Portland State University
PDXScholar

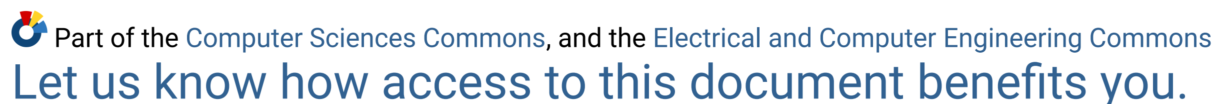
Dissertations and Theses | Dissertations and Theses

2011

Floorplan Design and Yield Enhancement of 3-D Integrated Circuits

Rajeev Kumar Nain
Portland State University

Follow this and additional works at: https://pdxscholar.library.pdx.edu/open_access_etds


Part of the Computer Sciences Commons, and the Electrical and Computer Engineering Commons
Let us know how access to this document benefits you.

Recommended Citation
Nain, Rajeev Kumar, "Floorplan Design and Yield Enhancement of 3-D Integrated Circuits" (2011). *Dissertations and Theses.* Paper 2810.
https://doi.org/10.15760/etd.2804

This Dissertation is brought to you for free and open access. It has been accepted for inclusion in Dissertations and Theses by an authorized administrator of PDXScholar. Please contact us if we can make this document more accessible: pdxscholar@pdx.edu.

ABSTRACT

An abstract of the dissertation of Rajeev Kumar Nain for the Doctor of Philosophy in Electrical and Computer Engineering presented January 06, 2011.

Title: Floorplan Design and Yield Enhancement of 3-D Integrated Circuits

The semiconductor industry has witnessed aggressive scaling of transistors following Moore's law, and has harnessed its benefits in terms of speed, density, and die size in the past several decades. At present transistor count has crossed one billion per chip, and transistor delay has been reduced to picoseconds range. However, the aggressive scaling has slowed down in deep submicron technology because of several challenges in VLSI design and manufacturability. Due to increasing power, performance, cost of fabrication, challenges in lithography, and other financial bottlenecks beyond 28nm, the industry has begun to look for alternative solutions. This has led to the current focus of the industry on stacked three-dimensional (3-D) ICs. Three-dimensional integrated circuits, in which multiple device layers are stacked vertically, are an alternative solution to interconnect related problems. One of the main objectives of 3-D ICs is to replace longer interconnects by shorter wires and vias. Thus it reduces total wirelength, signal delay, buffer count, and power consumption. In addition, 3-D ICs are more suitable for system-on-chip (SOC) design, in which heterogeneous technologies can be fabricated independently in different device layers prior to 3-D stacking. Thus different families of circuits such as logic, processor, memory, analog/RF circuits, sensors, optical I/Os etc. can be integrated in the

3-D stack. However, there are several challenges in the physical design of 3-D ICs such as optimal partitioning, floorplanning, placement and routing, yield, and reliability that need to be addressed before the mainstream acceptance of 3-D ICs by the electronics industry. Therefore several CAD tools and intelligent design methodologies are needed to evaluate and realize a 3-D design. Consequently, this work focuses on two sub-problems of 3-D IC design a) three-dimensional floorplanning, and b) yield enhancement of 3-D ICs.

We have developed a placement-aware 3-D floorplanning algorithm that enables additional wirelength reduction by planning for 3-D placement of logic gates in selected circuit modules during the floorplanning stage. Thus it also bridges the existing gap between 3-D floorplanning and 3-D placement. To reduce the solution space of 3-D floorplanning which is known to be an NP-hard problem, we derive a set of feasibility conditions on the topological representation of a floorplan. In addition, we have designed a fast module packing algorithm that satisfies a set of constraints for placement-aware 3-D floorplanning. Furthermore, we have designed an efficient evolutionary algorithm that is used in the proposed 3-D floorplanning algorithm for multi-objective combinatorial optimization. Our results show that the proposed placement-aware 3-D floorplanning algorithm is very fast, and it reduces the system level total wirelength by 9.8% compared to existing state-of-the-art floorplanning tools that do not plan for 3-D placement of floorplanning modules.

Our module packing algorithm and previously derived feasibility conditions are used to design another 3-D floorplanning algorithm with vertical module alignment (3-D FMA) that can be applicable in bus-driven 3-D floorplan design and heterogeneous 3-D IC

design. Our results show that 3-D FMA can generate good quality floorplans while satisfying a user defined set of constraints such as vertical alignment of modules, layer assignment of modules, and module repulsion constraints. It also scales well with increasing problem size.

Next, we identify the functional yield problem due to failure of vertical interconnects known as Through-Silicon Vias (TSVs) and propose yield improvement techniques based on via redundancy technique. Monte Carlo simulation results show that with proposed redundancy structures, high yield can be achieved for a large number of TSVs and a wide range of defect rates. We then present a stochastic methodology to estimate parametric yield in terms of the number of fast/slow chips in a bin.

Finally, we present a set of redundant via dependent analytical yield models for functional as well as parametric yield. The results obtained by the analytical models match closely with Monte Carlo simulation results. Thus they eliminate the need for computationally expensive Monte Carlo simulations. These analytical models can be used in fast estimation of yield, and can be used in yield-aware physical design such as 3-D floorplanning and P&R. We further derive an analytical model for a sweet spot between the numbers of fast/slow chips obtained using our proposed solutions, and present an analytical model for the estimation of total chip revenue. The total chip revenue model takes the prices of fast and slow chips as input, and for a given TSV defect rate and our redundancy configuration, it estimates the total number of fast/slow chips in a bin for the total chip revenue estimation.

DEDICATION

This dissertation is dedicated to my parents.

ACKNOWLEDGMENTS

I would like to thank my advisor Malgorzata Chrzanowska-Jeske, whose encouragement, supervision and support from the preliminary to concluding level enabled me to pursue the research. I sincerely thank all the members of my dissertation committee for their participation during various stages of my academic progress. My special thanks to Prof. Robert Daasch for providing useful feedback on 3-D IC yield work during my seminar presentation. Furthermore, I wish to thank my colleague Rehman Ashraf for the insightful discussions during the research work related to 3-D IC yield. My special thanks to Darcy Kennedy and Melissa Sutherland for their help during proof reading of my research articles, and this dissertation.

Finally I would like to thank my family for their endless support and love. Without their support, the completion of this work would not have been possible.

Floorplan Design and Yield Enhancement of 3-D Integrated Circuits

by

Rajeev Kumar Nain

A dissertation submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy
in
Electrical and Computer Engineering

Dissertation Committee:
Malgorzata Chrzanowska-Jeske, Chair
Dan W. Hammerstrom
W. Robert Daasch
Paul Van Halen
Fei Xie

Portland State University
©2011

TABLE OF CONTENTS

Acknowledgments ii
List of Tables vi
List of Figures vii
Glossary xi
Chapter 1: Introduction 1
1.1 Major Challenges of 2-D Integrated Circuits 1
1.2 Three-Dimensional Integrated Circuits 3
1.3 3-D IC Technologies 4
1.4 Through-Silicon Via Technologies 9
1.5 Major Challenges in 3-D IC Design 10
1.5.1 CAD Tools for 3-D ICs 10
1.5.2 Heat Extraction and Thermal Management 11
1.5.3 Power and Ground Delivery in 3-D ICs 13
1.5.4 Clock Tree Synthesis for 3-D ICs 13
1.5.5 TSV Management in 3-D ICs 14
1.5.6 TSV-Induced Design for Manufacturability and 3-D IC Yield 14
1.6 Current Status of 3-D Integrated Circuits 17
1.7 Contributions of the Dissertation 18
1.7.1 CAD Tool Design for Native 3-D Floorplanning 18
1.7.2 Yield Improvement of 3-D ICs in the Presence of TSV failure 20
1.8 Structure of the Dissertation 21
Chapter 2: Floorplanning of 2-D and 3-D Integrated Circuits 23
2.1 Shapes of Circuit Modules in a Floorplan 23
2.2 Classification of Floorplanning 24
2.3 Complexity of the Floorplanning Problem 25
2.4 2-D Floorplan Representations 27
2.5 3-D Floorplan Representations 28
2.6 Previous Work on 3-D Floorplanning 31
2.7 A Basic 3-D Floorplanning Tool 33
Chapter 3: Vertical Constraints in Sequence Pair Representation 36
3.1 Introduction to Sequence Pair 38
3.2 Vertical Constraints on Sequence Pairs 41
3.3 Two Layer Feasibility Condition 42
3.4 Graph Representations of Feasibility Conditions 44
3.5 3DCG: A Module Packing Algorithm with Vertical Constraints 49
3.6 LCSLS: A Fast Module Packing Algorithm with Vertical Constraints 53
Chapter 4: Placement-Aware 3-D Floorplanning 57
4.1 Motivation 57

4.2 Placement-Aware Constraints 59
4.3 Problem Formulation 62
4.4 Stochastic Combinatorial Optimization 62
4.5 Cost Estimation of wirelength reduction due to module splitting inside 3-D Modules 63
4.6 Perturbation of Solution Space 67
4.7 Cost Function 73
4.8 Design Flow of the Placement-Aware 3-D Floorplanning Algorithm 75
4.9 Experimental Results 78
4.9.1 Experimental Setup 78
4.9.2 Comparison of 3-D Packing Algorithms with Vertical Constraints 79
4.9.3 Comparison of Area and Wirelength Minimization without Vertical Constraints 81
4.9.4 Impact of Placement-Aware 3-D Floorplanning with Vertical Constraints on System Level Wirelength 85
4.9.5 Effect of Feasibility Conditions on the Solution Quality of 3-D FVC 91

Chapter 5: 3-D Floorplanning with Module Alignment 93
5.1 Introduction and Motivation 93
5.2 Problem Formulation 95
5.3 Combinatorial Optimization and the Cost Function 97
5.4 Perturbation of the Solution Space 100
5.5 Experimental Results 104
5.5.1 Effect of Module Alignment on MCNC Benchmarks 104
5.5.2 Effect of Increasing the Number of Module Alignment Constraints on 3-D Floorplanning using GSRC Benchmarks 106
5.5.3 Composite Effect of Increasing the Number of Various Constraints on 3-D Floorplanning using GSRC Benchmarks 106
5.5.4 Runtime Comparison of 3-D FMA with LTCG based 3-D Floorplanning Algorithm 108
5.5.5 An Example of a 4-Layer 3-D Floorplan with Various Constraints 109

Chapter 6: TSV-Induced 3-D IC Yield 112
6.1 TSV Fabrication Technologies 112
6.2 Alternative Via Technologies: Wireless Vias 114
6.3 Carbon Nanotube based Inductors for Wireless Vias 117
6.3 Yield as a Function of TSV Failure 118

Chapter 7: Via Redundancy for 3-D IC Yield Improvement 119
7.1 Redundancy Lattices 122
7.2 Redundancy Evaluation Factors 123
7.3 Redundancy Configuration in a Device Layer 125
7.3.1 Wireless Via Redundancy Configuration 125
7.3.2 Physical Via Redundancy Configuration 126
7.4 Yield Estimation by Monte Carlo Simulation 129
7.5 Modeling Area, Delay and Power of Redundant Vias 134
7.5.1 Area Tradeoff 134

7.5.2 Delay Tradeoff....135
7.5.3 Power Tradeoff....137
7.6 Effect of Redundancy on Parametric Yield....138
7.6.1 Estimation of the Total Number of Global Wires....139
7.6.2 Estimation of the Total Number of Fast Chips....140

Chapter 8: Redundant Via Dependent Analytical Yield Models....143
8.1 Nomenclature....144
8.2 Analytical Model for Quad Wireless Plus (QWP) Configuration....145
8.3 Analytical Model for Octal Wireless Plus (OWP) Configuration....149
8.4 Analytical Model for Octal TSV Complete (OTC) Configuration....150
8.5 Analytical Model for Fast/Slow Chips....151
8.6 Analytical Model for Chip Revenue....154
8.7 Extension of Analytical Yield Models of Two-Layer 3-D Chips to Multi-Layer 3-D Chips....155
8.8 Application of Yield Improvement Strategies during Floorplanning....157

Chapter 9: Conclusions and Future Work....161

Chapter 10: Summary of Major Contributions....167

References....170

Appendix A: Mathematical Proofs of the Feasibility Condition Theorems....184

Appendix B: Stochastic Rectangular 3-D Wirelength Distribution Models....188
B.1 Introduction....188
B.2 General Background....189
B.3 Rectangular 2-D Wirelength Distribution Model....192
B.4 Extension to Rectangular 3-D Wirelength Distribution Models....200
B.5 Experimental Results....201
B.5.1 Effect of Aspect Ratio on Wirelength Distribution and Total Wirelength 202
B.5.2 Comparison Among Rectangular 3-D Wirelength Distribution Models....203

Appendix C: Rent's Parameter Extraction....205
C.1 Extraction of Rent's Parameters from The Netlist of A Circuit Block....205
C.2 Rent Parameter For Modules of Floorplan Benchmarks....205

Appendix D: Sensitivity Analysis of the Cost Function of Floorplanning Algorithms....213
D.1 Sensitivy Analysis of the Cost Function of 3-D FVC Algorithm Without Module Splitting....213
D.2 Sensitivy Analysis of the Cost Function of 3-D FVC Algorithm With Module Splitting....216
D.3 Sensitivy Analysis of the Cost Function of 3-D FMA....219

LIST OF TABLES

TABLE 4.1: COMPARISON OF TOTAL MEASURED WIRELENGTH (226827.0 GATE PITCHES) OF 18 IFU CIRCUITS OF IBM POWER4 WITH THEIR ESTIMATED WIRELENGTH USING DIFFERENT WIRELENGTH DISTRIBUTION MODELS. 66

TABLE 4.2: RUNTIME COMPARISON OF 3-D PACKING WITH VERTICAL CONSTRAINTS 80

TABLE 4.3: COMPARISON OF FOOTPRINT AREA, WIRELENGTH AND VIA COUNT OPTIMIZATION WITHOUT VERTICAL CONSTRAINTS 83

TABLE 4.4: RUNTIME COMPARISON WITHOUT VERTICAL ALIGNMENT 84

TABLE 4.5: EFFECT OF 3-D FVC WITH MODULE SPLITTING ON THE SYSTEM LEVEL TOTAL WIRELENGTH ON A 4-LAYER 3-D FLOORPLAN. 88

TABLE 4.6: EFFECT OF FEASIBILITY CONDITIONS ON THE SOLUTION QUALITY OF 3-D FVC 90

TABLE 5.1: EFFECT OF DIFFERENT PLACEMENT CONSTRAINTS ON A 4-LAYER 3-D FLOORPLAN USING MCNC BENCHMARKS 105

TABLE 5.2: EFFECT OF INCREASING THE NUMBER OF MODULE ALIGNMENT CONSTRAINTS ON A 4-LAYER 3-D FLOORPLAN USING GSRC BENCHMARKS 107

TABLE 5.3: COMPOSITE EFFECT OF INCREASING THE NUMBER OF DIFFERENT CONSTRAINTS ON A 4-LAYER 3-D FLOORPLAN USING GSRC BENCHMARKS 107

TABLE 5.4: RUNTIME COMPARISON OF 3-D FMA WITH LTCG BASED 3-D FLOORPLANNER 110

TABLE 7.1: NUMBER OF TSVS REQUIRED TO BE CONNECTED IN PARALLEL WITH EACH PRIMARY TSV IN 3-D CHIPS TO OBTAIN 90% FUNCTIONAL YIELD 120

TABLE 7.2: MONTE CARLO YIELD RESULTS FOR REDUNDANT TSV CONFIGURATIONS. 130

TABLE 7.3: MONTE CARLO YIELD RESULTS FOR WIRELESS VIA REDUNDANCY 130

TABLE 7.4: COMPARISON OF 3-D IC YIELD OBTAINED BY MONTE-CARLO SIMULATION WITH INCREASING VIA COUNT AND USING THE MOST PROMISING VIA REDUNDANCY CONFIGURATIONS 133

TABLE 7.5: AREA PENALTY OF REDUNDANT LATTICES IN TERMS OF THE EQUIVALENT AREA OF A TWO-INPUT NAND GATE 135

TABLE 7.6: DELAY PENALTY AND IT'S SCALING WITH TECHNOLOGY NODE 135

TABLE 7.7: POWER PENALTY AND ITS SCALING WITH TECHNOLOGY NODE 137

LIST OF FIGURES

Figure 1.1: Trend between interconnect delay and gate delay for different technology nodes [124]. 2

Figure 1.2: Comparison between (a) 2-D ICs, and (b) 3-D ICs. In 2-D ICs, all blocks are integrated on a single silicon substrate. 3-D IC technology offers the flexibility of integration using heterogeneous types of substrates in a 3-D stack. 3

Figure 1.3: (a) Schematic of an epitaxially grown second device layer [98] (b) Schematic of the Ge seeded solid phase crystallization (SPC) process [1] 5

Figure 1.4: Cross-sectional view of different types of wafer bonding (a) face-to-face (b) face-to-back. [4] 7

Figure 1.5: Different types of stacking methods used in 3-D IC fabrication (a) Wafer-to-Wafer, (b) Die-to-Die, and (c) Die-to-Wafer. 8

Figure 1.6: TSVs in a stacked 3-D IC. [28] 9

Figure 1.7: Technology range and applications of vertical interconnects. [10] 10

Figure 1.8: Cross sectional view of a 3-D IC using integrated microchannel cooling technology. [22] 12

Figure 1.9: (a) High thermo-mechanical stress shown by the white rectangle at the edge of a TSV [38], (b) An SEM image of a visible crack due to stress in a TSV [23], and (c) High stress sites in a multi-layer 3-D IC [27]. 16

Figure 2.1: (a) the structure of a T-tree, (b) A compacted placement of modules and the corresponding T-tree. [62] 29

Figure 2.2: (a) true 3-D floorplanning [125] (b) quasi 3-D floorplanning [13] 30

Figure 2.3: Flow-chart of a basic 3-D floorplanning tool inherited at the beginning of the research. The shaded boxes indicate the portions of the algorithm that have been modified as a part of this research work. 35

Figure 3.1 : Example of vertical constraints in 3-D SoC (a) module alignment, bus planning, and each layer containing analog/RF blocks are vertically aligned together, (b) analog/RF blocks have been assigned in on the top layer and they have been separated from noisy digital block 37

Figure 3.2: (a) oblique grid for $\Gamma^{+} = \{a\ b\ c\ d\}$ and $\Gamma^{-} = \{c\ a\ d\ b\}$, (b) resultant placement of blocks, and (c) different packing for the same sequence pair due to change in sizes of modules 39

Figure 3.3: Weighted graphs G_h (left) and G_v (right) of the sequence pair shown in Figure 3.2 39

Figure 3.4: (a) An example of a two-layer grouped sequence pair, and (b) its corresponding floorplan in two device layer. 41

Figure 3.5: (a) Module pairs $\{A_1, A_2\}$ and $\{B_1, B_2\}$ under vertical constraints in two device layers are shown. (b) If A_1 is moved rightward as shown by the arrow to align with A_2, B_1 moves further away from B_2. Both pairs cannot be aligned simultaneously, and thus are infeasible. 43

Figure 3.6: Construction of a feasibility condition graph: (a) Given sequence pairs $\{\Gamma_1^+ ; \Gamma_1^-\}$ of layer 1 and $\{\Gamma_2^+ ; \Gamma_2^-\}$ of layer 2. Module pairs $\{A_1, A_2\}$ and $\{B_1, B_2\}$ are vertically constrained. (b) Constrained sequence pairs (c) The resultant feasibility condition graph. 45

Figure 3.7: Graph representation showing six out of eight cases which are feasible conditions for two module pairs in two device layers. 46

Figure 3.8: Graph representation of infeasible sequences pair constraints for two module pairs in two device layers 46

Figure 3.9: 3-D-X constraint graph creation. (a) 2-D constraint graph along +X axis in Layer 1 (b) 2-D constraint graph along +X axis in Layer 2 (c) Merging the two 2-D constraint graphs using a global source and a global sink node. To vertically align node2 and node7, two edges are inserted between node2 and node7 in the cyclic fashion (i.e. 2→7 and 7→2). All newly introduced edges are dotted and have zero weights. 52

Figure 3.10: Creation of a constrained adjacency list *Adj_X* for 2-device layers. (a) $A = \{A_1, A_2\}$, $B = \{B_1, B_2\}$, ..., $E = \{E_1, E_2\}$ are vertically constrained in two device layers. (b) Adj_X obtained by merging the graphs of two layers and denoting the merged nodes by their group name. 54

Figure 4.1: (a) A 3-D floorplan of 2-D modules (b) a new 3-D floorplan with sub-module pairs $\{A_1, A_2\}$ and $\{C_1, C_2\}$ transformed as 3-D modules after module splitting and imposing vertical constraints. 59

Figure 4.2: Sub-module pair placed in (a) consecutive (b) alternate device layers. Sub-modules in the consecutive device layers minimize the TSV height. 61

Figure 4.3: Sub-module pair with (a) the same planar location (b) different planar locations. Vertical constraint with the same planar location of sub-modules produces smaller intra-module wirelength. 61

Figure 4.4: Predicted vs. placed and routed wirelengths of ISPD'98 benchmark circuits. Wirelength of 3-D placement and routing is normalized w.r.t. 2-D design to show the percentage reduction due to 3-D design [75]. 66

Figure 4.5: An example of Module split move. (a) Sub-modules $\{A_1, B_1, C_1, D_1\}$, $\{A_2, B_2, C_2, D_2\}$ are vertically constrained in layer 1 and layer 2 under the feasibility configuration of a clique of size 4 and their sequence pairs are shown. Module "w" initially resides in layer 2 which is about to be split. (b) Resultant sequence pairs of layer 1 and layer 2 after splitting. 71

