparison study of esd protection for rfics: performance vs. parasitics. In *2000 IEEE Radio Frequency Integrated Circuits (RFIC) Symposium Digest of Papers (Cat. No.00CH37096)*, pages 235–238, June 2000.

[14] Global Semiconductor Alliance. Electrostatic discharge (esd) in 3d-ic packages. Jan. 2015. https://www.3dincites.com/wp-content/uploads/GSA-ESDA-3D-IC_ESD_Whitepaper_1.pdf.

[15] C. Hao, N. Ding, and T. Yoshimura. An efficient algorithm for 3d-ic tsv assignment. In *2016 14th IEEE International New Circuits and Systems Conference (NEWCAS)*, pages 1–4, June 2016.

[16] X. He and S. Dong. Pin assignment for wire length minimization after floorplanning phase. In *2009 IEEE 8th International Conference on ASIC*, pages 1294–1297, Oct 2009.

[17] G. Hellings, M. Scholz, M. Detalle, D. Velenis, M. de Potter de ten Broeck, C. R. Neve, Y. Li, S. Van Huylenbroek, S. . Chen, E. . Marinissen, A. L. Manna, G. Van der Plas, D. Linten, E. Beyne, and A. Thean. Active-lite interposer for 2.5 amp;amp; 3d integration. In *2015 Symposium on VLSI Circuits (VLSI Circuits)*, pages T222–T223, June 2015.

[18] A. Jain, A. Saha, and J. Rao. Soc design methodology: a practical approach. In *18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design*, pages 10–11, Jan 2005.

[19] M.D Ker, J.J Peng, and H.C Jiang. Esd test methods on integrated circuits: an overview. In *ICECS 2001. 8th IEEE International Conference on Electronics, Circuits and Systems (Cat. No.01EX483)*, volume 2, pages 1011–1014 vol.2, Sep. 2001.

[20] T. Koide, S. Wakabayashi, and N. Yoshida. An integrated approach to pin assignment and global routing for vlsi building-block layout. In *1993 European Conference on Design Automation with the European Event in ASIC Design*, pages 24–28, Feb 1993.

[21] Norman L. Koren. Pin assignment in automated printed circuit board design. In *Proceedings of the 9th Design Automation Workshop*, DAC '72, pages 72–79, New York, NY, USA, 1972. ACM.

[22] M. Lapudes. What will 7nm and 5nm look like? Jan 2015.

[23] C. Lee, C. Hung, C. Cheung, P. Yang, C. Kao, D. Chen, M. Shih, C. C. Chien, Y. Hsiao, L. Chen, M. Su, M. Alfano, J. Siegel, J. Din, and B. Black. An overview of the development of a gpu with integrated hbm on silicon interposer. In *2016 IEEE 66th Electronic Components and Technology Conference (ECTC)*, pages 1439–1444, May 2016.

[24] Z. Li, M. Zhang, and Y. Long. Pin assignment optimization for large-scale high-pin-count bga packages using simulated annealing. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 6(10):1465–1474, Oct 2016.

[25] Q. Liu and H. Li. A hierarchical ip protection approach for hard ip cores. In *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1566–1569, May 2015.

[26] L. Mory-Rauch. Pin assignment on a printed circuit board. In *15th Design Automation Conference*, pages 70–73, June 1978.

[27] K N. Tu. Reliability challenges in 3d ic packaging technology. *Microelectronics Reliability*, 51:517523, 03 2011.

[28] R. K. Pal, S. P. Pal, and A. Pal. An algorithm for finding a non-trivial lower bound for channel routing. In *Proceedings Tenth International Conference on VLSI Design*, pages 531–532, Jan 1997.

[29] C. Pan, Y. Huang, and C. Wang. Characterization and simulation for 2.5-d interposer. In *2015 IEEE Electrical Design of Advanced Packaging and Systems Symposium (EDAPS)*, pages 74–77, Dec 2015.

[30] Pravriti. Vlsi design flow, November 2017. https://www.vlsifacts.com/vlsi-design-flow/.

[31] Brenda S. Baker, Sandeep Bhatt, and Frank Thomson Leighton. *An approximation algorithm for Manhattan routing*, volume 2, pages 205–229. 01 1984.

[32] K. Saban. Xilinx stacked silicon interconnect technology delivers breakthrough fpga capacity, bandwidth, and power efficiency. Dec. 2012. https://www.xilinx.com/support/documentation/white_papers/wp380_Stacked_Silicon_Interconnect_Technology.pdf.

[33] B. M. D. Sawyer, Y. Suzuki, R. Furuya, C. Nair, T. Huang, V. Smet, K. Panayappan, V. Sundaram, and R. Tummala. Design and demonstration of a 2.5-d glass interposer bga package for high bandwidth and low cost. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 7(4):552–562, April 2017.

[34] D. P. Seemuth, A. Davoodi, and K. Morrow. Automatic die placement and flexible i/o assignment in 2.5d ic design. In *Sixteenth International Symposium on Quality Electronic Design*, pages 524–527, March 2015.

[35] D. Stow, Y. Xie, T. Siddiqua, and G. H. Loh. Cost-effective design of scalable high-performance systems using active and passive interposers. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 728–735, Nov 2017.

[36] A.Z Wang, H.G Feng, K Gong, R.Y Zhan, and J Stine. On-chip esd protection design for integrated circuits: an overview for ic designers. *Microelectronics Journal*, 32(9):733 – 747, 2001.

[37] F. Yazdani. A novel low cost, high performance and reliable silicon interposer. In *2015 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–6, Sep. 2015.

[38] C. Yeh and M. Ker. Optimized layout on esd protection diode with low parasitic capacitance. In *2010 10th IEEE International Conference on Solid-State and Integrated Circuit Technology*, pages 1701–1703, Nov 2010.

[39] H. Yin, Y. Zhu, C. Wei, A. Klimashov, and D. Bartle. Skyworks capacitor model for esd applications. In *2011 IEEE International Conference of Electron Devices and Solid-State Circuits*, pages 1–2, Nov 2011.

[40] Y. Zhao, C. Hao, and T. Yoshimura. Tsv assignment of thermal and wirelength optimization for 3d-ic routing. In *2018 28th International Symposium on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 155–162, July 2018.

[41] W. Zhong, S. Chen, , and T. Yoshimura. Lagrangian relaxation based pin assignment and through-silicon via planning for 3-d socs. In *2013 IEEE 10th International Conference on ASIC*, pages 1–4, Oct 2013.

[42] D. Zhou. Lower bound on the channel routing problems. In *[Proceedings] 1992 IEEE International Symposium on Circuits and Systems*, volume 1, pages 13–16 vol.1, May 1992.

[43] Y. Zhou, Y. Yan, and W. Yan. A method to speed up vlsi hierarchical physical design in floorplanning. In *2017 IEEE 12th International Conference on ASIC (ASICON)*, pages 347–350, Oct 2017.