Figure 4.6: Flowchart of the proposed placement-aware 3-D floorplanning using vertical constraints (3-D FVC). The grey shaded portions have been modified from the initial algorithm. The left most rectangular boxes in orange shade are the new methods. 77

Figure 4.7: Runtime of 3-D FVC (with increasing problem size) using various packing algorithms with vertical constraints. LCSLS is faster than 3DCG. 80

Figure 4.8: Runtime comparison of various 3-D floorplan algorithms with increasing problem size. Thermal optimization and vertical alignment were disabled. All runtimes have been scaled linearly to 3 GHz CPU speed. 84

Figure 4.9: Comparison of (a) inter-module wire, (b) total wirelength (c) footprint area, and (d) inter-module via count, obtained using different floorplanning tools. Total wirelength is the sum of inter and intra module wirelength. Vertical constraints are applied only when module splitting (MS) is activated. 3-D FVC (with MS) reduces total wirelength by 9.8%. The bar charts represent the aggregate data of all the benchmarks. Please note that each chart has a different y-scale. 87

Figure 4.10: A 4-layer 3-D floorplan of ami49. Please notice that parts of module 0 (0A, 0B), 1, 2, 3, 4, 5, 7, 29, 32, 43, 44 and 47 have been placed in consecutive device layers. The planar locations of sub-modules are the same. 89

Figure 5.1: Example of different placement constraints in 3D floorplanning. 94

Figure 5.2: Runtime Comparison of 3-D FMA with LTCG 110

Figure 5.3: A 4-layer 3-D floorplan of ami49 obtained using 3D-FMA. Module groups under MA constraints are {0, 3}, {1, 2, 5, 32}, and {10, 17, 20, 48}. MR constraint groups are {4, 33} and {11, 28}. Layer assignment constraints in Layer 1 = {3, 33, 40}, Layer 2 = {0, 7, 32}, Layer 3 = {12}, and Layer 4 = {13}. 111

Figure 6.1: TSVs in 3-D ICs using (a) via-first, and (b) via-last methods. [107] 112

Figure 6.2: Concept of ACCI (a) Capacitive ACCI (b) Inductive ACCI. [82] 115

Figure 6.3: Schematic of (a) transmitter, and (b) receiver circuits of a wireless via using inductive coupling [82]. 116

Figure 7.1: Proposed lattice structures for redundant via insertion in a device layer of a 3-D IC (a) Quad Lattice, (b) Octal Lattice, and (c) Dual Lattice. The redundant via can either be a wireless via or a physical TSV. 123

Figure 7.2: Quad Wireless Plus (QWP) configuration. If all the four TSVs within the shaded quad lattice fail, then it can be repaired by the neighboring wireless vias. 127

Figure 7.3: Interaction of an Octal Lattice (shaded region at the center) with its neighborhood lattices in an Octal Wireless* Configuration 128

Figure 7.4: Interaction of an Octal Lattice (grey shaded region at the center) with its neighboring lattices in an Octal Wireless Plus configuration. 128

Figure 7.5: Interaction of a Dual (grey shaded) lattice with the neighboring lattices (encircled by dotted lines) in a Dual TSV redundancy configuration. 129

Figure 7.6: Performance Reduction due to via re-routing through MUX logic. The chip's target frequency is 2.5 GHz (ideal case). 136

Figure 7.7: A critical path spanning across two device layer using a TSV. We assume that a critical path crosses only once through a TSV..138

Figure 7.8: Comparison between the number of fast chips obtained (a) for 4% defect rate using Quad Wireless Plus (QWP), Octal Wireless Plus (OWP), and Octal TSV Complete (OTC) configurations (b) for 1% and 4% defect rates by Quad Wireless Plus (QWP) configuration. 142

Figure 8.1: Comparison of Analytical model results with Monte Carlo simulation results for yield obtained using (a) Quad Wireless Plus configuration, and (b) Octal Wireless Plus Configuration for different numbers of TSVs in 3-D ICs. 148

Figure 8.2: Analytical and Monte Carlo simulation results for Fast/Slow chips for Quad Wireless Plus configuration for (a) 1% defect rate (b) 4% defect rate.....................153

Figure 8.3: Decomposition of a 3-layer chip into a pair of identical two-layer 3-D ICs for yield calculation using superposition. 156

Figure 8.4: The flow chart of a Yield-Aware 3-D Floorplanning...159

GLOSSARY

3DCG	3-D Constraint Graph
3-D STAF	3-D Scalable and Temperature Aware Floorplanning
ACCI	AC Coupled Interconnects
BCB	Benzocyclobutene
BSG	Bounded Sliceline Grid
BEOL	Back-end-of-line
CAD	Computer Aided Design
CBA	Combined Bucket Array
CBL	Corner Block List
CMOS	Complementary Metal Oxide Semiconductor Field Effect Transistor
CMP	Chemical Mechanical Polishing
CNT	Carbon Nano Tube
CPU	Central Processing Unit
CTE	Coefficient of thermal expansion
DAG	Directed Acyclic Graph
D2D	Die-to-Die
D2W	Die-to-Wafer
DTL	Dual TSV Lattice
EA	Evolutionary Algorithm
ES	Evolution Strategy
FEM	Finite Element Method
FPGA	Field Programmable Gate Array
GSRC	Giga Scale Research Center
GSP	Grouped Sequence Pair
HPWL	Half Perimeter Wire Length
IEEE	Institute of Electrical and Electronics Engineers
IBM	International Business Machine
IC	Integrated Circuit

IFU	Instruction Fetch Unit
ITRS	International Technology Roadmap for Semiconductors
LNA	Low Noise Amplifier
LCS	Longest Common Subsequence
LCSLS	Longest Common Subsequence and Lateral Shifting
MC	Monte Carlo
MCNC	Microelectronics Center of North Carolina
MUX	Multiplexer
N	Number of logic gates
OWL*	Octal Wireless* Lattice
OWP	Octal Wireless Plus Configuration
OW*	Octal Wireless* Configuration
OTL	Octal TSV Lattice
P&R	Placement & Routing
QWL	Quad Wireless Lattice
QWP	Quad Wireless Plus Configuration
QTL	Quad TSV Lattice
Rx	Receiver
SOC	System on Chip
SA	Simulated Annealing
SEM	Scanning Electron Microscope
SP	Sequence Pair
SPC	Solid Phase Crystallization
SRAM	Static Random Access Memory
TCG	Transitive Closure Graph
TSV	Through Silicon Via
Tx	Transmitter
VLSI	Very Large Scale Integration
W2W	Wafer-to-Wafer

CHAPTER 1: INTRODUCTION

1.1 MAJOR CHALLENGES OF 2-D INTEGRATED CIRCUITS

As device size continues to decrease, designers integrate more and more functional blocks on a single die of a 2-D IC. As a result, circuit density, complexity of wiring, and the number of global wires continue to increase. Furthermore, aggressive scaling with new technology results in a smaller wire cross-section, shorter wire pitch and longer global interconnects. Due to these scaling trends, interconnect delays continue to increase whereas the gate delay continues to decrease with shrinking technology node. Therefore interconnect delay dominates over gate delay in deep submicron technology [1], and remains a critical bottleneck of high performance VLSI chip design. According to ITRS, interconnect delay determines the system performance. For example, for a 1mm minimum pitch Cu global wire without repeater as 45nm technology node has a 542 ps delay, whereas a 10 level (logic depth) FO_4 delay is around 150ps at 45 nm [124]. Figure 1.1 shows the trend of global wire delay and gate delay with respect to different technology nodes. From Figure 1.1, increasing gap between the gate delay and the interconnect delay can be observed with advancement in the technology node.

In the era of Information Technology, the need for system-on-chip (SOC) has been constantly increasing. The SOC integrates different types of circuits such as logic, memory, analog/RF blocks, etc within a single chip. This heterogeneous integration on a single die becomes difficult because different families of circuits might require different types of substrates. Furthermore, the unwanted interaction of electrical signals among these circuits

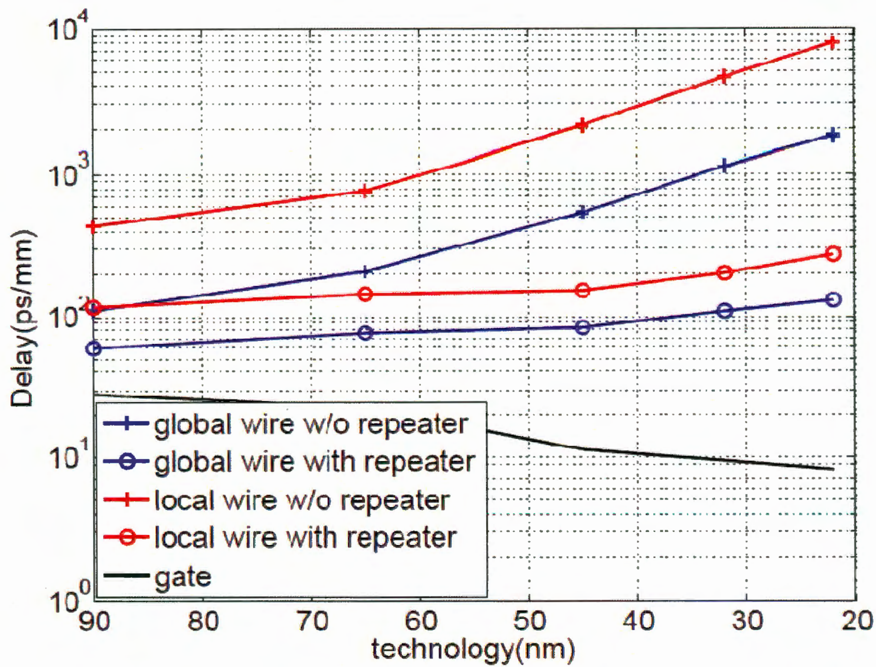
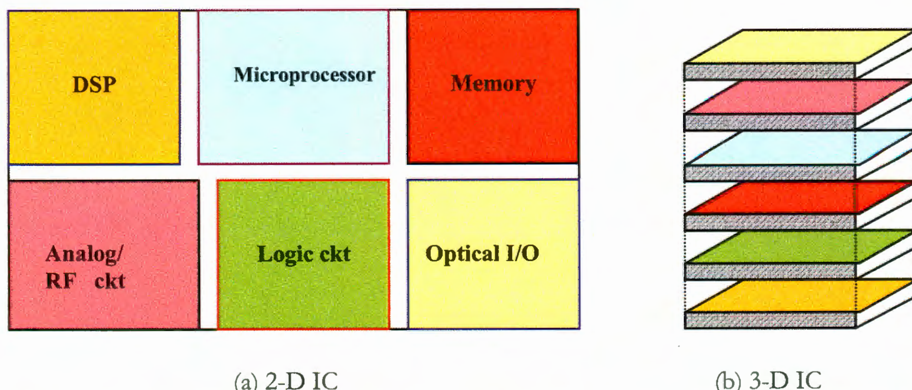


Figure 1.1: Trend between interconnect delay and gate delay for different technology nodes [124].

(such as substrate noise in a mixed signal design) might require additional design steps and increase design complexity.

The electronics industry miniaturizes ICs by advanced lithography technology that is expected to continue through a 22 nm technology node and beyond. However, due to the increasing power, performance, and financial bottlenecks beyond 28 nm, industry has begun to look for alternative solutions. Therefore it has led to the current focus of the industry on stacked three-dimensional (3-D) integrated circuits. Research works have shown that 3-D ICs can reduce global wirelength between 28 – 56% [1],[2],[3],[4] which will in turn improve performance of 3-D chips, reduce buffer count, and decrease power consumption in wires. Researchers in [121] have presented a case study of 3-D design for ternary content-addressable memory (TCAM), and an 8192-point FFT processor fabricated



(a) 2-D IC (b) 3-D IC

Figure 1.2: Comparison between (a) 2-D ICs, and (b) 3-D ICs. In 2-D ICs, all blocks are integrated on a single silicon substrate. 3-D IC technology offers the flexibility of integration using heterogeneous types of substrates in a 3-D stack.

at 180nm technology. The 3-D TCAM design shows 23% power reduction whereas the 3-D FFT shows 22% reduction in cycle time, 28% improvement in clock frequency, and 18% reduction in energy per Fourier transform compared to the same designs fabricated as a single layer 2-D chip [121]. Thus 3-D ICs offer advantages over 2-D chips even at the same technology node.

1.2 THREE-DIMENSIONAL INTEGRATED CIRCUITS

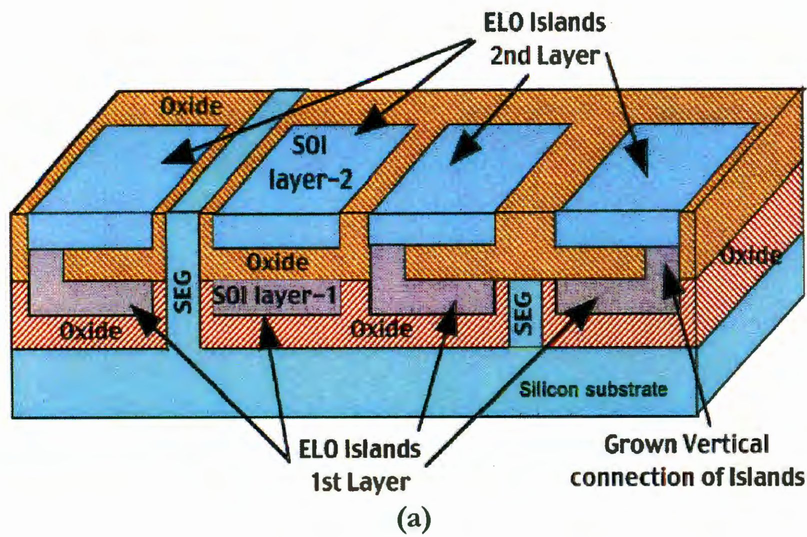
Three-dimensional (3-D) integrated circuits, in which multiple device layers are stacked vertically, are an alternative solution to interconnect related problems. One of the main objectives of 3-D ICs is to replace longer interconnects by shorter wires and vias. Thus it reduces total wirelength [1]-[4], signal delay, buffer count, and power consumption. In addition, 3-D ICs are more suitable for system-on-chip (SOC) design, in which heterogeneous technologies can be fabricated independently in different device layers prior to 3-D stacking. Thus different families of circuits such as logic, processor, memory, analog/RF circuits, sensors, optical I/Os etc. can be integrated in the 3-D stack.

Furthermore, circuits fabricated using different types of substrates and different technology nodes can also be integrated in 3-D ICs as shown in Figure 1.2. In addition, memory intense digital systems can attain lower latency and higher throughput/bandwidth by stacking memory on top of digital logic circuits. It has been shown by the researchers that 3-D technology can close the processor-memory performance gap [96],[97].

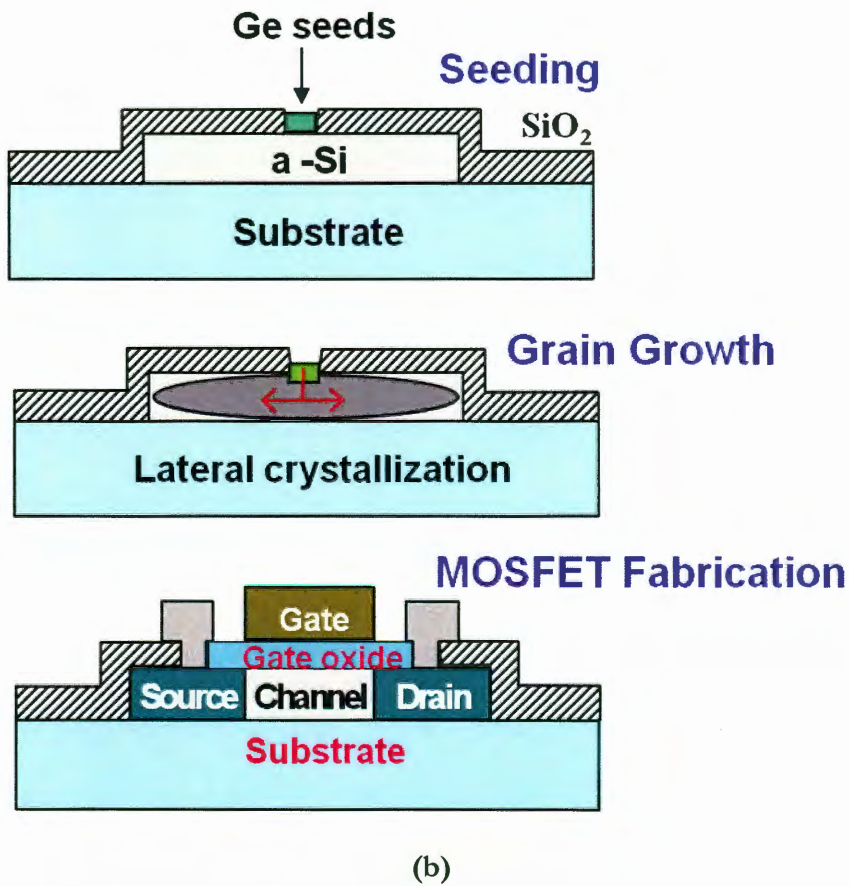
1.3 3-D IC TECHNOLOGIES

The main technologies available for 3-D ICs are Silicon Epitaxial Growth, Solid Phase Crystallization and Processed Wafer Bonding [1]. In the *Silicon Epitaxial Growth* technique, an additional Si layer is formed by etching holes in the passivated wafer and epitaxially growing a single crystal Si that is seeded from open windows in the inter-layer dielectric (ILD). The silicon crystal first grows vertically and then laterally to cover the ILD as shown in Figure 1.3(a) [98]. Since the newly grown layer of Si is a single crystal with few defects, the quality of devices fabricated on the epitaxial layer can be theoretically as good as those fabricated on the wafer surface. However, the high temperature (~1000°C) involved in this process causes significant degradation in the quality of lower layer devices. Also, this technique cannot be used over metallization layers.

As an alternative to high temperature Silicon Epitaxial Growth, the *Solid Phase Crystallization* (SPC) technique offers low temperature deposition and crystallization of amorphous silicon (a-Si), on top of the lower active layer devices. The amorphous film is randomly crystallized to form a polysilicon film. The device performance is enhanced with local crystallization using low-temperature processes (<600°C) such as patterned seeding of Germanium as



(a)



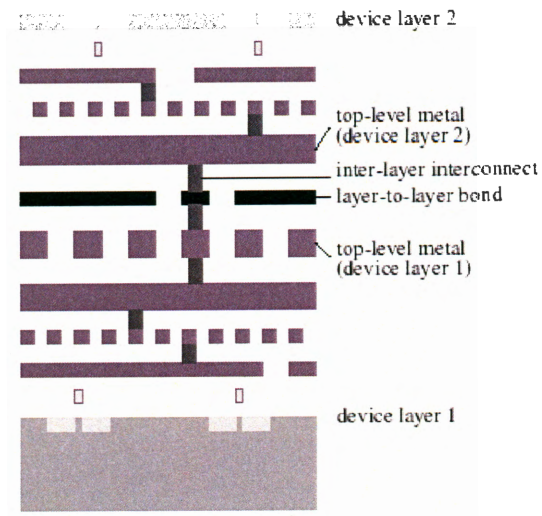
(b)

Figure 1.3: (a) Schematic of an epitaxially grown second device layer [98] (b) Schematic of the Ge seeded solid phase crystallization (SPC) process [1].

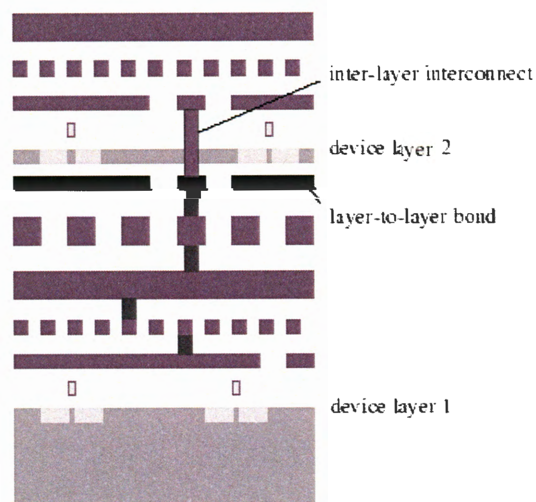
shown in Figure 1.3(b). In this method, Ge seeds implanted in narrow patters made on a-Si are used to induce lateral crystallization. This results in the formation of small islands, which are nearly single-crystal Si [1]. This technique offers multiple active layer creation compatible with the current CMOS processing environment. However, the electrical characteristics of the fabricated devices are still not as good as single-crystal devices [99] due to the fabrication processes (in upper device layer) involving temperature around 500° C [99] .

Another attractive technique, *Processed Wafer Bonding,* allows two fully processed wafers with fabricated devices and some interconnects to be bonded together using copper wafer bonding [5],[6]. This technique, also known as 3-D stacking, is very suitable for further bonding of more device layers in the vertical direction and provides similar electrical properties of devices on all active layers.

The processed wafer bonding technique allows face-to-face or face-to-back stacking as shown in Figure 1.4. In a face-to-face 3-D stacking, the silicon devices and interconnects in two device layers face each other. Thus it only allows stacking of two-device layers. On the other hand, in face-to-back stacking, the silicon devices of one layer face the thinned substrate of another layer. Thus it is suitable for more than two layer 3-D integration. Please note that face-to-face bonding does not need inter-layer vertical interconnects because the electrical connections between two layers are established using metal layers. However, face-to-back bonding requires such vertical interconnects known as through-silicon vias (TSVs) that can pass through the thinned substrate of another layer.

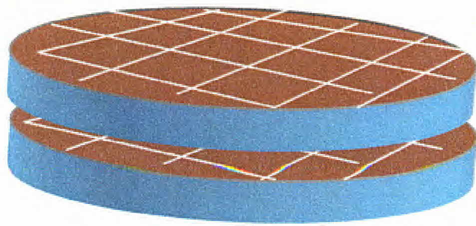


(a)

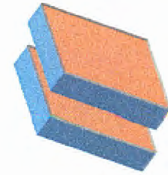


(b)

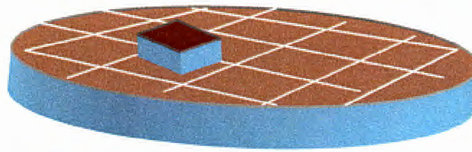
Figure 1.4: Cross-sectional view of different types of wafer bonding (a) face-to-face (b) face-to-back. [4]



(a) Wafer-to-Wafer (W2W)



(b) Die-to-Die (D2D)



(c) Die-to-Wafer (D2W)

Figure 1.5: Different types of stacking methods used in 3-D IC fabrication (a) Wafer-to-Wafer, (b) Die-to-Die, and (c) Die-to-Wafer.

Currently there are three different stacking methods used in 3-D IC fabrication: a) wafer-to-wafer (WTW), b) die-to-wafer (DTW), and c) die-to-die (DTD) as shown in Figure 1.5. In wafer-to-wafer integration, entire wafers are bonded together [1],[7]. It offers the highest throughput and the thinnest via diameter. However, WTW suffers serious yield loss due to lack of control of stacking a good die with a bad die. Die-to-wafer and die-to-die methods allow the die to be diced and tested prior to 3-D stacking. Thus these methods offer known-good-die stacking which can improve the 3-D yield. The yield improvement, however, comes at lower throughput, lower TSV density, additional testing and bonding cost [8].

1.4 THROUGH-SILICON VIA TECHNOLOGIES

Through-Silicon Vias (TSVs) are vertical interconnects that entirely pass through a silicon die in order to provide electrical connectivity across different device layers in a 3-D stack as shown in Figure 1.6. There are two different types of TSV fabrication technologies that depend on the order of their fabrication during 3-D integration. They are known as *via-first* and *via-last* technologies. In the via-first technology, TSVs are fabricated before CMOS or BEOL (back-end-of-line) metallization. The diameters of TSVs fabricated using this technology are typically in the range of 1 - 10μm with aspect ratio (i.e. height : diameter) ranging from 3:1 to 10:1. In contrast, the via-last TSVs are created after BEOL or bonding. Thus, the processing for via-last can be done at the foundry or at the packaging house. The diameters of TSVs obtained by the via-last technology are larger, typically in the range of 10 - 50μm, with aspect ratios from 3:1 to 15:1 [9]. A detailed description of TSV fabrication

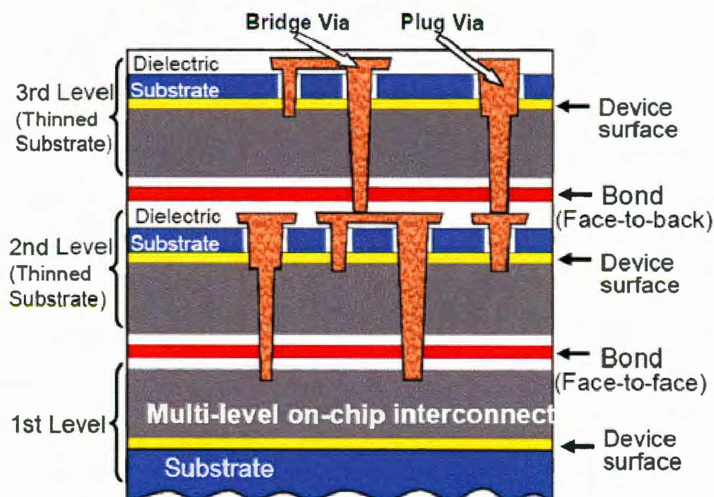


Figure 1.6: TSVs in a stacked 3-D IC. [28]

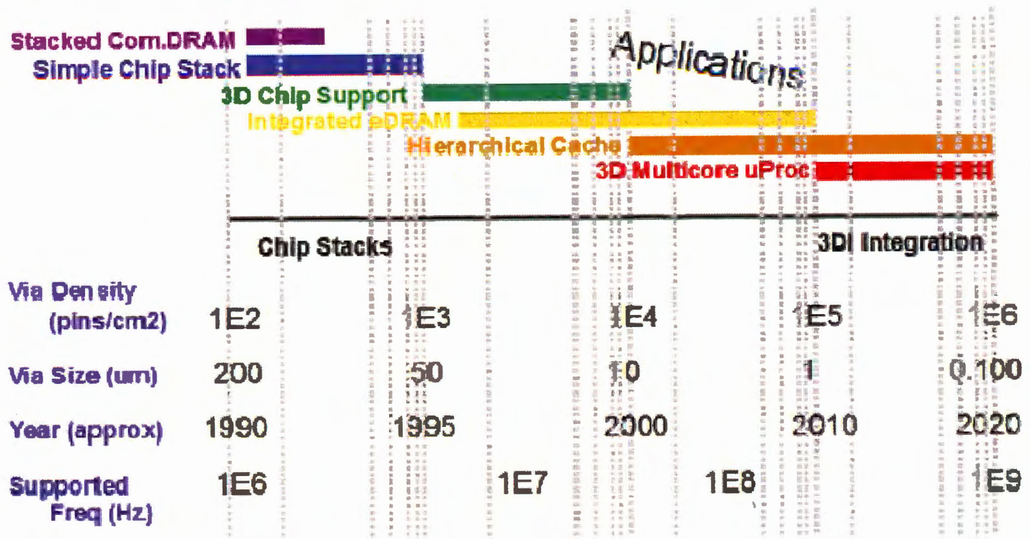


Figure 1.7: Technology range and applications of vertical interconnects. [10]

technologies will be presented in Chapter 6.

1.5 MAJOR CHALLENGES IN 3-D IC DESIGN

1.5.1 CAD Tools for 3-D ICs

3-D IC design technology and 3-D CAD are still in the formative stages of development. Therefore various CAD tools, including those for physical design, are being developed. An easy solution to the lack of physical design tools for 3-D ICs is to build quasi 3-D tools that are based on the extension of existing CAD tools for 2-D ICs. These quasi 3-D tools can handle simpler 3-D designs wherein existing 2-D designs are simply stacked and connected vertically without any major design changes. An obvious example of this type of design is the stacking of processor and memory dies. In this scenario, the only change needed is the insertion of TSVs in the layout to deliver signal, power, ground, and clock in the vertical

direction. The insertion of TSVs can be performed in the layout's white space or by slightly modifying the layout to leave the space for TSVs.

As the TSV technologies mature, their diameter is projected to shrink below 0.1μm and via density is expected to increase by an order of magnitude by the year 2020 [10] as shown in Figure 1.7. These future trends call for fine grained 3-D IC designs. Therefore there is a need for native 3-D CAD tools that are built from scratch for fine-grained 3-D optimizations such as 3-D module floorplanning, 3-D gate placement across consecutive device layers for performance, and power improvement. Furthermore, TSV-aware 3-D design verification and analysis tools such as timing, power, signal integrity, 3-D DRC/LVS, etc. need to be integrated and efficiently managed.

In the 2009 edition of ITRS [11], a new metric has been introduced that captures the physical design technology requirements for 3-D ICs. The metric is called "*percentage of native 3-D physical design technology in TSV-based 3-D IC implementation flow.*" This metric is around 10% at present, and is expected to grow further in future. Thus by the year 2025, all physical design tools that will be used for 3-D ICs are required to be *native 3-D* to meet the demand of future 3-D designs.

1.5.2 Heat Extraction and Thermal Management

Heat extraction and thermal management are some of the most significant challenges for 3-D ICs. The power density in 3-D ICs increases drastically due to stacked device layers that pack more circuits in a small footprint. Furthermore, due to enhanced performance, switching activity increases. Thus 3-D ICs generate more heat compared to the

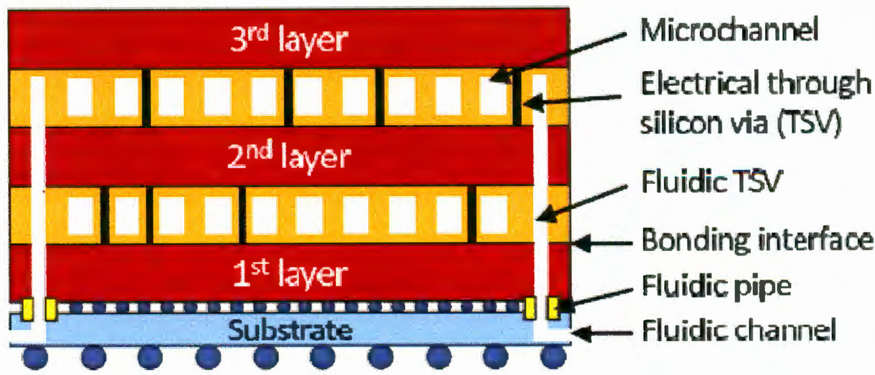


Figure 1.8: Cross sectional view of a 3-D IC using integrated microchannel cooling technology. [22]

conventional 2-D chips. However, there is only one source of heat removal (through a heat sink) for a 3-D IC, the same as a 2-D chip. Thus thermal modeling, simulation and accurate temperature estimation within 3-D ICs is important. This has led to the proposal to insert thermal vias that create different vertical paths from the upper device layer towards the heat sink [12]. Incorporation of the thermal model based on a resistive thermal network during 3-D floorplanning [13] and 3-D placement [14] has also been proposed. Based on the thermal model, hot modules which generate large amounts of heat are identified. These hot modules are placed closer to the heat sink whereas the cooler modules are placed away from the heat sink to minimize the peak temperature of 3-D ICs.

In addition to the above heat removal techniques, researchers have been actively searching for other innovative and alternative heat removal techniques. Microchannel liquid cooling [15] has been shown to be a promising solution [16]-[19] because the combination of microfluidic channels and liquid coolant media could offer very low thermal resistance between the chip surface and the coolant. Researchers from IBM [20],[21] have recently

demonstrated the feasibility of integrated microchannel 3-D ICs as shown in Figure 1.8. Microchannel heatsinks are distributed among different device layers, and cooling fluid is delivered to each microchannel heatsink through fluidic TSVs and pipes. The presence of a cooling path at each layer enables direct heat dissipation from each individual device layer [22].

1.5.3 Power and Ground Delivery in 3-D ICs

Power-ground (P/G) delivery in 3-D ICs is among the major challenges in 3-D technology. The on-chip power-ground networks in different device layers are vertically connected using P/G TSVs that demand higher current per TSV. The number of P/G TSVs is limited in order to prevent Placement & Routing (P&R) congestion. Furthermore, signal routing should be performed carefully in order to prevent coupling noise between P/G TSVs and signal wires. This leads to a complex optimization problem and research is needed on P/G network synthesis, optimization, and analysis to address the noise and power integrity issues along with minimizing congestion and the on-chip resources such as P/G wires, P/G TSVs, and on-chip decoupling capacitors [9].

1.5.4 Clock Tree Synthesis for 3-D ICs

Sequential circuit elements such as flip-flops and latches are basic building blocks in digital circuit design. In homogeneous 3-D ICs, they have the potential to be located in almost every device layer. To operate these circuits, clock signal delivery across different device layers is important while reducing power consumption, clock-skew, slew and jitter. Similar to signal and P/G TSVs, clock TSVs are used for clock delivery and they occupy layout space and cause coupling. Furthermore, the clock tree is the longest wire which contains

many buffers to control clock skew. Since the delay of clock wires, buffers and TSVs are significantly affected by temperature; extreme care should be taken for minimal skew based on the given non-uniform thermal profile within 3-D ICs.

1.5.5 TSV Management in 3-D ICs

As described in sub-sections 1.5.3 and 1.5.4, TSVs are used for establishing several kinds of electrical connections across different dies in 3-D ICs. In addition to providing vertical connections, they consume routing resources and may create congestion. The location of TSVs, via pitch and their diameters have significant impact on the quality and reliability of 3-D IC layout as well. The number of TSVs used in 3-D ICs depends on the design applications and their partitioning styles to divide circuits into multiple dies. Research work is needed to study the tradeoff among core-level, block-level and gate-level partitioning across the different device layers within 3-D ICs.

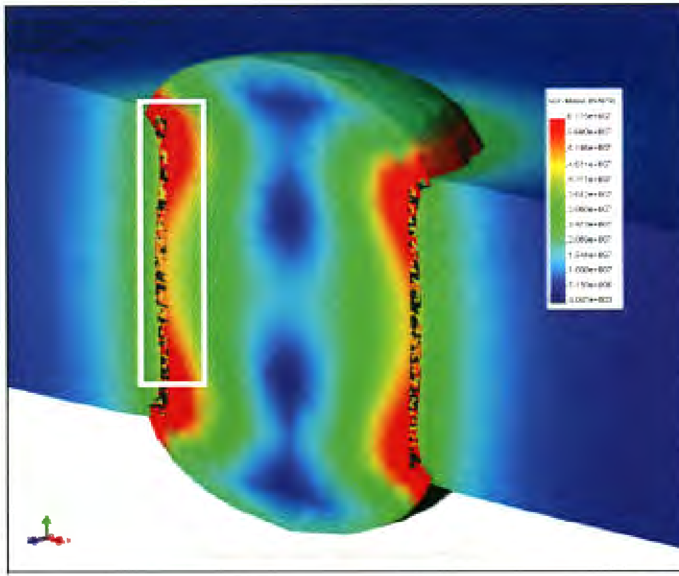
1.5.6 TSV-Induced Design for Manufacturability and 3-D IC Yield

TSVs in 3-D ICs cause non-uniform layout density distributions on the active, poly, and metal layer. This density variation causes variation during the Chemical Mechanical Polishing (CMP) steps for the individual die and requires new TSV-aware solutions. Furthermore, due to the high thermal profile in 3-D ICs, TSVs suffer from thermo-mechanical stress. The thermo-mechanical stress occurs due to a mismatch in the coefficient of thermal expansion (CTE) of copper TSVs and the silicon that surrounds them. As the temperature rises during the operation of a chip, TSVs expand more than the surrounding dielectric as shown in Figure 1.9(a). In this figure the high stress sites within 3-D ICs can be observed. Researchers from IBM have reported an SEM (Scanning Electron

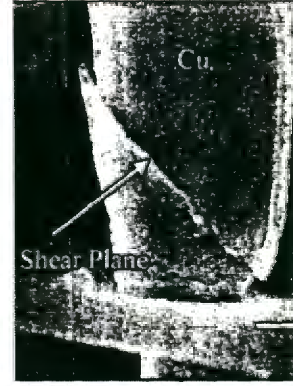
Microscope) image of a visible crack in a TSV due to high stress [23] as shown in Figure 1.9(b). Figure 1.9(c) shows high stress sites in a multi-layer 3-D IC that are prone to cracking. Thus TSVs are prone to failure or may attain plasticity due to the high thermo-mechanical stress resulting in significant yield loss.

Yield is directly related to the cost-effectiveness of 3-D technology [1],[24]. In spite of its importance, very few research works directly address the yield problem. Patti et al. [24] suggests insertion of redundant circuits in the uniform circuit architectures such as memories and FPGAs to make them reparable in the presence of defects. Recently, Ferri et al. [25] proposed the improvement in parametric yield of DTW and DTD integration methods by carefully matching the speed of the dies being matched in the stack. Smith et al. [26] proposed the matching of wafers in WTW integration to minimize the stacking of good dies with a bad die in order to enhance the yield. Finally, Smith et al. [8] presented the impact of 3-D IC yield on the cost of WTW, DTW and DTD stacking methodologies. These studies consider the failure of 2-D dies in the 3-D stack and assume that the 3-D integration process is defect free. However in real life, this assumption might not be accurate. It is a well known fact that TSVs suffer from thermo-mechanical stress and may attain plasticity [27],[28] causing the failure of vertical interconnects. Bentz et al. [27] proposed an FEM based model for the study of thermo-mechanical stresses of TSVs within 3-D ICs, and reported the variation in stresses due to via pitch, via diameter, and Benzocylcobutene (BCB) thickness which is used as a glue layer for stacking.

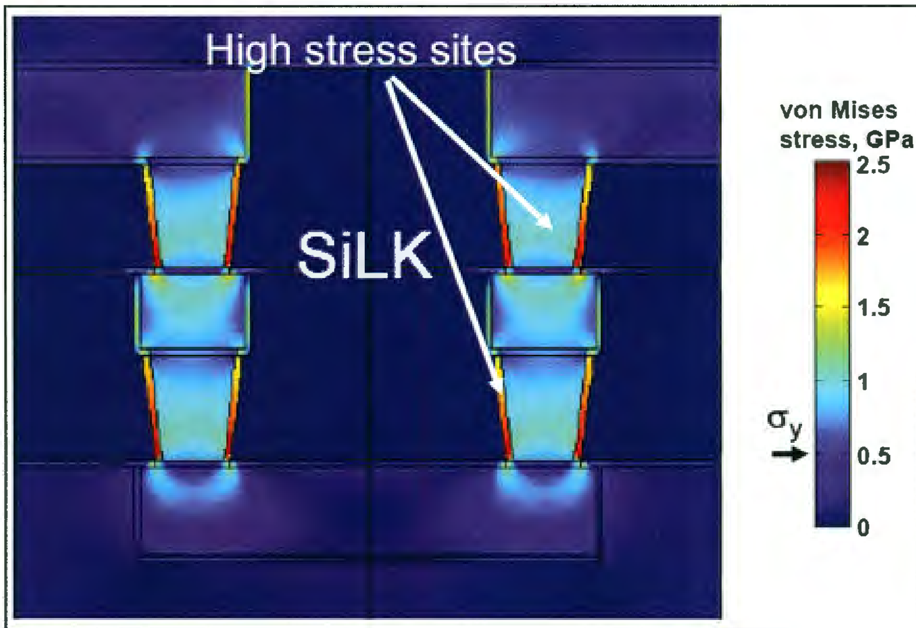
A recent study performed on a chain of 10K vias fabricated on a 10μm via pitch showed that only 67% of the physical vias were fully functional and met the desired electrical



(a)



(b)



(c)

Figure 1.9: (a) High thermo-mechanical stress shown by the white rectangle at the edge of a TSV [38], (b) An SEM image of a visible crack due to stress in a TSV [23], and (c) High stress sites in a multi-layer 3-D IC [27].

characteristics [29]. Another study shows 99.98% via operability of regular arrays of 256x256 vertical interconnects on a 30μm via pitch [30]. The large variation in yield could have been caused by the change in thermo-mechanical stresses [27],[28] due to difference in via density and via diameter. One important thing to note is that these studies were performed on a uniform configuration, which can be suitable for regular architectures such as memory and FPGAs, and redundancy in circuitry can be inserted to avoid yield loss [24]. In a 3-D system-on-chip (3-D SOC) with heterogeneous circuit architecture, a single TSV failure due to thermo-mechanical stress will cause the entire chip to fail and therefore reduce the yield. The circuit redundancy in a 3-D SOC might have too much overhead in terms of silicon real estate (i.e. area). Thus the effect of thermo-mechanical stress in TSVs leading to yield loss needs to be explored. Furthermore, yield-aware 3-D design techniques are required to improve 3-D IC yield.

1.6 CURRENT STATUS OF 3-D INTEGRATED CIRCUITS

There are a few commercial 3-D ICs which are present in the market at present. IBM has produced a 3-D power amplifier chip [123]. In addition, a 3-D image sensor by Toshiba was unveiled in the year 2008, and MEMS devices are being produced using 3-D technology [122]. Researchers from NCSU have fabricated test chips for a 3-D FFT, and a 3-D ternary content-addressable memory (TCAM) [121]. Georgia Tech's GTCAD lab has taped-out the first many core microprocessor from academia as a 2-layer 3-D test chip. These are small scale designs, and current research works in industry and academia focus on ASIC like 3-D integration, such as microprocessors stacked as CPU and L2 cache, and memory stacking on top of a microprocessor for improved speed and bandwidth.

1.7 CONTRIBUTIONS OF THE DISSERTATION

In section 1.5 we discussed the major challenges faced by the semiconductor industry in achieving mainstream acceptance of 3-D ICs. This dissertation focuses on two aspects of such challenges: a) CAD tool design for *native 3-D* floorplanning, and b) yield improvement of 3-D ICs in the presence of TSV failure.

1.7.1 CAD Tool Design for Native 3-D Floorplanning

Most of the current designs are composed of various modules/blocks and require floorplanning before placement can be performed. A good floorplan provides compact chip area and improves the timing performance of a circuit. Floorplanning is a well studied problem for 2-D ICs. Moving from 2-D to 3-D ICs, however, increases the complexity and the solution space [45]. Therefore development of efficient floorplan algorithms is important. As discussed in sub-section 1.5.1, the existing floorplanning [13],[45],[64],[65] tools are actually quasi-3-D floorplanning tools, i.e. they simply perform 3-D floorplanning of 2-D modules only. Therefore they miss the opportunity for 3-D placement of logic cells and hence they will not be able to completely harness the advantages of 3-D ICs. In other words, there exists a significant gap between existing 3-D floorplanning and 3-D placement tools that prohibits the usage of 3-D placement.

To bridge the existing gap between 3-D floorplanning and 3-D placement, a novel placement-aware 3-D floorplanning algorithm has been designed (Chapter 4). It uses stochastic wirelength distribution models for rectangular 2-D and 3-D modules (Appendix B) to estimate the advantages of 3-D module placement (in terms of wiring reduction) and

identifies potential 2-D modules that should be converted into 3-D modules to harness the advantages of 3-D placement. A set of constraints have been derived (Chapter 3) on the topological representation of floorplanning that identifies feasible solutions and eliminates infeasible solutions quickly in the proposed placement-aware floorplanning. Thus it reduces the solution search space and improves the runtime. Finally a novel module-packing algorithm has been designed that quickly computes the geometrical 3-D floorplan (Chapter 3) while satisfying the set of constraints for placement-aware 3-D floorplanning.

Furthermore, a 3-D floorplanning algorithm with module alignment has been designed (Chapter 5) that can be useful in bus-driven 3-D design. It also considers heterogeneous 3-D stacking in which a certain set of modules can only be placed in specified types of substrates/device layers due to technology and the thermal profile. In addition, a set of modules can be specified by designers that need to be placed away from each other or need not overlap due to thermal or noise sensitivity. This algorithm also uses the set of feasibility constraints and fast 3-D packing algorithm presented in Chapter 3.

This work falls under the constrained combinatorial optimization problem category. Most of this research has been published in IEEE sponsored international conferences and journals. For example, the statistical wirelength distribution models for rectangular 2-D and 3-D modules (Appendix B) is published in [32]. The initial version of the proposed placement-aware 3-D floorplanning algorithm appears in [33]. In addition, an improved version of the algorithm (Chapter 4) that uses a set of constraints for feasible solutions (Chapter 3) along with the fast 3-D module packing algorithm satisfying such constraints

(Chapter 3) is currently in press for publication in IEEE Transactions on Very Large Scale Integration Systems [34].

1.7.2 Yield Improvement of 3-D ICs in the Presence of TSV failure

As discussed in sub-section 1.5.6, TSVs are prone to failure due to thermo-mechanical stress within 3-D ICs, and may cause serious yield loss. To mitigate the yield loss problem of 3-D ICs, a novel set of yield improvement strategies that focus on defects in through silicon vias is proposed. Furthermore, the parametric yield and chip revenue that can improve profitability are estimated. A quantitative analysis of the impact of our approach on chip area, delay and power is also studied. Initially Monte-Carlo simulations are used to evaluate the effect of our proposed yield enhancement techniques. The Monte-Carlo simulation results demonstrate that our strategies can improve the 3-D IC yield significantly.

Since Monte-Carlo simulations are computationally expensive and time consuming, we derive a set of analytical models for the most promising yield improvement strategies. Furthermore, analytical models for parametric yield such as slow and fast chips, and an expression for the sweet spot between them are derived. The sweet spot insures that the number of fast chips produced for a particular yield improvement technique will always be greater than the number of slow chips. The analytical models can be very helpful in quickly analyzing the functional and parametric yield for a given 3-D design. In addition, it can also be incorporated in the physical design CAD tools such as floorplanning and P&R for yield-aware 3-D IC design.

This part of the research has been initially published in an IEEE sponsored workshop [35], and another in a conference [36]. In addition, a matured version of this work (Chapter 7) along with analytical models (Chapter 8) has been submitted to IEEE Transactions on Very Large Scale Integration (TVLSI) Systems and is currently under review [37].

1.8 STRUCTURE OF THE DISSERTATION

The remaining part of this dissertation is organized as follows:

Chapter 2 presents the formal discussion about floorplanning in 2-D and 3-D ICs. It presents information about the different types of topological representations for 2-D floorplanning and their extensions to 3-D floorplanning. Next, a summary of previous work on 3-D floorplanning is presented. The second half of Chapter 2 includes a brief overview of the initial software that was inherited at the beginning of the research. Then an overview of changes made during the research to solve the targeted placement-aware 3-D floorplanning problem is presented.

Chapter 3 introduces the concept of vertical constraints during 3-D floorplanning and its importance. Then a set of feasibility conditions are derived on a popular topological representation of floorplanning and they are later extended to a graph-based formulation. Furthermore, a set of module packing algorithms satisfying these vertical constraints are presented.

Chapter 4 contains the architecture of the proposed placement-aware 3-D floorplanning algorithm, its implementation details followed by experimental results and its comparison with existing state-of-the-art 3-D floorplanning tools.

Chapter 5 includes the problem formulation for the 3-D floorplanning with Module alignment, its importance, implementation details and experimental results.

In Chapter 6, the TSV-induced 3-D IC yield problem is formulated, and other possibilities of vertical interconnects are discussed. Chapter 7 presents the proposed yield improvement techniques and Monte-Carlo simulation results for functional yield and parametric yield. In addition, it also presents tradeoffs in terms of area, delay and power associated with different yield improvement strategies. Chapter 8 contains the derivation of analytical models for the most promising strategies and comparison of analytical models' results with Monte-Carlo simulation results.

Chapter 9 presents the conclusions and future work, and Chapter 10 presents the summary of major contributions.

Appendix A contains the mathematical proofs of feasibility condition theorems discussed in Chapter 3. Appendix B presents the derivation of rectangular 3-D wirelength distribution models that are used to estimate the advantages of 3-D placement for different modules during the placement-aware 3-D floorplanning. Appendix C presents a methodology for Rent's parameter estimation, and a list of Rent's parameters assigned to each module of floorplanning benchmark circuits for performing experiments. Finally, the sensitivity analysis of the cost functions of 3-D floorplanning algorithms are given in Appendix D.

CHAPTER 2: FLOORPLANNING OF 2-D AND 3-D INTEGRATED CIRCUITS

Floorplan design is an important step in physical design of VLSI circuits to plan the position of circuit blocks or modules within a minimum area along with several other design cost parameters such as wirelength, substrate noise, temperature, power, etc. in order to optimize circuit performance. Floorplanning is performed after circuit partitioning in the physical design cycle. Once the module positions are chosen during floorplanning which ensures the optimization of various design cost parameters, the positions of modules are fixed in the placement stage, i.e. they cannot be moved to other locations in order to maintain the optimized values of the design cost parameters. With the advent of deep sub-micron technology and advancement in VLSI technology there has been an increasing need for early floorplanning which enables design budgeting for major design parameters such as wiring, area, substrate-noise [39], power/thermal estimation [40], and bus planning [41] when exact circuit information is unknown. If the floorplanning stage is neglected, placement might not meet these timing/power/thermal parameters, which can result in failure of the chip. Thus floorplanning guides the placement stage, which helps in meeting design parameters.

2.1 SHAPES OF CIRCUIT MODULES IN A FLOORPLAN

The shapes of circuit modules are usually assumed to be rectangular and the aspect ratio for each module is defined as the ratio of the width to the height. The modules are classified as hard and soft modules. In the early stages of the design cycle, the exact dimensions of the modules are unknown. Thus it is reasonable to make assumptions about the area of each

block and have a flexible aspect ratio. Hard blocks have fixed aspect ratios while soft blocks have a fixed aspect ratio range. In addition to the rectangular shapes of modules, rectilinear shaped blocks are occasionally desired to save chips area. There are two types of rectilinear blocks *a)* convex rectilinear which has an "L" or "T" shape, and *b)* concave rectilinear which has a "U" shape. These rectilinear shapes are usually handled by partitioning them into small rectangular blocks, and imposing rectilinear constraints on the floorplan representation such that the partitioned parts of a rectilinear block preserve its original rectilinear shape. The partition based rectilinear shape constraints have been reported on several floorplan representations such as on SP [68],[100],[116], BSG [114][115], O-tree [117], CBL [118], B* tree[119], and TCG [120]. Although the rectilinear block packing can be done using any of these representations, it is easy to satisfy these constraints on sequence pair for which a fast module packing algorithm satisfying these constraints has been proposed in [100].

In this research, only hard modules with rectangular shapes have been considered because *a)* rectilinear shapes can be constructed by the methods described in the previous paragraph, and *b)* rectilinear blocks are not very frequently present in designs.

2.2 CLASSIFICATION OF FLOORPLANNING

Floorplanning can be classified in two categories: slicing and non-slicing. A slicing floorplan is obtained by recursively bisecting a rectangle using a horizontal or vertical line. It has a smaller solution space which implies a faster runtime. However, it limits the set of reachable layout topologies [42] and can degrade layout density, especially when modules are of

different sizes. Slicing floorplans were predominantly used when design complexity was not a major concern. Unfortunately, most of today's real designs are complex and of the non-slicing type, which requires complex topological representations for floorplanning. In addition, fixed outline floorplanning has been proposed by researchers in which module packing is done within a specified die size [31],[48]. To use the area within fixed outline floorplanning, rectilinear shape modules have been proposed. The method for handling rectilinear blocks has been described previously in section 2.1. In recent years, it has become mandatory for floorplan designers to move to non-slicing floorplanning [42],[43],[44]. Thus only non-slicing floorplanning is considered in this work.

2.3 COMPLEXITY OF THE FLOORPLANNING PROBLEM

Floorplanning is an NP-hard problem and its solution search space increases exponentially with the problem size. Due to the simultaneous optimization of various design parameters (such as area, wirelength, noise, etc.), floorplanning becomes a combinatorial optimization problem. For a design consisting of "*n*" modules, the solution search space for a 2-D floorplanning is $(n\,!)^2$ [45]. Moving from 2-D to a k-layer 3-D floorplanning the solutions space further increases to $n^{k-1}(n!)^2/(k-1)!$ [45]. Let us consider 2-D/3-D floorplans containing 50 modules (i.e. $n = 50$) which is a reasonable size of floorplan according to the MCNC benchmark. In this scenario, the size of the solution search space will exponentially grow and today's modern computers cannot solve it within an acceptable/limited time by any deterministic search approach. Thus stochastic search based algorithms which search for an acceptable (near optimal) solution within a limited amount of time, are efficient for

the floorplan design. These stochastic algorithms iteratively perturb the solution space with the hope of improving the quality of the solution.

Simulated annealing is one of the most popular optimization engines that have been used by floorplan designers [46],[47],[48],[49],[50]. Simulated annealing is based on the analogy of annealing in metallurgy in which heating is used above recrystallization temperature of a material which allows the molecules to separate. In the next stage, cooling is gradually applied and molecules are annealed to refine the material's structure, induce ductility, and relieve internal stresses. Following the analogy, simulated annealing starts with a randomly generated initial solution, and iteratively searches for a better solution using a pre-defined set of moves. It involves a temperature scheduling in which the temperature is set higher at the beginning, and is lowered iteratively upon improvement of the solution quality.

An evolutionary algorithm is an alternative method for stochastic search which is based on Darwin's theory of evolution and starts with an initial set of solutions (i.e. population) in parallel. The individual solution in the population competes for survival among other individuals of the population. Following nature's rule, the individuals evolve from one generation to another using mutation and reproduction processes. In the mutation process, the properties of an individual are changed by perturbing the solution quality using a pre-defined set of moves, and an offspring is generated. In the reproduction process, the properties of two individual are mixed, and one or more offspring are created. The parents and offspring compete for survival, and best fit individuals become the next generation parents. Evolutionary algorithms have been used by several researchers [33],[34],[51],[52] for floorplan design.

2.4 2-D FLOORPLAN REPRESENTATIONS

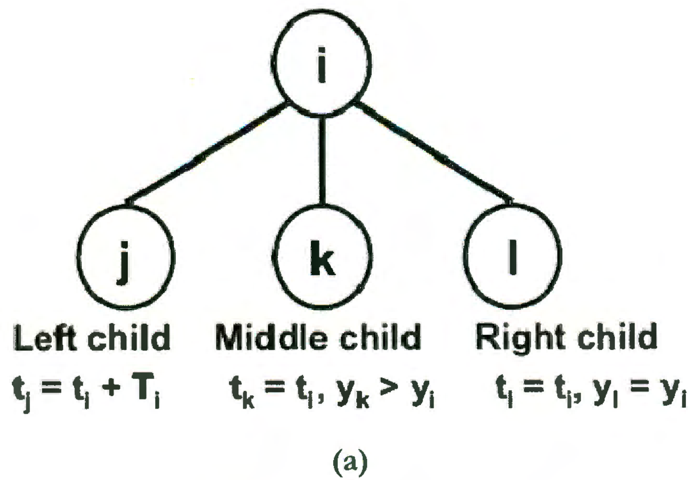
Due to the NP-hard complexity of floorplanning problems, it is important to use a suitable topological representation for representing floorplans. The topological representation should be simple, effective and the algorithm to transform it to a geometric floorplan should take the least computational effort. Researchers have come up with various non-slicing floorplan representations such as Sequence Pair (SP) [53], Bounded Sliceline Grid (BSG) [54], O-tree [55], B*-tree [44], Corner Block List (CBL) [56], and Transitive Closure Graph (TCG) [57]. These representations can be distinguished based on the *P-admissibility* [53] of the solution space which satisfies the following requirements: *1)* the solution space is finite, *2)* every solution is feasible, *3)* packing and cost evaluation can be performed in polynomial time, and *4)* the best evaluated packing in the space corresponds to an optimal floorplan [53]. The *P-admissible* representations are BSG, SP and TCG. Both SP and TCG have one-to-one mapping between their topological representations and corresponding floorplans. In contrast, BSG itself generates multiple representations corresponding to one packing which implies larger solution space than SP and TCG. The runtime for packing from sequence pair is only $O(n \lg \lg n)$ [58] compared to TCG and BSG which have $O(n^2)$ complexity [54],[57]. Given an O-tree or a B* tree, it may not be feasible to find a packing corresponding to the original representation, and thus they are not *P-admissible.* Corner block list (CBL) has a smaller solution space and faster packing scheme but it cannot guarantee a feasible solution in each perturbation, and thus it is not *P-admissible* as well.

P-admissibility provides a standard for the classification of floorplan representations. Since P-admissible representations have one-to-one mapping between the topological

representation and their geometric floorplan, these representations have smaller solution space compared to those representations which generate infeasible solutions, have one-to-many or many-to-one mapping between floorplan representations and their geometric floorplan. Considering the large solution space of floorplanning (as described in section 2.3), p-admissible representations can be a better option. It does not mean that non p-admissible representations cannot produce good floorplan solutions, due to NP hard nature of the problem and the stochastic search process, which is very random. However, a comparison of floorplan solution quality obtained using different *p-admissible* and *non p-admissible* representations shows that p-admissible representations produce better solution quality [57].

2.5 3-D FLOORPLAN REPRESENTATIONS

The topological representations of 3-D-floorplanning can be grouped into two classes: the *true 3-D* and the *quasi-3-D*. The true-3-D representations contain placement information for modules in the *x*, *y*, and *z* directions, and modules can be treated as solid cuboids with finite *length*, *width* and *height* as shown in Figure 2.1(b). Researchers have extended several existing 2-D representations of a floorplan to true 3-D representations, e.g. 3-D reconfigurable functional unit operation (RFUOP) [59], 3-D slicing tree [60], and sequence triple [61], in which representation for the third dimension is appended. Present 3-D technology has a limited number of device layers and the inter-layer height is fixed. Thus true-3-D representations would have more redundancy in the z-axis data structure resulting in an increased time and space penalty. A T-tree representation was presented in [62], which also considers modules as 3-D boxes. The T-tree representation was inspired by the



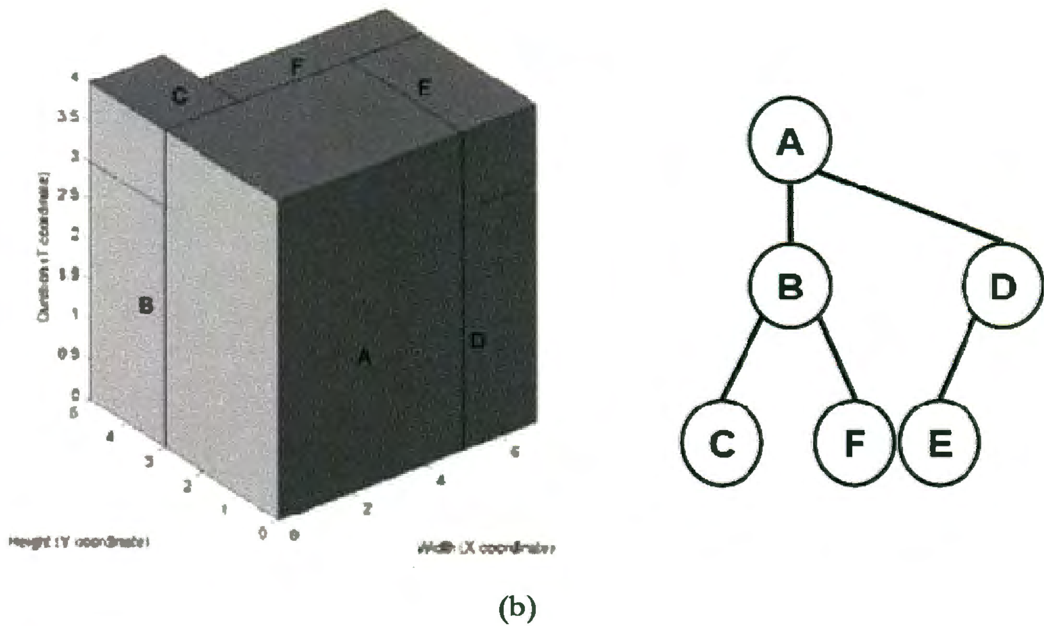


Figure 2.1: (a) the structure of a T-tree, (b) A compacted placement of modules and the corresponding T-tree. [62]

binary representation of a 2D floorplan called B* tree [44] in which a node (module) has at most a *left child* and a *right child* that represent the dimensional relation of modules in a 2-D plane. In the T-tree representation a *middle child* is added which represents the information along the third dimension as shown in Figure 2.1(a). The T-tree represents the geometric

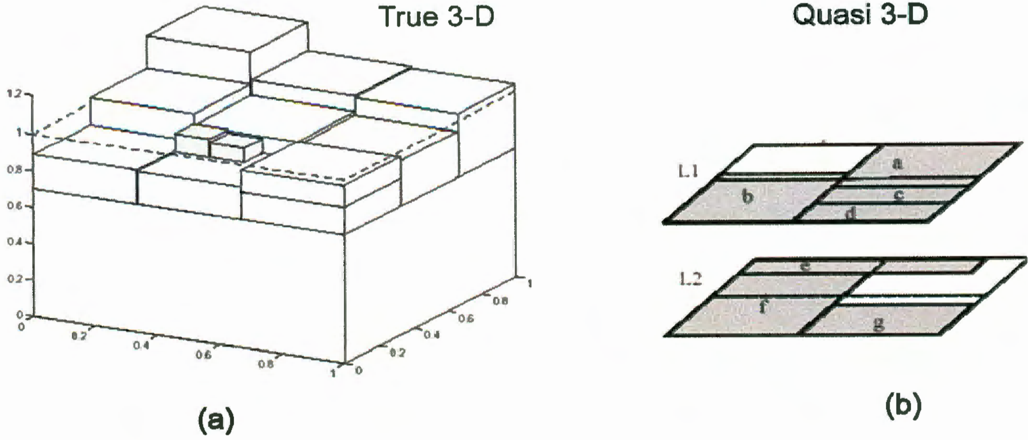


Figure 2.2: (a) true 3-D floorplanning [125] (b) quasi 3-D floorplanning [13].

relation between two modules as follows. If a node n_j is the left child of node n_i, then module *j* must be placed adjacent to module *i* in the T+ direction where T is the third dimension [62]. If a node n_k is the middle child of node n_i, module *k* must be placed in the Y+ direction of module *i* with the t-coordinate of module *k* equal to the t-coordinate of module *i* (i.e. $t_k = t_i$, and $y_k > y_i$). Finally, if node n_l is the right child of node n_i module *l* should be placed in the X+ direction of module *i* with the t and y-coordinates equal to those of module *i* (i.e. $t_l = t_i$, $y_l = y_i$). Although it reports linear time complexity for operations on T-tree, the packing scheme has $O(n^2)$ complexity. An example of a T-tree and its corresponding module packing is shown in Figure 2.1(b).

The quasi 3-D representation constructs floorplans of different device layers with an array of 2-D-representations. Each device layer has its own 2-D representation. Unlike true 3-D representations, module placement information along the z-axis is not included, and modules do not have a height in the quasi 3-D representation. The examples of quasi 3-D

representations are two-layer BSG [2], four-layer SP [63], and four-layer TCG [13]. Using these representations, intra-layer operations within a layer and inter-layer operations across different layers are performed for the perturbation of solution space. Using quasi 3-D representations, many 3-D floorplanning algorithms have been reported recently in the literature. In Chapters 4 and 5, we will compare our floorplanning results with three existing state-of-the-art 3-D floorplanners.

2.6 PREVIOUS WORK ON 3-D FLOORPLANNING

Cong et al. [13] proposed a thermally driven 3-D floorplanning algorithm. It uses a compact resistive thermal model along with simulated annealing for optimization. Although the compact resistive thermal model is reasonably fast, it is still much more computationally expensive (9.7x) compared to non-thermal-driven floorplanning. In order to mitigate the computational overhead of thermal driven floorplanning, the authors of [13] have presented approximate closed form solutions of heat flow in vertical and horizontal directions. The closed form heat flow solution is very fast, but less accurate. Thus Cong et al. have also presented a hybrid approach in which the less accurate but fast closed form equations are used during simulated annealing, but accurate thermal resistive network model is used for 20 consecutive iterations whenever the temperature of the simulated annealing process drops down. Thus it provides a tradeoff between accuracy and computational overhead which is reduced to 3.2x compared to 9.7x when the accurate model was used. This work was the first to address the thermal optimization during 3-D floorplanning.

Hung et al. presented an interconnect and thermal aware floorplanning algorithm for 3-D microprocessors [64]. Hung et al. have developed a thermal modeling tool called HS3D to identify the hot spot and temperature of modules during floorplanning. The authors use the B* tree representation (quasi 3-D) and simulated annealing engine for optimization. The algorithm considers power consumption in wires during optimization. Hung et al. used the detailed model of an Alpha-like microprocessor in Verilog and extracted the actual power consumption in function modules and intra-module interconnects using *Design Compiler* and *First Encounter* tools. Then they used a 2-D floorplanning tool to compute the inter-module wirelength. By summing the inter-module wirelength and intra-module wirelength, they obtain the total wirelength in a 2-D chip, and use the power consumption in interconnects from First Encounter as real power data. Next they normalize the inter-module wirelength of a 3-D chip (during floorplanning) with respect to the inter-module wirelength of the 2-D chip. The normalized value is multiplied by the power consumption in all wires of the 2-D chip to scale and estimate the power consumption in the 3-D chip. The area, aspect ratio, and power consumption obtained after placement & routing is used as input to the floorplanning algorithm. The algorithm is unique in a sense that it uses the power consumption in interconnects along with the power consumption in circuit module to estimate the peak temperature of the chip.

Li et al. proposed a hierarchical 3-D floorplanning algorithm for wirelength optimization [45]. The algorithm proposes a statistical method to partition and place small sets of modules in each device layer, and then performs 2-D floorplanning in each device layer without performing any inter-layer moves. Thus it decomposes the 3-D floorplanning

problem into multiple 2-D floorplanning problems. Therefore any good 2-D floorplanning algorithm can be used in this approach.

Zhou et al. recently proposed a scalable temperature and leakage aware 3-D floorplanning algorithm [65]. It uses an adaptive force directed technique, and integrates a power-thermal analysis to close the leakage-thermal feedback loop. The authors define different types of directed forces: *a)* thermal force and *b)* filling force. The thermal force is used to minimize temperature by keeping two hot modules away from each other. The filling force is used to eliminate overlap between blocks, and to evenly distribute modules. The power-thermal analysis model is used to minimize the leakage power along with chip temperature, area, wirelength, and via count. The reported floorplan solutions are better and the floorplanner is fast compared to the thermal-driven 3-D floorplanning tool presented in [13].

These 3-D floorplanning algorithms provide different methodologies and focus on different sets of objectives during floorplan optimization. Each of them is unique and ground breaking. However, one important thing to notice is that they all perform 3-D floorplanning of 2-D modules only, and thus they limit the advantages of 3-D integration.

2.7 A BASIC 3-D FLOORPLANNING TOOL

At the beginning of this research work, a basic 3-D floorplanning tool was inherited from Benyi Wang, a previous researcher in the research group [101]. It is based on an evolutionary algorithm and Sequence Pair representation. A 3-D floorplan is represented by a set of sequence pairs which is called a *Grouped Sequence Pair* (GSP), proposed by Yu Xia for test scheduling of core based SoC[102]. It starts with a randomly generated initial set of

solutions (parents). Each solution goes through stochastic perturbation, and new solutions (offspring) are generated. Parents and offspring compete for survival and best fit set of solutions become the next generation parents. The algorithm iteratively searches for a better floorplanning solution (using a pre-defined set of moves) until termination criteria is reached.

The tool included a module-splitting move which randomly used to select a module and aggressively used to split it into *m* parts, where *m* was the total number of device layers in a 3-D chip. Although it had a *split* move incorporated, it did not have any mathematical model to quickly estimate the cost of splitting. Furthermore, the cost function of the floorplanner included *area* and *inter-module wirelength* only. As a result, the *split* move had virtually no effect on the solution quality because the optimization engine only used to see an increase in area due to splitting, but it could not see any positive effect. Thus it mostly used to reject floorplan solutions with split modules. In addition, its 3-D constraint graph based module packing algorithm had $O(n^3)$ time complexity which made the tool very slow. This packing algorithm will be elaborated in Chapter 3.

The basic algorithm (which was inherited) was at the very initial stage of development. However, it provided a framework for continuing research as a part of this dissertation. Figure 2.3 shows the flow chart of the initial 3-D floorplanning tool that was inherited at the beginning of this research. Please note that the shaded parts of the algorithm in Figure 2.3 indicate that significant changes were made as a part of this research. A detailed explanation of these changes will be presented in Chapters 3, 4, and 5.

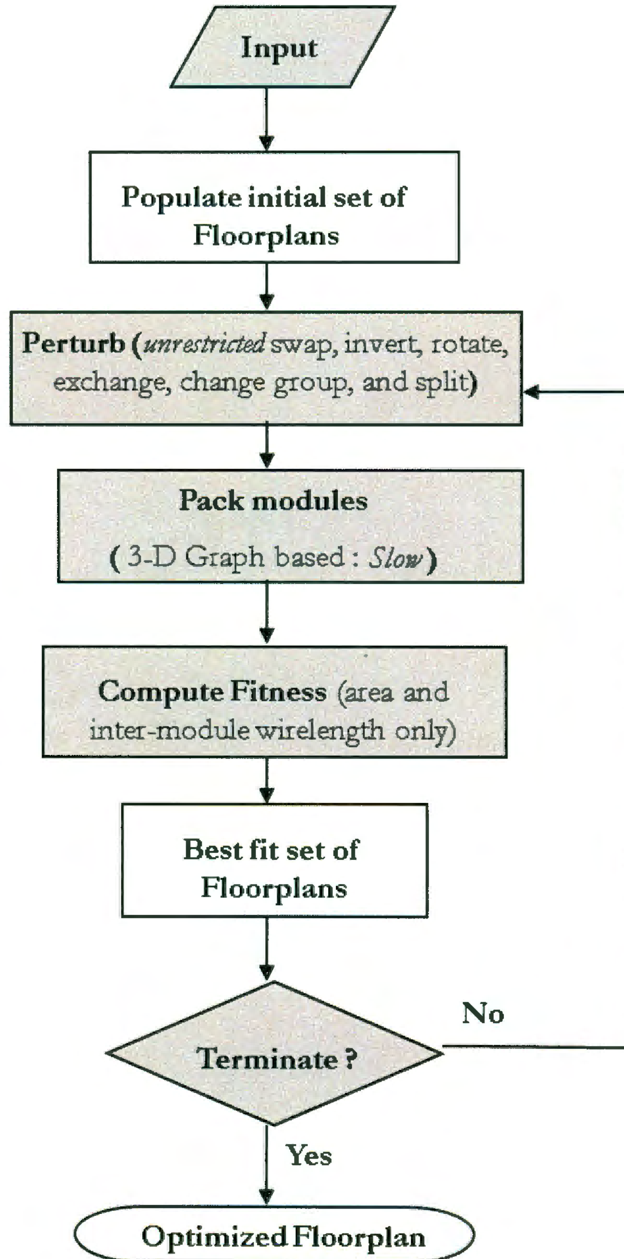
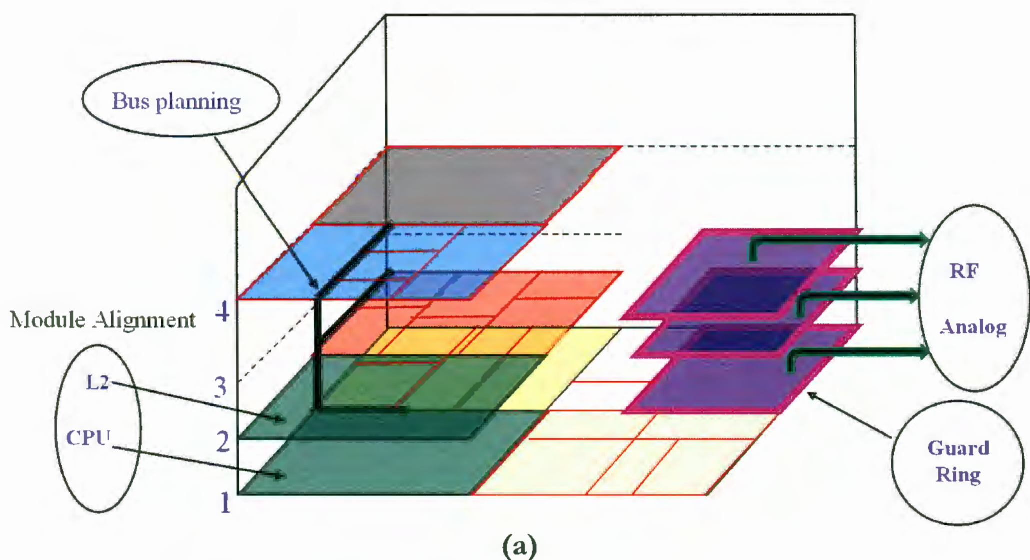


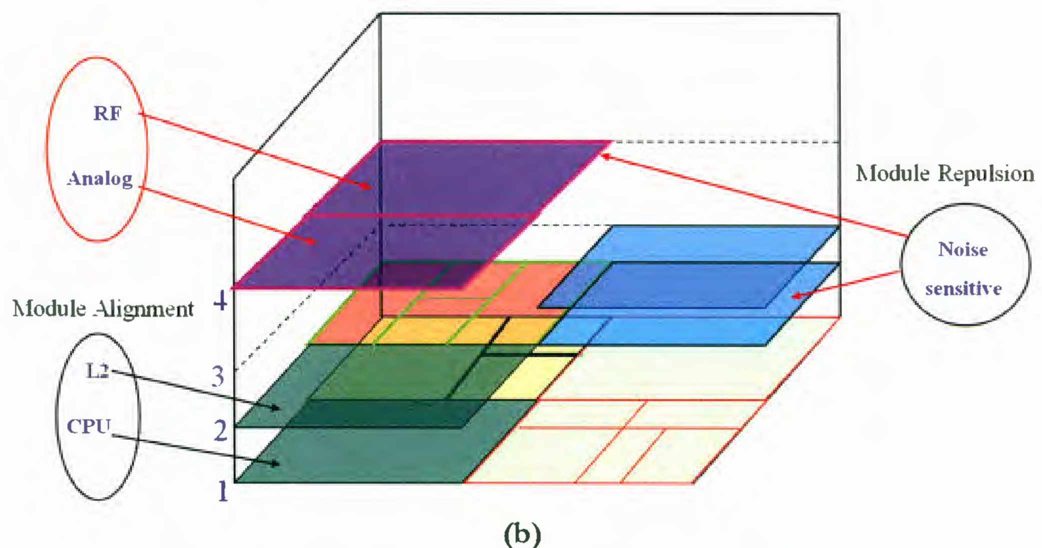
Figure 2.3: Flow-chart of a basic 3-D floorplanning tool inherited at the beginning of the research. The shaded boxes indicate the portions of the algorithm that have been modified as a part of this research work.

CHAPTER 3: VERTICAL CONSTRAINTS IN SEQUENCE PAIR REPRESENTATION

In this chapter, we focus on the vertical constraints of multi-layer 3-D ICs that enable vertical alignment of modules in different device layers. Vertical alignments, such as overlapping or non-overlapping constraints, might be necessary in designs such as a microprocessor composed of CPU and L2 Cache sub-blocks that are placed in consecutive device layers. The vertical alignment is also required in a bus-driven floorplanning design [66] where a set of blocks can be connected by a rectangular strip of bus (horizontally or vertically). In another application, a group of modules might be required to be vertically aligned within a large digital system. Examples of different applications of vertical constraints are shown in Figure 3.1. Furthermore, in a 3-D SoC design in which every device layer has digital and analog blocks as shown Figure 3.1(a), analog blocks residing in different device layers may have to be vertically aligned for close connections, and isolated from the rest of the circuits to avoid interaction with noisy blocks. An example would be an audio amplifier block, TV output, and LCD driver, stacked on top of each other, with the image processor integrated close to the LCD driver (on the same device layer). Similarly, a video/camera interface is integrated on the same device layer as TV, and the audio amplifier is close to the DSP block. In another case, if analog blocks have been placed in a particular device layer, these blocks may need to be kept away from noisy digital blocks as shown in Figure 3.1(b). An example of such a scenario is an FM and an audio amplifier block fabricated close to each other, yet separated away from a DSP or/and an image



(a)



(b)

Figure 3.1 : Example of vertical constraints in 3-D SoC (a) module alignment, bus planning, and each layer containing analog/RF blocks are vertically aligned together, (b) analog/RF blocks have been assigned in on the top layer and they have been separated from noisy digital block.

processor block. These requirements must be known and preserved during the floorplanning phase.

In 2006, Law et al. [67] presented a vertical alignment in a 3-D floorplan using layered TCG to address the bus-driven 3-D floorplanning. However, the layered TCG based 3-D floorplanning is very slow and runtime grows rapidly with problem size (the runtime will be compared in Chapter 5). Furthermore, vertical constraints on other competitive representations such as sequence pair, and identification of feasible solutions to reduce the solution search space have not been explored. We will derive the vertical constraints on sequence pairs in this chapter.

3.1 INTRODUCTION TO SEQUENCE PAIR

In the sequence pair (SP) representation of a floorplan, two permutations ($<\Gamma^+>; <\Gamma^->$) of a set of modules are sufficient to define a 2-D packing [53]. For example:

(<…a … b>; < … a … b …>)	place ***a*** to the left of ***b***
(<…a … b>; <…b … a.….>)	place ***a*** above ***b***

Thus a sequence pair provides the relative positions of modules without their physical information. Given a sequence pair, one can construct an oblique lattice structure with the lines labeled in the same order as they appear in Γ^+ and Γ^-. For example, let us assume that ($<\Gamma^+>; <\Gamma^->$) = (<a, b, c, d>; <c, a, d, b>). Each module is placed at a lattice point which is at the intersection of two lines with the same label as shown in Figure 3.2(a). Figure 3.2(b) shows the resultant placement of modules satisfying their relative position as defined by the sequence pair. Please note that if the shapes of modules are different, then the floorplan for the same sequence pair will be different. Figure 3.2(c) shows an example of a different floorplan of the same sequence pair due to a change in the sizes of modules.

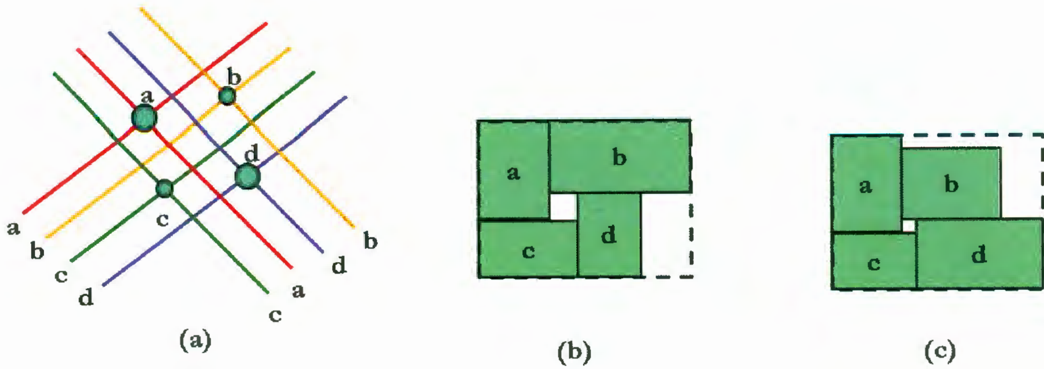


Figure 3.2: (a) oblique grid for $\Gamma^+ = \{a\ b\ c\ d\}$ and $\Gamma^- = \{c\ a\ d\ b\}$, (b) resultant placement of blocks, and (c) different packing for the same sequence pair due to change in sizes of modules.

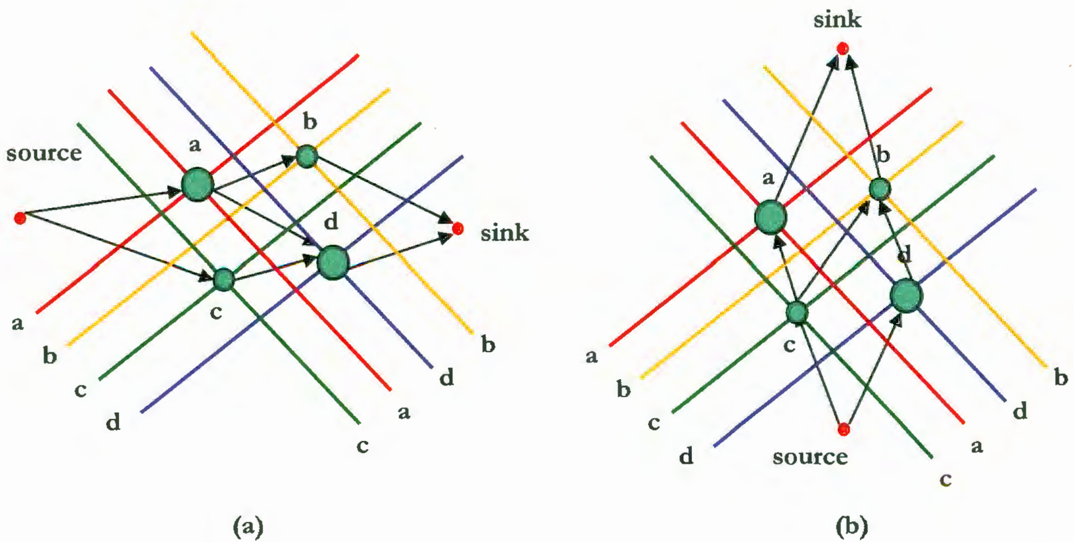


Figure 3.3: Weighted graphs G_h (left) and G_v (right) of the sequence pair shown in Figure 3.2

Two weighted, acyclic digraphs (i.e. directed graphs) are constructed in which the modules at the lattice points form a vertex set. Figure 3.3 shows a method for constructing the horizontal and the vertical constraint graphs. In the horizontal graph G_h an edge is placed from module i to module j iff "i is to the left of j". Then source and sink vertices are added to each graph. A weight is associated with each edge of the horizontal (or vertical) graph which indicates the width (or height) of the respective module. A longest path algorithm applied on the horizontal and vertical graphs computes the width and height of the chip as well as the co-ordinates of each module.

The Sequence Pair representation is extended to a pseudo 3-D floorplan representation. A classification of pseudo 3-D floorplan representations is given in Chapter 2 (see Section 2.5).Thus a 3-D floorplan is represented by a set of sequence pairs, and we call it a *Grouped Sequence Pair* (GSP). Each sequence pair of the GSP represents the module packing of one particular device layer. The sequence pairs representing different layers are independent of each other. Therefore inter-layer relations between modules are not included. Figure 3.4 shows the example of a two-layer grouped sequence pair and its corresponding floorplan. The advantages of GSP representation are that a GSP preserves all good properties (which were previously discussed in Section 2.4) of sequence pairs defined by *P-admissibility*, and it is very easy to perturb the solution search space on the GSP topological representation. Furthermore, it is possible to design a fast module packing algorithm that satisfies vertical constraints on the geometrical floorplan. A fast packing algorithm has been developed as a part of this research and it will be presented in Section 3.6. Similar to the grouped sequence pair, it is possible to define vertical constraints on other quasi 3-D floorplan representations

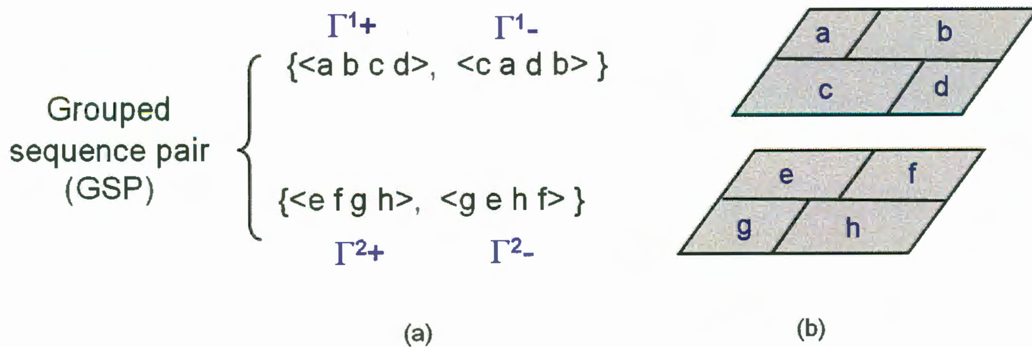


Figure 3.4: (a) An example of a two-layer grouped sequence pair, and (b) its corresponding floorplan in two device layer.

as well. However, the key point is whether the module packing and easy perturbation of the solution space can be achieved or not while satisfying the vertical constraints. For example, BSG, B*-tree, O*-tree, and layered TCG all have graph based module packing algorithms which have longer runtime than grouped sequence pair representation. Furthermore, satisfying vertical constraints further increases the runtime complexity of the module packing algorithm. Satisfying vertical constraints on true 3-D floorplan representations such as T-tree, 3-D slicing tree, and sequence triple will be easy because these representations treat modules as rectangular 3-D boxes. However, the solution search space has more redundancy in the z-axis data structure as described in Section 2.5.

3.2 VERTICAL CONSTRAINTS ON SEQUENCE PAIRS

Vertical constraints are specific relations between blocks that are assigned to different device layers. They are specified by defining geometrical relations between the xy-coordinates of the bottom-left corner of blocks. The vertical constraints require geometrical information of module packing. As we discussed in the previous section, this geometrical

information is not available in the sequence pair representation. However, blocks are assigned to different device layers and their vertical constraints should be established among multiple sequence pairs. Since the sequence pair specifies the relative positions of modules in a device layer, it is possible to identify a set of feasible GSPs that may lead to the fulfillment of the vertical constraints during geometric floorplanning. Thus in this work we derive feasibility conditions that identify feasible GSPs, which lead to satisfaction of vertical constraints during the geometric floorplan. These feasibility conditions can be checked on a GSP representation and our 3-D packing algorithms (Section 3.5 and Section 3.6) will always satisfy the vertical constraints in the presence of these feasibility conditions. We call them *GSP feasibility conditions*, and they can be used in two ways: a) to detect and prune an infeasible GSP; or b) to guide stochastic moves to generate feasible GSPs only. Thus, the GSP feasibility conditions can significantly reduce the search space and therefore speed up the search process. In this chapter, we discuss vertical relations by using a simple example of specific vertical constraints that only require that the xy-coordinates of the modules under vertical constraint are the same, i.e. the modules are vertically aligned.

3.3 TWO LAYER FEASIBILITY CONDITION

In this section, we present the feasibility conditions on a GSP that only contains two device layers. The two layer feasibility conditions are easy to explain. The two layer feasibility condition will be later extended to multiple device layers in this chapter. As it was discussed in Sections 3.1 and 3.2, each sequence pair in a GSP is independent of the other sequence pairs, and therefore the inter-layer placement information between two or more sequence pairs of a GSP is absent. Thus there is no straightforward method to detect the feasible set

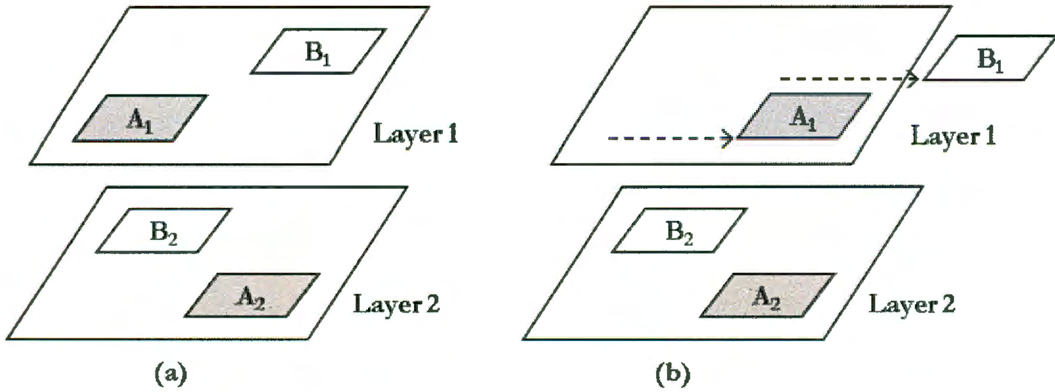


Figure 3.5: (a) Module pairs {A_1, A_2} and {B_1, B_2} under vertical constraints in two device layers are shown. (b) If A_1 is moved rightward as shown by the arrow to align with A_2, B_1 moves further away from B_2. Both pairs cannot be aligned simultaneously, and thus are infeasible.

of GSPs. In this work, we study the relative order of modules in different sequence pairs of a GSP, and identify a set of common patterns that is present in each feasible GSP. We will elaborate our work using the following example:

Let us consider an example of four modules $\mathbf{A_1}$, $\mathbf{A_2}$, $\mathbf{B_1}$ and $\mathbf{B_2}$ in two device layers. The modules to be aligned vertically are labeled with the same letter and distinguished with the subscripts that indicate their device layers. Thus $\mathbf{A_1}$ and $\mathbf{A_2}$ need to be aligned vertically. Similarly $\mathbf{B_1}$ and $\mathbf{B_2}$ must be aligned vertically. Let us consider the example of an infeasible GSP that is composed of the following sequence pairs representing two device layers:

Layer 1: {<……A_1…….B_1….>;   <……A_1……..B_1……..>}

Layer 2: {<……B_2, …….A_2…>;   <… B_2……A_2……...>}

According to these sequence pairs, A_1 has to be to the *left* of B_1 in layer 1 while A_2 has to be to the *right* of B_2 in layer 2. Please notice that A_1 and B_1 are constrained in layer 1 along the +X axis only because they have to satisfy and preserve their relative positions defined by the sequence pair of layer 1. They are, however, allowed to have different Y coordinates as shown in Figure 3.5(a). Similarly, A_2 and B_2 are restricted in the X-direction only. To satisfy the vertical constraint, the $\{A_1, A_2\}$ and $\{B_1, B_2\}$ module pairs should align simultaneously. It can be observed from Figure 3.5(b) that if A_1 moves rightward to align with A_2, B_1 moves further away to the right as well in order to preserve the sequence pair. Thus it results in B_1 moving further away from B_2. Similar observation can be made if B_2 is moved rightward to align with B_1. It is clear that when aligning one pair, the modules in the other pair get even more separated. Therefore these two module pairs can never overlap simultaneously which implies that the vertical constraints imposed on these two module pairs can never be satisfied for the given GSP. In the next sub-section we will define feasibility conditions for these two pairs based on their relative orders in each of the sequences of the GSP.

3.4 GRAPH REPRESENTATIONS OF FEASIBILITY CONDITIONS

In order to better visualize and represent the groups of feasible and infeasible solutions we define a graph representation for the feasibility conditions. Let us assume that $\{\Gamma_1^+; \Gamma_1^-\}$ and $\{\Gamma_2^+; \Gamma_2^-\}$ represent the sequence pairs of layer 1 and layer 2 respectively (see Figure 3.6(a)). For each sequence of the original GSP, we first construct a *constrained sequence.* We scan a sequence from left to right, detect constrained modules, and record them in the *constrained sequence* in the order of their detection. Using this procedure, we obtain $\{\psi_1^+, \psi_1^-\}$

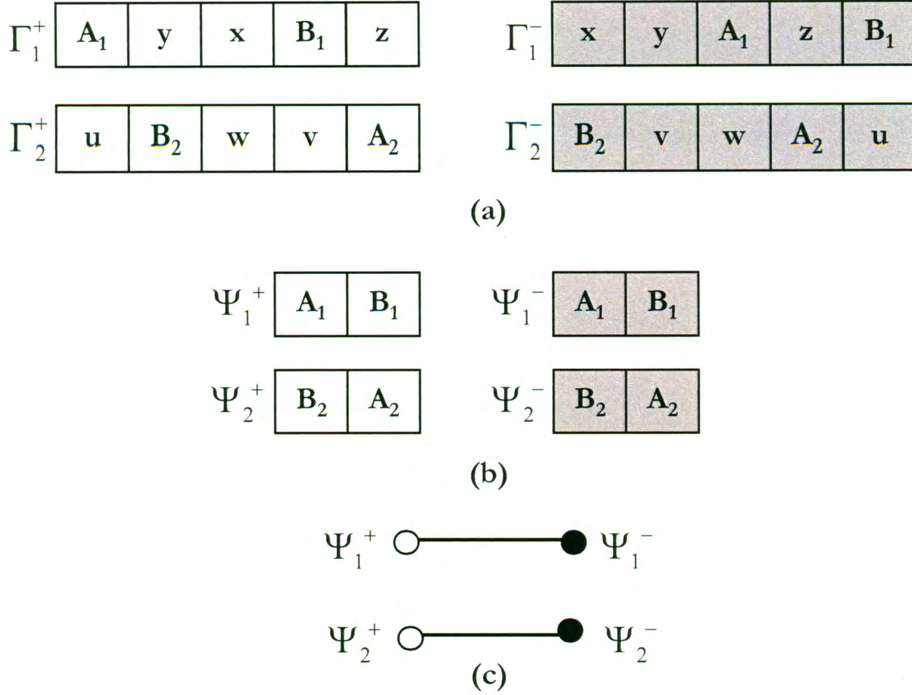
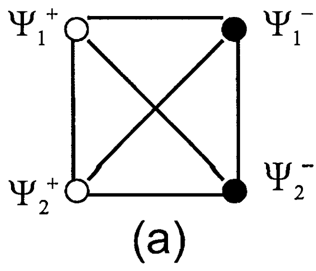
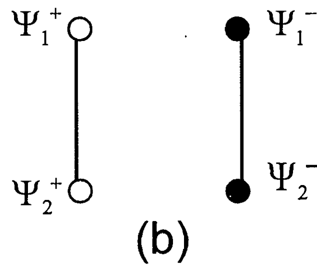


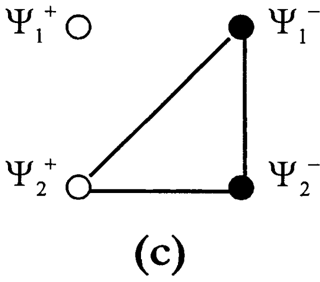
Figure 3.6: Construction of a feasibility condition graph: (a) Given sequence pairs $\{\Gamma_1^+ ; \Gamma_1^-\}$ of layer 1 and $\{\Gamma_2^+ ; \Gamma_2^-\}$ of layer 2. Module pairs $\{A_1, A_2\}$ and $\{B_1, B_2\}$ are vertically constrained. (b) Constrained sequence pairs (c) The resultant feasibility condition graph.

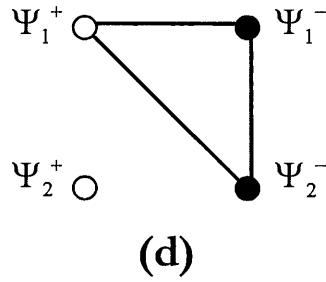
and $\{\psi_2^+, \psi_2^-\}$ as two sets of *constrained sequence pairs* as shown in Figure 3.6(b). Now we define a graph G(V, E) where vertices represent constrained sequences. Positive constrained sequences (ψ_1^+, ψ_2^+) are white nodes while negative constrained sequences (ψ_1^-, ψ_2^-) are black (see Figure 3.6(c)). We include an edge between two vertices if the orders of constrained modules in two *constrained sequences* represented by these vertices are the same. Thus we obtain a feasibility condition graph as shown in Figure 3.6(c).

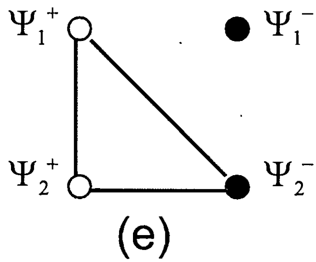
All the feasible conditions are shown in Figure 3.7. Feasibility of these cases was examined using the method presented in Section 3.3 (Please see Figure 3.5) and found that all module pairs can be vertically aligned simultaneously. An alternative method to theoretically











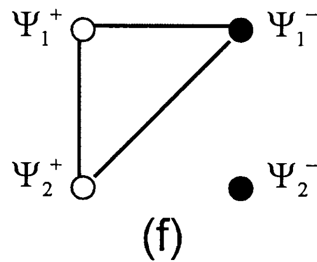
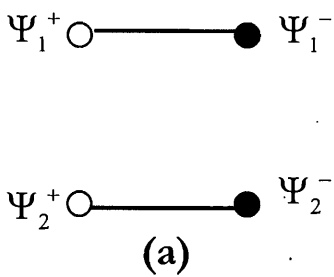


Figure 3.7: Graph representation showing six out of eight cases which are feasible conditions for two module pairs in two device layers.



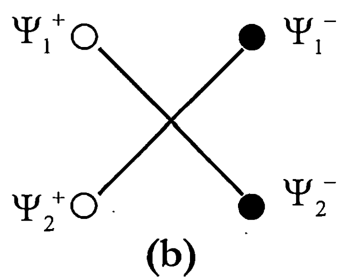


Figure 3.8: Graph representation of infeasible sequences pair constraints for two module pairs in two device layers.

examine the feasibility conditions using the xy-coordinates of the lower left corners of modules has been given in Appendix A. The infeasible cases are presented in Figure 3.8 in which all module pairs cannot be vertically aligned simultaneously. These cases can be identified on GSP and eliminated immediately without any unnecessary computation. The unnecessary computation involves transformation of topological representation (i.e. GSP) to geometric module packing in each device layer. Since the floorplanning algorithms using stochastic search methods are iterative processes, the cost of unnecessary computation for module packing of infeasible solutions can add up and reduce the speed of the algorithm. The exact cost of this computation for each infeasible solution depends on the problem size (i.e. total number of modules in a floorplan) that will be discussed in Section 3.5. Please note that there are 25% infeasible cases (two out of eight configurations). A generic theorem for a feasibility condition for two module pairs in two layers is stated as theorem 1:

Theorem1:

Feasibility Condition for two pairs of modules: Given a two-layer feasibility condition graph **G**(V, E), there exists a feasible solution to the vertical constraint problem for two module pairs represented by the graph if:

(a) ***G*** *contains a clique of size K (K* $\in$ *{3,4}), or*

(b) ***G*** *contains two cliques of size 2, such that each clique contains only nodes of the same color.*

The formal proof of theorem 1 is given in Appendix A. During software implementation, theorem 1 can be implemented (satisfied) by checking if either $\{\psi_1^+, \psi_2^+\}$ or $\{\psi_1^-, \psi_2^-\}$ are connected by an edge, i.e. modules in those node pairs are in the same order. Since we compare the orders of modules (under vertical constraints), the feasibility conditions can be

expanded to more than two pairs of modules constrained in two device layers. For example, let us assume $L_1 = \{A_1, B_1, C_1\}$ and $L_2 = \{A_2, B_2, C_2\}$ are vertically constrained in two layers. There are three possible ways of selecting two modules from the three modules of L_1. Thus we can decompose L_1 in three combinations of two modules as $\{\{A_1, B_1\}, \{A_1, C_1\}, \{B_1, C_1\}\}$ such that modules in each pair maintain their relative order defined in L_1. Similarly $\{\{A_2, B_2\}, \{A_2, C_2\}, \{B_2, C_2\}\}$ will be the decomposed pairs of L_2. From this decomposition, we will have three feasibility configurations of two module pairs as $[\{A_1, B_1\} \cup \{A_2, B_2\}]$, $[\{A_1, C_1\} \cup \{A_2, C_2\}]$, and $[\{B_1, C_1\} \cup \{B_2, C_2\}]$. Please note that all three decomposed configurations will map to Figure 3.7(a), i.e. they will satisfy theorem 1. Therefore $\{A_1, B_1, C_1\}$ and $\{A_2, B_2, C_2\}$ are feasible i.e. all module pairs (A_1, A_2), (B_1, B_2) and (C_1, C_2) under vertical constraints can be aligned simultaneously. A theorem for a two layer feasibility condition is stated as theorem 2:

Theorem 2:

Two Layer Feasibility Condition: Let $L_1 = \{A_1, B_1, C_1, D_1, \ldots\}$ and $L_2 = \{A_2, B_2, C_2, D_2, \ldots\}$ are two sets of modules located in two different device layers L_1 and L_2 respectively. The packing on L_1 and L_2 are represented by Constrained Sequence Pairs SP_1 and SP_2 respectively. SP_1 and SP_2 are feasible if module pairs $\{(A_1, A_2), (B_1, B_2), (C_1, C_2), (D_1, D_2), \ldots\}$ can be vertically aligned simultaneously. The vertical alignment of all these module pairs is feasible if:

Every combination of two module pairs decomposed from L_1 and L_2 (without changing their relative orders) construct a feasible configuration by satisfying theorem 1 i.e.

$\{u_1, u_2\}$ and $\{v_1, v_2\}$ form **a feasible configuration** $\forall \{u_1, v_1\} \in SP_1$; $\forall \{u_2, v_2\} \in SP_2$.

The formal proof of theorem 2 is given in Appendix A. The *two layer feasibility condition* theorem can be easily extended to include modules vertically constrained in more than two layers. Consider an L layer 3-D floorplan with vertical constraints. If the modules are constrained in more than two layers, each combination of 2-device layers can be considered and its feasibility can be detected based on theorem 2. If every such combination satisfies theorem 2, then a solution to the vertical constraint will be feasible. The multi-layer feasibility condition is stated as theorem 3:

Theorem3:

Multi Layer Feasibility Condition: Given a set of multi-layer constrained sequence pairs, there exists a feasible solution to the vertical constraint problem if:

Each combination of 2-device layers satisfies the 2-layer feasibility condition theorem (i.e. theorem 2).

The formal proof of theorem 3 is given in Appendix A.

3.5 3DCG: A MODULE PACKING ALGORITHM WITH VERTICAL CONSTRAINTS

A module packing algorithm decodes the topological representation (i.e. data structure) of a floorplan to a geometric floorplan. However, it does not search for an optimal solution. A floorplanning algorithm searches for an optimal solution, and uses the module packing algorithm as a decoder of a floorplan representation to a real floorplan. Thus, module packing is just a small part of the overall floorplanning algorithm as seen in Figure 2.3. A good module packing algorithm should be able to quickly translate a topological representation into a geometric floorplan satisfying the vertical constraints. In a quasi 3-D

representation, module packing of each layer is generally performed independently because each device layer has its own independent topological representation. Thus the packing of each device layer does not have information about the module packing of the remaining device layers in a 3-D floorplan. In case of vertical alignment of modules located in different device layers, their vertical alignment information has to be shared across multiple device layers for the packing algorithm to satisfy the vertical constraints. We present a graph-based algorithm for module packing that satisfies vertical constraints among modules in different device layers in a 3-D floorplan. This algorithm was inherited from Benyi Wang, a former Ph.D. student of my advisor, Prof. Chrzanowska-Jeske. The algorithm was a part of the basic 3-D floorplanning algorithm [101] discussed in section 2.7.

In this method, we create two global 3-D constraint graphs (*3-D-X* and *3-D-Y*) to evaluate the 3-D module packing. The global constraint graphs combine the geometric floorplan of all device layers and help us define the vertical constraints of modules that are located in different device layers. We will only elaborate the method to create a *3-D-X* constraint graph in this section because the same method is valid for *3-D-Y* as well. Let us assume that there are only two device layers in the 3-D floorplan. Layer 1 has four modules {1,2,3,4} and layer 2 contains modules {5,6,7,8}. Across these two layers, module 2 has to be vertically aligned with module 7. We use the following steps to create the global *3-D-X* constraint graph:

1. Create the 2-D constraint graph G_h along the X-axis [53] for each device layer from the sequence pair. Thus we obtain two such constraint graphs corresponding to two device layers as shown in Figure 3.9(a) and Figure 3.9(b). Each directed edge of

the graph is assigned a weight (not shown in Figure 3.9) equal to the width of a module connected at the tail of the edge. Edges connected to the source and sink nodes have zero weights.

2. We merge the constraint graphs of both layers by adding a *global source* node in the 3-D-X graph and connecting it with the source nodes of both device layers (src1 and src2) using two zero weight edges (see Figure 3.9(c)). Similarly we add a *global sink* node and connect the sinks of each layer with zero weight edges.

3. If node 2 and node 7 have to be vertically aligned, then we insert two edges of zero weight between node2 and node7 in cyclic (i.e. 2⟶7 and 7⟶2) fashion to represent the vertical alignment as shown in Figure 3.9(c). Please note that in the case of partial vertical alignment, where two modules need to be partly overlapped, these two cyclic edges can be assigned non-zero weights.

Similarly, we create a *3-D-Y* constraint graph. The Bellman-Ford's shortest path algorithm is used to find the critical path (by changing the signs of edge-weights) on *3-D-X* and *3-D-Y* in order to calculate the module packing along with the chip size. The Bellman-Ford algorithm also detects a negative cycle that indicates no solution. Our packing algorithm is similar to a 1-D compaction algorithm [68] and it has $O(n^3)$ runtime complexity [68].

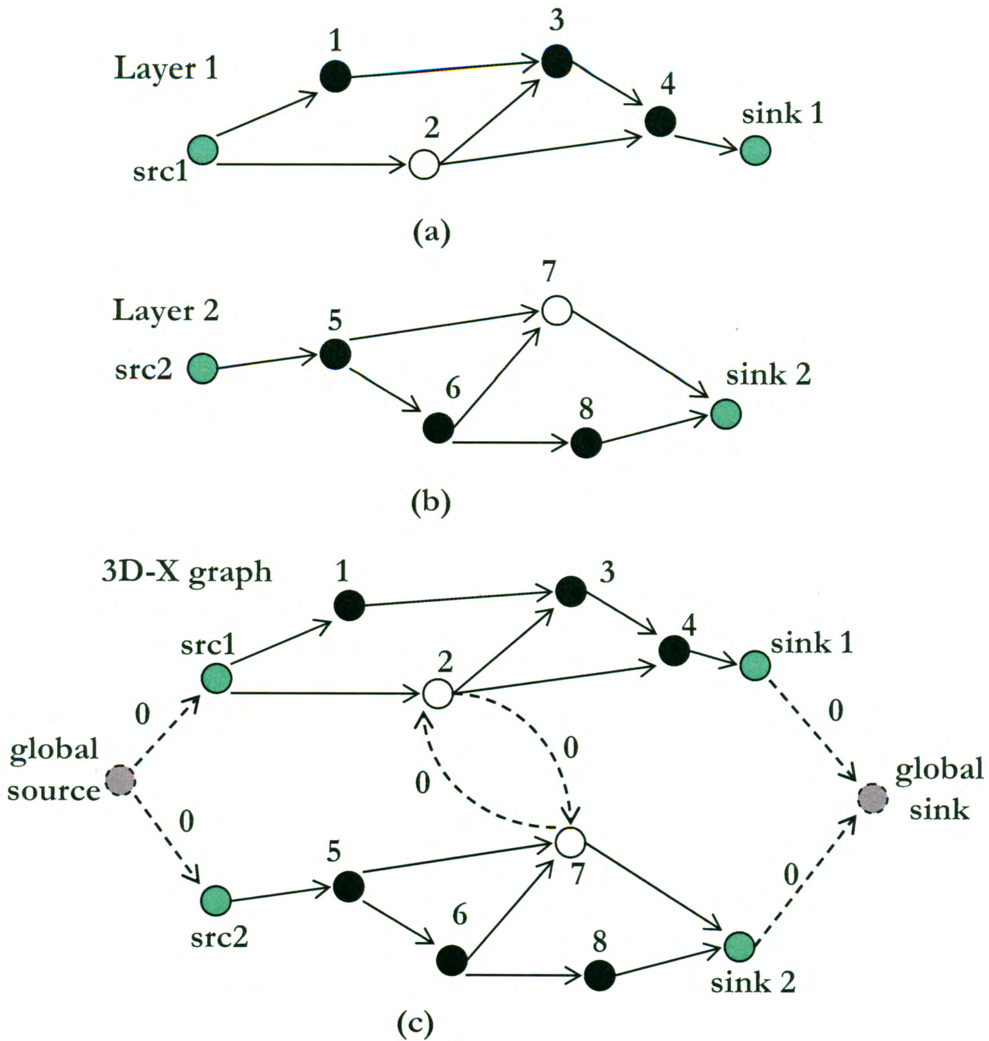


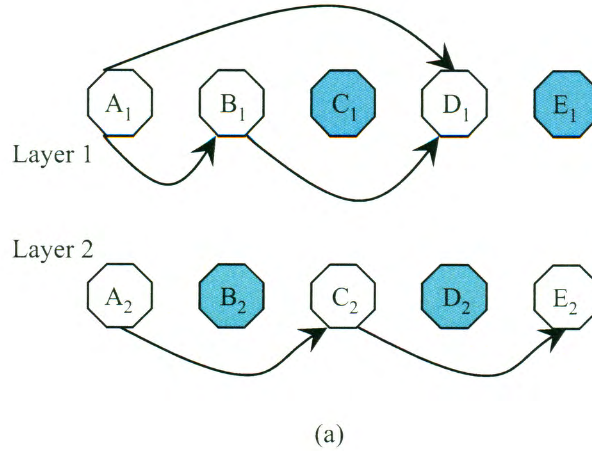
Figure 3.9: 3-D-X constraint graph creation. (a) 2-D constraint graph along +X axis in Layer 1 (b) 2-D constraint graph along +X axis in Layer 2 (c) Merging the two 2-D constraint graphs using a global source and a global sink node. To vertically align node2 and node7, two edges are inserted between node2 and node7 in the cyclic fashion (i.e. 2→7 and 7→2). All newly introduced edges are dotted and have zero weights.

3.6 LCSLS: A FAST MODULE PACKING ALGORITHM WITH VERTICAL CONSTRAINTS

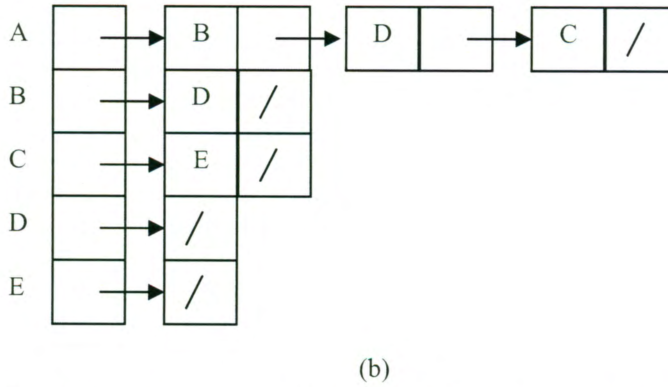
The module packing algorithm using 3-D constraint graphs (3DCG) discussed in the previous section is slow due to $O(n^3)$ runtime complexity. In addition, it requires the construction of graphs from the grouped sequence pair (GSP) which takes further computational effort. In this section, we present an alternative packing algorithm using Longest Common Subsequence (LCS), and Lateral Shifting is presented as a part of this research that does not require the creation of large constraint graphs like the 3DCG algorithm. We call it the LCSLS algorithm.

The LCSLS is a fast algorithm for module packing with vertical constraints. We discuss the strategy for packing with alignment of constrained modules along the X-axis because the same strategy is valid for alignment along the Y-axis. As the name of our algorithm says, we compute packing of each device layer by LCS [58] and then laterally shift the modules under vertical constraint to align them vertically. However, the order of shifting a module is very important for minimum packing and minimum number of lateral shifting. We achieve that by performing a topological sort on a constrained adjacency list *Adj_X*. The construction steps of *Adj_X* are as follows:

Let $\{A_1, B_1, C_1, D_1, E_1\}$, and $\{A_2, B_2, C_2, D_2, E_2\}$ be the modules in Layer 1 and Layer 2 as shown in Figure 3.10(a). An edge between $A_1 \rightarrow B_1$ in Figure 3.10(a) shows that A_1 is to the *left* of B_1 which is obtained from GSP. A small subgraph is created in each device layer by introducing edges between modules. Let $A = \{A_1, A_2\}$, $B = \{B_1, B_2\}$, ...,$E = \{E_1, E_2\}$



(a)



(b)

Figure 3.10: Creation of a constrained adjacency list *Adj_X* for 2-device layers. (a) A = {A_1, A_2}, B = {B_1, B_2}, ..., E = {E_1, E_2} are vertically constrained in two device layers. (b) Adj_X obtained by merging the graphs of two layers and denoting the merged nodes by their group name.

are grouped such that all elements inside a group are under vertical constraints, i.e. they need to be vertically aligned. We merge these sub-graphs by replacing each sub-module by its group name. For example, $\mathbf{A_1}$ and $\mathbf{A_2}$ are denoted by **A** only. Finally, we construct the constrained adjacency list (*Adj_X*) as shown in Figure 3.10(b). By performing the topological sorting on *Adj_X*, we obtain the order of groups such that all elements of a group are aligned first before moving to the next group. Since topological sorting works on

a directed acyclic graph (DAG), we check for the presence of cycles while sorting, and a cycle indicates no solution. The main steps of the LCSLS algorithm are as follows:

1. Topologically sort the *Adj_X* and obtain a list $\boldsymbol{L}$ containing the sorted sub-module *groups*. If cycles are detected during topological sorting, exit/terminate because there will be no solution. (Please note that, similar to Bellman-Ford algorithm used to find packing in 3DCG algorithm, LCSLS also detects a cycle for infeasible packing).

2. Calculate module packing along the X-axis in each device layer using the Longest Common Subsequence (LCS) [58] method in $O(n \lg \lg n)$ runtime. This will give the x-coordinates of each module *without* vertical alignment and x-dimensions of all layers of the chip.

3. Extract a group $\boldsymbol{g}$ from the topologically sorted list $\boldsymbol{L}$.

4. Shift the x-coordinate of a module $\boldsymbol{m_i}$ (in layer i), $\boldsymbol{m_i} \in \boldsymbol{g}$ along the +X-axis to align it with the *rightmost* module of $\boldsymbol{g}$. For example, let us assume that $\boldsymbol{g} = \{A_1, A_2, A_3, A_4\}$ and all elements of $\boldsymbol{g}$ are located in four different device layers. If A_4 has the largest x-coordinate then all other members of $\boldsymbol{g}$ should be aligned with A_4 only.

5. Shift all modules which are to the right of $\boldsymbol{m_i}$ in layer i by the minimum distance such that there is no overlap caused due to the shifting of $\boldsymbol{m_i}$.

6. Repeat steps 4 to 6 for all $\boldsymbol{m_i} \in \boldsymbol{g}$.

7. Go to step 3 and repeat the process for all $\boldsymbol{g} \in \boldsymbol{L}$.

Now we perform a complexity analysis of the proposed LCSLS algorithm. Let s be the number of vertically constrained groups, and n be the total number of modules such that $s < n$. The topological sort of step 1 will take $O(s + e) \sim O(s^2)$, where e is the total number of edges. Step 2 takes $O(n \lg \lg n)$ time. Step 3 will executes in $O(1)$ time. Steps 4 – 5 take $O(n)$ and step 6 takes constant time because the size of g can at most be equal to the total number of device layers which is very small and fixed. Thus, steps 4 – 6 take $O(n)$ time and Step 7 repeats them s times in a loop. Therefore, the runtime from steps 3 – 7 will be $O(s.n)$. Finally, the total runtime of LCSLS (step 1 to step 7) will be $O(s^2 + n \lg \lg n + s.n)$. However, in most practical cases, the total number of vertically constrained groups will be much smaller than the total number of modules, i.e. $s << n$. In those cases, the runtime will be reduced to $O(n \lg \lg n)$ only. In the worst case scenario $s \approx n$, the runtime will be $O(n^2)$ which is still faster than the 3DCG packing algorithm The LCSLC algorithm can also be used for partial vertical alignment by controlling the spatial distance during lateral shifting.

Both 3DCG and LCSLS algorithms produce identical floorplans for a given GSP and a set of vertical constraints. The quality of an optimized floorplan solution is determined by the design of a floorplanning algorithm instead of a module packing algorithm. Module packing algorithms only determine how fast they can decode a topological floorplan representation to a geometric floorplan. Thus module packing algorithms only affect the runtime of a floorplanning algorithm. From the time complexity analysis of 3DCG and LCSLS algorithms, it appears that 3DCG has $O(n^3)$ complexity, whereas LCSLS has average case $O(n \lg \lg n)$ and worst case $O(n^2)$ complexity. Thus LCSLS is asymptotically faster than the 3DCG algorithm which will be experimentally verified in Chapter 4 (please see Section 4.9).

4.1 MOTIVATION

As discussed in Chapter 2 (Section 2.6), the existing 3-D floorplanning algorithms [13],[45],[64],[65] assume that an entire module is placed on one device layer. These algorithms do not consider the possibility of distributing cells of a module in multiple device layers to reduce its wirelength. In other words, floorplanning solutions might benefit from 3-D placements of cells within some modules. If a 3-D floorplan is completed without considering 3-D placement inside modules, it is too late to perform 3-D cell placement because a particular module has been assigned to a single device layer. This idea came from the unpublished work of Benyi Wang and Chrzanowska-Jeske [101] which provided a groundwork for this research. However, the advantages of 3-D placement have been reported by many researchers. For example, a min-cut partitioning based 3-D placement [3], used on ISPD'98 placement benchmark circuits, indicates a 28% – 51% reduction in total wirelength [14] (for 2 to 5 layer 3-D placement) compared to the total wirelength of a 2-D implementation of the same benchmark. Furthermore, in 2008, Ma et al. [94] presented a case study of the design driver of a superscalar microprocessor. Ma et al. first implemented the design driver as a 3-D chip consisting of only 2-D micro-architectural blocks, performed 2-D placement & routing within each block, and recorded the chips performance, temperature and power. In the next step, Ma et al. implemented the micro-architectural blocks of the design driver of the superscalar microprocessor as 3-D modules. For each block, they performed different 3-D block implementation by varying the number

of device layers in each implementation. Thus for a particular block, they performed one-layer, two-layer, three-layer, and four-layer 3-D module implementations. Next they performed thermally driven 3-D placement of logic gates inside all 3-D modules, and characterized these modules in terms of temperature, power, and speed. They developed a 3-D box packing tool, and treated each module as a 3-D box to perform floorplanning using simulated annealing based optimization to select 3-D modules (based on each module's characterized values in terms of speed, power and temperature) to optimize the speed, temperature and power of the new 3-D implementation of the design driver of the superscalar microprocessor. Please note that this case study used a block selection based 3-D floorplanner and requires multiple 3-D implementations of the same block (by varying the number of device layers for each block), which might not be feasible for a real design containing large number of modules because it will increase the design cost and the design time.

Finally Ma et al. compared *a)* the *initial* 3-D implementation in which all modules were implemented as 2-D modules, with *b)* the *new* 3-D implementation in which modules were 3-D modules and each module had a different number of device layers. During the comparison, Ma et al. reported that multi-layer (3-D) micro-architectural blocks (i.e. case *b*) can improve performance by 14%, reduce power by 10% – 30%, and decrease temperature by 11% compared to single-layer (2-D) blocks (i.e. case *a*) in 3-D microprocessors. Thus 3-D placement of cells inside modules can further reduce total wirelength, power and temperature resulting in improved performance of 3-D ICs.

Emerging advanced 3-D technologies with physical vias under one micrometer diameter have been reported [10],[69] and via density has been projected (by the researchers of IBM) [10] to reach up to 100,000/cm^2 approximately by the year 2010 as shown in Figure 1.6. It will be a ten-fold increase in via density since the year 2000. Due to the increasing trend in via density, the total number of vias should not be a strong limiting factor. Furthermore, year 2009 study [69] of 4-layer 3-D ICs shows that the area overhead due to vias in large designs (200M gates) is 4% of the total area, whereas it is 10% in small designs (5M gates). Therefore assuming the *increased via density* and *negligible area overhead* of vias in large designs, a novel 3-D floorplanning approach of distributing cells of a module across consecutive device layers might be beneficial (as discussed in the previous paragraph) for further wirelength reduction in future 3-D ICs.

4.2 PLACEMENT-AWARE CONSTRAINTS

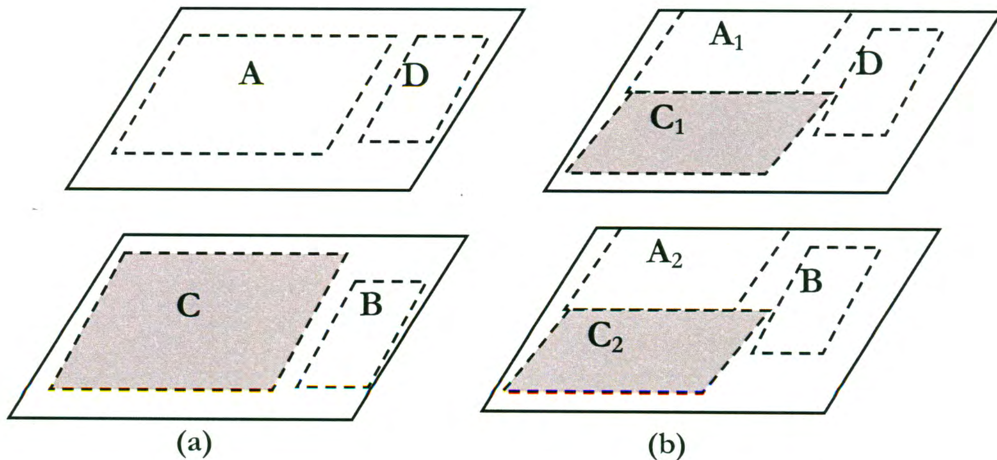


Figure 4.1: (a) A 3-D floorplan of 2-D modules (b) a new 3-D floorplan with sub-module pairs {A_1, A_2} and {C_1, C_2} transformed as 3-D modules after module splitting and imposing vertical constraints.

In placement-aware 3-D floorplanning, a module is divided into multiple sub-modules which are assigned to different device layers, i.e. a 2-D module is transformed into a 3-D module as shown in Figure 4.1. The decision to convert a 2-D module to a 3-D module should consider a few important objectives: *a)* the 3-D module should be able to use the 3-D placement tools because 3-D placement of logic cells will be required; *b)* 3-D modules should be able to attain a minimum intra-module wirelength because it will reduce the system level total wirelength; *c)* 3-D modules should have minimum intra-module TSV heights because it will affect the signal delay; and *d)* the effect of module splitting can be evaluated quickly. To satisfy these important objectives, we assume that: *a)* each sub-module has the same dimensions; *b)* sub-modules are placed in consecutive device layers (Figure 4.2); *c)* sub-modules have the same planar location (Figure 4.3); and *d)* a device layer contains only one sub-module of a split module. Collectively, we call all these conditions *placement-aware constraints*. The same planar location condition for identical sub-modules makes them vertically aligned and is reduced to *vertical constraint* (Chapter 3) that guarantees minimal intrinsic wirelength of the 3-D module as shown in Figure 4.3.

The selfsame dimension and equal planar location of sub-modules permits designers to use existing 3-D placement tools [3],[14],[70] inside 3-D modules. According to technology and design requirements, these constraints can be relaxed, i.e. a module can also be split into non-identical sub-modules, and sub-modules may not have the same planar location. However, to be able to use the existing 3-D placement tools inside 3-D modules, minimize the intra-module wiring, and evaluate the wiring cost we only split a module into identical sub-modules, and keep their planar locations identical. If the modules would be allowed be

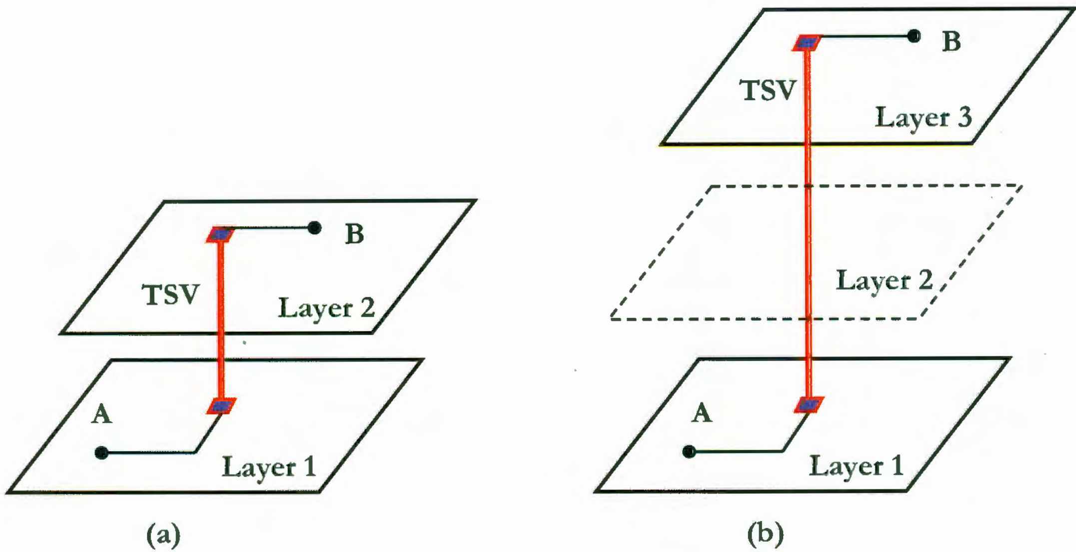


Figure 4.2: Sub-module pair placed in (a) consecutive (b) alternate device layers. Sub-modules in the consecutive device layers minimize the TSV height.

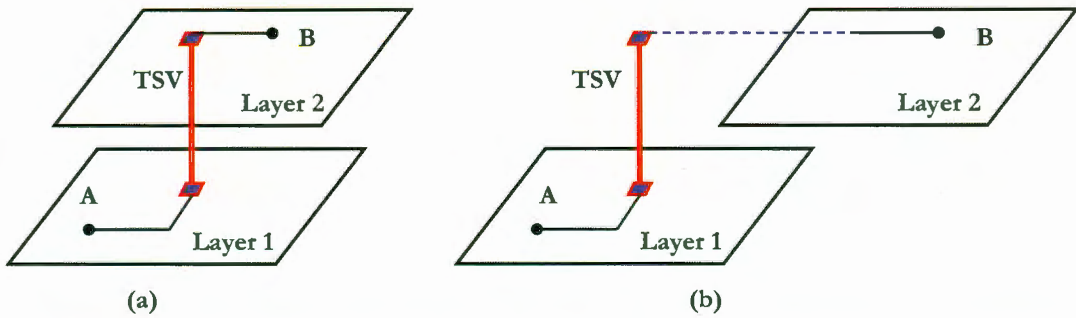


Figure 4.3: Sub-module pair with (a) the same planar location (b) different planar locations. Vertical constraint with the same planar location of sub-modules produces smaller intra-module wirelength.

split into non-identical rectangles, a new model to estimate the wiring cost of splitting will be required. The evaluation of wiring cost and the need for a wirelength estimation model will be discussed in Section 4.5. In addition, new 3-D placement and 3-D routing tools will need to be developed.

4.3 PROBLEM FORMULATION

Consider a set of *n* rectangular modules where each module M_i has a fixed area A_i, width W_i and height H_i connected by *m* nets. Let (k_i, p_i) be the Rent's parameter of M_i and L be the total number of fixed active layers. Let (x_i, y_i, z_i) denote the lower left corner of the module M_i, where $1 \leq z_i \leq L$ and z_i belongs to the set of natural numbers. A module can be split into S sub-modules. A placement-aware 3-D floorplan is an assignment of (x_i, y_i, z_i) for each M_i and each sub-module M_{ij} of a split module M_i such that no two modules or sub-modules on the same device layer overlap, and all sub-modules M_{ij} of the same module M_i satisfy the placement-aware constraints. We seek a solution to the following problem:

Placement-aware 3-D Floorplanning with Vertical Constraints Problem (3-D FVC): *Given a technology node, a set of n rectangular modules, with areas, aspect ratios and known Rent's parameters, connected by m nets in L device layers, and a set of placement-aware constraints, find a 3-D floorplan that satisfies all the placement-aware constraints and minimizes chip area, inter-module and intra-module wirelength while controlling the number of inter-module and intra-module vias.*

Please note that Rent's parameters (k,p) can be preprocessed from the optimized netlist (*after logic synthesis*) of modules as reported in [77]. A brief description of the Rent's parameter extraction is given in Appendix C.

4.4 STOCHASTIC COMBINATORIAL OPTIMIZATION

We use grouped sequence pair (GSP) [102] representation and an evolutionary algorithm (EA) [101] in our 3-D floorplanning algorithm. Unlike simulated annealing (SA), an evolutionary algorithm (EA) processes a population of potential solutions in parallel (a

detailed description of SA and EA was presented in Section 2.3). Each individual in the population is a unique solution. Parents are subjected to stochastic "reproduction" that produces offspring. All parents produce one or more offspring at each generation; parents and offspring are ranked according to fitness and best fit individuals become the next generation population. This subset of an evolutionary algorithm is known as *evolution strategy* (ES), in which the selection process is purely deterministic. In *evolution strategy* mutation is used as the main operator [103]. Recombination (i.e. crossover) with small probability can be used along with mutation to escape out of local minima. However, it is not a "must have" condition in evolution strategy. A simple crossover can be achieved by mixing the sequences of sequence pairs from two parents. Such crossover will create a very large perturbation in the solution search space. Furthermore, in the presence of vertical constraints, the crossover operator will most likely produce an infeasible solution. Therefore recombination is not included and we use *mutation* operators only. At the beginning, we randomly construct a different *Grouped Sequence Pair* (GSP) for each individual solution. Although many CAD algorithms use simulated annealing, we chose the evolutionary algorithm because we wanted to explore parallel searching of the solution space. The conclusion will be presented in Section 4.9 while comparing the results of our EA based algorithm with the results of other SA based algorithms.

4.5 COST ESTIMATION OF WIRELENGTH REDUCTION DUE TO MODULE SPLITTING INSIDE 3-D MODULES

Floorplanning is usually performed when no module layouts are given. Therefore it is unknown what the total wirelength is within a 2-D module and how will it change when the

module is partitioned into multiple sub-modules. Statistical wirelength prediction methods become very useful in this scenario. Several wirelength distribution models are widely available in the literature for system level interconnect prediction of square-shaped 2-D and 3-D chips [1],[71],[72],[73]. Some of these models have been further extended to handle rectangular shapes [32]. These models show significant reduction in wirelength of 3-D chips compared to 2-D ICs. The models are based on Rent's Rule [74]:

$$T = kN^p \tag{4.1}$$

where T is the number of I/O terminals of the chip/module, N is the total number of gates, k is Rent's coefficient and p is Rent's exponent. Additionally, the 3-D wirelength distribution models assume that all device layers have the same area and aspect ratio, and are placed exactly at the same **x-y** coordinates. To be able to use these 3-D wirelength models we split a module into sub-modules of the same size only. Please note that in case of non-identical sub-modules, a new mathematical/statistical model for wirelength estimation will need to be developed. Furthermore, it is desired to have an analytical model for the mathematical model to quickly estimate intra-module wirelength during floorplanning. Due to a non-trivial mathematical model, there is no direct extension from the previous work. It will require further research in this direction.

We assume that *a)* Rent's parameters (k, p) and the average fanout of logic gates inside each module are given, *b)* a rectangular module contains only 2-input NAND gate based circuitry, and *c)* gates are organized into a homogeneous array inside the module. Based on the area of a 2-input NAND gate cell, we estimate the total number of gates N.

Let us assume that we start with a rectangular 2-D module containing N gates and T input/output terminals which is split into two identical sub-modules. To find the wirelength reduction due to such partition, we estimate total wirelength inside the initial 2-D module and the final 3-D module using our rectangular 2-D and 3-D wirelength distribution models [32]. The models in [32] have been generalized by incorporating the aspect ratios (a_r) of the modules as input to the models. Our models also converge with models from [1],[71],[72],[73] for a_r=1 which is a special case of square shape. A brief derivation of the rectangular wirelength distribution models and our analytical solutions is given in Appendix B. We have also derived the closed form analytical solutions for total wirelength estimation inside 2-D/3-D modules and therefore we are able to estimate the wirelength reduction in constant time. Experimental results related to our rectangular wirelength distribution models have been shown in Appendix B.

We have observed from our experiments that when 2-D modules are split, the reduction in wirelength inside the 3-D module varies between 28 to 50% for 2 to 5 device layers, which is in close agreement with the 3-D placement result reported in [3]. The authors of [3] use ISPD'98 placement benchmark circuits and perform placement of logic cells in 2-D/3-D designs and report percentage reduction in wirelength in 3-D chips with respect to 2-D designs. Das et al. [75] have reported similar reduction in wirelength (see Figure 4.4) of 3-D designs with respect to 2-D designs which matches closely with the actual wirelength obtained after placement and routing of ISPD'98 placement benchmark circuits. This verifies the accuracy of our estimation. In addition, we estimated the total wirelength of 18 IFU control logic circuitry (2-D modules) of an IBM POWER4 dual core microprocessor

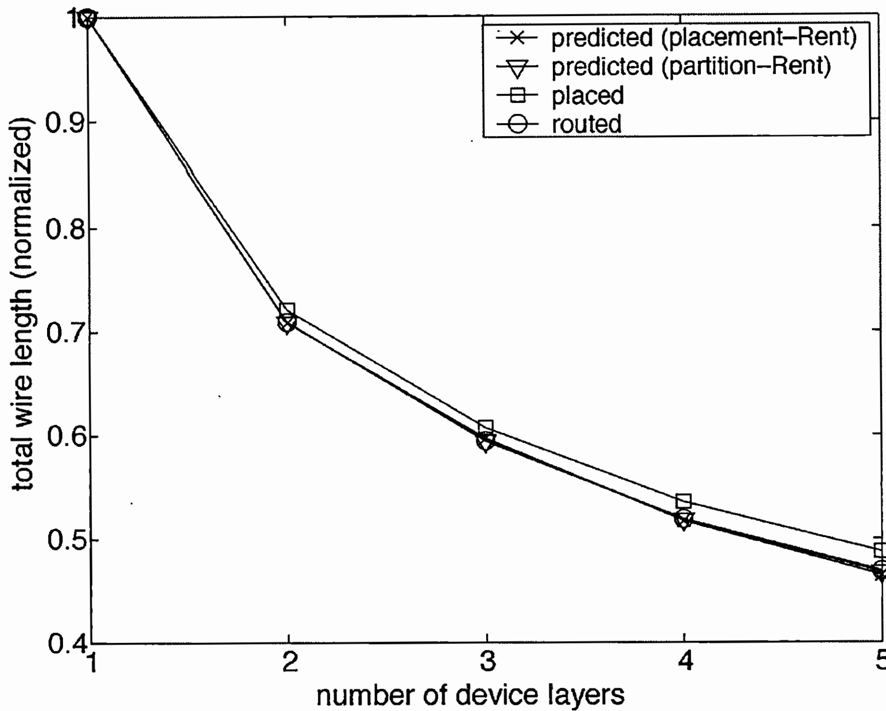


Figure 4.4: Predicted vs. placed and routed wirelengths of ISPD'98 benchmark circuits. Wirelength of 3-D placement and routing is normalized w.r.t. 2-D design to show the percentage reduction due to 3-D design [75].

TABLE 4.1: COMPARISON OF TOTAL MEASURED WIRELENGTH (226827.0 GATE PITCHES) OF 18 IFU CIRCUITS OF IBM POWER4 WITH THEIR ESTIMATED WIRELENGTH USING DIFFERENT WIRELENGTH DISTRIBUTION MODELS.

Wirelength Estimation Methods	Estimated Wirelength (gate pitch)	Error (%)
Our (k, p)	212684.0	-6.24
Our (k*, p*)	225139.6	-0.74
Davis (k, p) [76]	141256.9	-38.0
Christie (k*, p*) [76]	148317.0	-35.0
Donath (R_{p*}) [76]	206593.7	-9.0

and compared them with the actual (measured) wirelength obtained from [76]. These IFU modules have different rectangular shapes spread over a wide spectrum of aspect ratios. A cumulative estimation of the wirelength is shown in Table 4.1. The Rent's parameter (*k*, *p*)

and (*k**, *p**) were directly taken from the tables reported in [76],[77].The top two rows show our total wirelength estimations which are closest to the measured total wirelength when compared with other existing methods reported in [76]. This again validates our estimation.

4.6 PERTURBATION OF SOLUTION SPACE

In every generation, each floorplan is perturbed by randomly selecting one move from a set of pre-defined moves. The initial software inherited from [101] consisted *swap*, *invert*, *rotate*, *exchange*, *change group*, and *module-split* moves. However, all the existing moves have been modified as a part of this dissertation such that the new algorithm searches within feasible search space. In addition, some new moves such as *insert*, *submodule merge*, and *change-feasibility configuration* have been introduced.

1) ***Swap:*** positions of two randomly chosen modules are exchanged; we perform this operation either between non-constrained modules or between constrained sub-modules. For example, let us assume that modules **b** and **e** have been chosen in the initial sequence <a, x_2, **b**, c, d, **e**, y_2> for swapping. The sub-modules x_2 and y_2 in the sequence are under vertical constraints (with their respective sub-modules in another device layer) but **b** and **e** are not under vertical constraints. Therefore swapping between **b** and **e** is allowed. After performing the swap operation, the final sequence will become <a, x_2, **e**, c, d, **b**, y_2>. When swapping occurs between constrained sub-modules (such as swapping between x_2, and y_2), the feasibility configuration is kept unchanged by the method that will be explained in the *Module-split* move.

2) ***Invert:*** the order of a sequence between two randomly chosen points is reversed. The invert operation is achieved by continuously swapping modules from these two points and moving inward. In addition, the orders of modules under vertical constraints are not swapped such that feasibility conditions are preserved. For example, let us assume that the sequence <a, $\boldsymbol{x_2}$, b, c, d, e> needs to be completely reversed and $\boldsymbol{x_2}$ is under vertical constraint. Then we first take "*a*" and "*e*" from two extreme ends and swap them. Next we chose "x_2" and "*d*", but *do not* swap because x_2 is under vertical constraint. After that, "*b*" and "*c*" are swapped. Thus the inverted sequence becomes <e, $\boldsymbol{x_2}$, c, b, d, a>.

3) ***Insert:*** randomly select *a non-constrained module* in a sequence and move it to another randomly selected location in that sequence. For example, let us assume that we are given a sequence <a, x, **b**, c, d, **e**, y> in which a non constrained module **b** has been randomly chosen to be moved from its original location to a location just before module **e** in the sequence. After the *insert* operation, the new sequence becomes <a, x, c, d, **b**, **e**, y>. Since this move operated on non constrained modules, it does not disturb the feasibility configuration.

4) ***Rotate:*** swap a module's width and height. We only rotate the modules which are not square shaped. This move is operated on the geometric information of a module (i.e. height and width), and therefore it does not disturb the feasibility configuration.

5) ***Exchange:*** positions of two randomly chosen *non-constrained modules* on two different device layers are exchanged. For example, let us assume that sequence pairs of a two

layer 3-D IC are SP1 and SP2 such that SP1 = {<a, b, c, d, e >, <a, b, c, d, e >}, and SP2 = {<u, v, w, x, y, z >, <z, y, x, w, v, u >}. If module **b** of SP1 is randomly chosen to be exchanged with a randomly chosen module **w** of SP2, then we simply exchange these modules in both sequences of the SP1 and SP2. The resultant sequence pairs become as SP1 = {<a, **w**, c, d, e >, <a, **w**, c, d, e >}, and SP2 = {<u, v, **b**, x, y, z >, <z, y, x, **b**, v, u >}. Please notice that modules **w** and **b** shown in the bold letters in SP1 and SP2 have been exchanged. Since *exchange* is performed on non-constrained modules, this operation does not disturb the feasibility configuration.

6) ***Change group:*** randomly select a *non-constrained module* from a device layer and move it to a randomly selected different device layer. This move on *non-constrained modules* does not disturb the feasibility configuration.

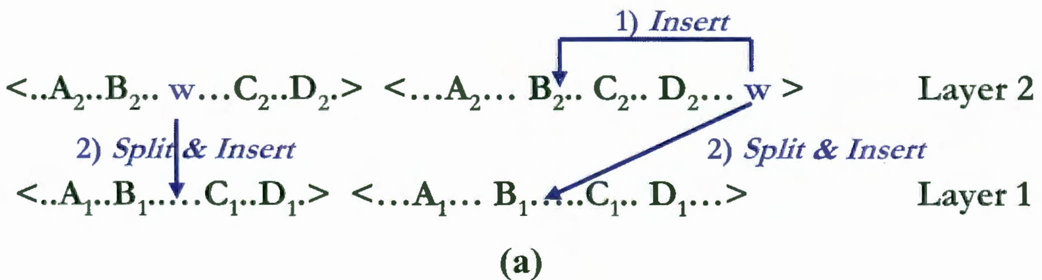
7) ***Module-split:*** split a randomly chosen module ***w*** into sub-modules; place them in consecutive device layers and preserve the feasibility conditions. As we can see from Figure 4.4, the single largest wiring reduction can be achieved at two device layers (approximately 30% reduction with respect to 2-D design). When a module is split into more than two layers, the incremental reduction in wirelength drastically reduces and the reduction saturates to four device layers. Thus we split modules into two sub-modules only. In addition, we restrict the feasibility configuration such that every combination of two sub-module pairs constructs the same type of feasibility configuration as it will be explained at the end of the move description.

8) ***Submodule-merge:*** merge all sub-modules of a previously split module and restore its original shape. The restored module is placed in a device layer that is randomly chosen as one of its sub-module's device layers.

9) ***Change-feasibility configuration:*** change the present feasibility configuration to a different feasibility configuration (Figure 3.7). For example, if the present configuration is a clique of size 4 as shown in Figure 3.7(a), change it to a clique of size 3 as shown in Figure 3.7(c) by simply swapping the constrained modules in ψ_1^+ to disjoint this node from the original clique of size 4. In contrast, when we change the feasibility configuration from a clique of size 3 to a clique of size 4, then we reorder the constrained sequence ψ_1^+ by moving a constrained module from one position to another.

During the *module-split* move, we restrict the feasibility configuration such that every combination of two sub-module pairs constructs the same type of feasibility. For example, let us consider sub-modules $\{A_1, B_1\ C_1\}$ and $\{A_2, B_2, C_2\}$ that are vertically constrained in two layers. Sub-modules, defined by the same name but using different subscripts, are split sub-modules of the same module which need to be vertically aligned. If we select the feasibility configuration of a clique of size 4, then every combination of two sub-module pairs such as $\{(A_1\ B_1), (A_2\ B_2)\}$, $\{(A_1\ C_1), (A_2\ C_2)\}$ and $\{(B_1\ C_1), (B_2\ C_2)\}$ will all have a size 4 clique. We sacrifice some feasible solutions but save runtime. Since we grow the feasibility configuration by splitting modules sequentially, we save the configuration information. Before splitting, we retrieve the saved feasibility configuration and find the position of a

module *w* in the sequence pair which is about to be split. We identify the next consecutive layer where a split part of the original module will be placed. For example, in a 4-layer chip, if "*w*" is in layer 2, then its second split part can be either sent to layer 1 or layer 3, which is determined randomly. Let us assume that layer 1 is selected and the initial sequence pair in layer 2 is shown in Figure 4.5(a). $\{A_2, B_2, C_2, D_2\}$ in layer 2 are initially under vertical constraints with $\{A_1, B_1, C_1, D_1\}$ of layer 1 with the feasibility configuration of a clique of size 4. Module "*w*" of layer 2 has been chosen for splitting. Take the positive sequence of *layer 2* as a reference which indicates that "*w*" resides between B_2 and C_2. In the negative sequence of *layer* 2, use *insert* to move w_2 such that it can reside between B_2 and C_2 as shown by the dotted arrow in Figure 4.5(a). In the sequence pair of *layer 1*, find the indices such



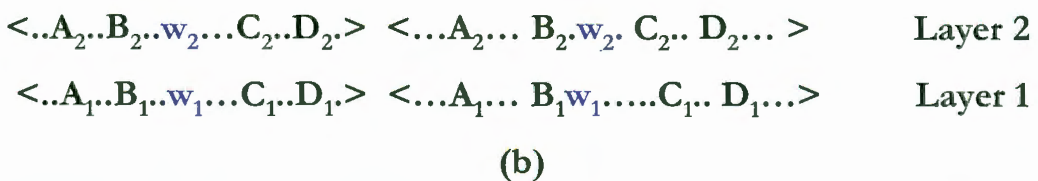


Figure 4.5: An example of Module split move. (a) Sub-modules $\{A_1, B_1, C_1, D_1\}$, $\{A_2, B_2, C_2, D_2\}$ are vertically constrained in layer 1 and layer 2 under the feasibility configuration of a clique of size 4 and their sequence pairs are shown. Module "w" initially resides in layer 2 which is about to be split. (b) Resultant sequence pairs of layer 1 and layer 2 after splitting.

that part of "*w*" can be placed between B_1 and C_1 after splitting. *Split & insert* the sub-module w_1 in both sequences of *layer 1* to these indices as shown by the bold arrows of Figure 4.5(a). The final GSP is shown in Figure 4.5(b). Please note that the feasibility configuration remains unchanged.

A non-uniform probability is assigned for the selection of each move. We dynamically change the probabilities of all moves in three different stages based on the quality of the solution as it proceeds towards convergence. At the beginning, we assign higher probabilities to those moves which create large perturbations (e.g. invert, exchange, change-group, and change feasibility configuration). For *insert, swap,* and *rotate,* we increase their probabilities gradually in stage 2 and stage 3 while reducing the probabilities of larger perturbing moves. We also use multi-stage termination criteria in the designed floorplan. For example, if a solution is stuck into local minima at stage 1 and reaches the termination criteria, then we immediately move the probability table to stage 2 and reset the termination criteria. This insures that the floorplanner will use all the three stages of the probability table that are dynamically varied during runtime.

In general, a set of modules in a floorplanning problem might be identified as eligible for splitting; otherwise candidates might be chosen based on predefined criteria. We first choose a set of modules which is eligible for splitting based on module sizes. We normalize the areas of all modules compared to the average area of modules. Next we disqualify the smaller modules from being an eligible candidate for module splitting because splitting smaller modules might not give significant benefits in terms of the system level total wirelength reduction. In the next step we prepare small buckets containing a set of

modules. The total number of modules can be different in each bucket. For each individual solution of the evolutionary algorithm, we randomly assign one of these buckets that contain a set of module split candidates. The module split move only chooses a module from the assigned bucket corresponding to each individual floorplan in the population. Once a module gets split, it is removed from the bucket. If the sub-modules of a module are merged, the module is back injected into the module split bucket. In addition to this strategy, we tried keeping all candidates together in one bucket, and assigning the same bucket to each individual floorplan solution. However, our experiments did not show good results possibly due to aggressive splitting which results in a large packing/footprint area and hence bad (drastically increased) inter-module wirelength..

4.7 COST FUNCTION

In every generation of the evolutionary algorithm, we obtain a set of new floorplan solutions due to minimization of a weighted cost function. We incorporate inter-module wirelength, reduction in intra-module wirelength due to splitting, via count, and dead space in the cost. When vertical constraints are disabled, then the cost function optimizes dead space, inter-module wirelength, and inter-module via count only using the following cost function:

$$Cost = \alpha \ DS + \beta \ WL + \gamma_1 \ VC_{\text{inter}} \tag{4.2}$$

Upon activation of placement-aware module splitting, the cost function changes to:

$$Cost = \alpha \ DS + \beta \ (WL - \Delta W_{\text{intra}}) + \gamma_1 \ VC_{\text{inter}} + \gamma_2 \ VC_{\text{intra}} \tag{4.3}$$

where ***DS*** is the dead space, ***WL*** is inter-module wirelength (*obtained from net information*), $\Delta \mathbf{W}_{\mathbf{intra}}$ is reduction in intra-module wirelength due to placement of 3-D modules (*obtained using the statistical method of section 4.5*), and VC_{inter} and VC_{intra} are inter-module and intra-module via count respectively. The constants α, β, γ_1, and γ_2 are real-valued tuning parameters that are designer specified for changing the quality. These tuning parameters give additional weight to any component of the fitness function.

In addition to the proposed cost function of eqn. (4.2) and eqn. (4.3), several other cost functions such as *a)* adding total area instead of dead space, *b)* adding the reciprocal of $\Delta \mathbf{W}_{\mathbf{intra}}$ instead of subtracting it from the inter-module wirelength, *c)* combining inter and intra module via counts, and *d)* introducing system level total wirelength instead of reduction in inter-module wirelength were tried and their solution qualities were examined. It was observed that none of these cost functions gave good solution quality. Therefore the best performing cost function as shown in eqn. (4.3) was selected.

Although the values of tuning parameter seem trivial, there is no straightforward way to determine their exact values. One of fundamental problems for tuning parameters is that various components of the fitness function may have different importance, as well as they may have different units [104]. For example, the dead space (DS) has a unit in μm^2 whereas wirelength has a unit in μm. Similarly, via count has no unit, and if the height of vias are considered then they have a unit in μm. Thus determination of tuning parameters involves learning experience based on the experimental results on different types of floorplanning problems. For the floorplanning benchmarks that are frequently used by academic researchers, we have observed the following ranges of tuning parameters: *a)* α is used

between 0.5 to 2.0, *b)* β is used between 0 to 20, *c)* γ_1 is used between 100 to 5000, and d) γ_2 is used between 500 to 15000.

The values of these tuning parameters can noticeably change the quality of floorplanning solutions. For example, if α is chosen very high, then the floorplanner will put more emphasis on area. Similarly, if β is kept high and α is kept low, the floorplanner will put more emphasis on wirelength. Furthermore, if β is kept zero then the floorplanner will not optimize wirelength at all. If γ_1 is kept in its lower range then the floorplanner will insert more inter-module vias in order to minimize wirelength. In contrast, if γ_1 is kept in its upper range, inter-module via height will be minimized, but inter-module wirelength might go up. If γ_2 is kept in its lower range, the floorplanner may try to split larger numbers of modules. Although it seems desirable, the increase in vertical constraints (imposed by placement aware module splitting) also increases area (see section 4.9), which will in turn worsen the inter-module wirelength.

4.8 DESIGN FLOW OF THE PLACEMENT-AWARE 3-D FLOORPLANNING ALGORITHM

Figure 4.6 shows the flow chart of our placement-aware 3-D floorplanning algorithm that was extended from the initial idea of module splitting (*Please note that the initial software inherited from Benyi is not a placement-aware floorplanner as it only optimizes area and inter-module wirelength, and it does not have any engine/model to capture the effect of placement inside 3-D modules*). We start with a finite set of randomly generated floorplans without any split modules. Then we improve the quality of the solution by making changes in the solution space using a pre-

defined set of moves (*Section 4.6)* and their probabilities are dynamically varied. During perturbation, we explore the possibility of 3-D placement inside a module (*module-split move in Section 4.6*) by statistically estimating the reduction in wirelength *(Section 4.4*). *If a module is split into multiple parts, its sub-modules are put under vertical constraints (Section 3.2) at runtime.* Furthermore, we restrict moves *(see Section 4.6)* such that they satisfy the feasibility condition theorems of Section 3.4. We use the LCSLS algorithm (*Section 3.6*) to evaluate the module packing.

As described in Section 4.6, we change the probabilities of selecting various moves in three stages during the floorplanning process. To avoid premature termination, if the termination criterion reaches before the end of the first stage, we switch the probabilities of moves to the second stage and reset the termination criteria. Similarly if a premature termination criterion is reached in the second stage we immediately move to the third stage. This step ensures that the floorplanner utilizes all three stages and uses all perturbing moves of Section 4.6. We call it a multi-stage termination criterion.

In Figure 4.6, the leftmost rectangles show the significant modifications made, or the new methods (algorithms) introduced during the various steps of the floorplanning algorithm.

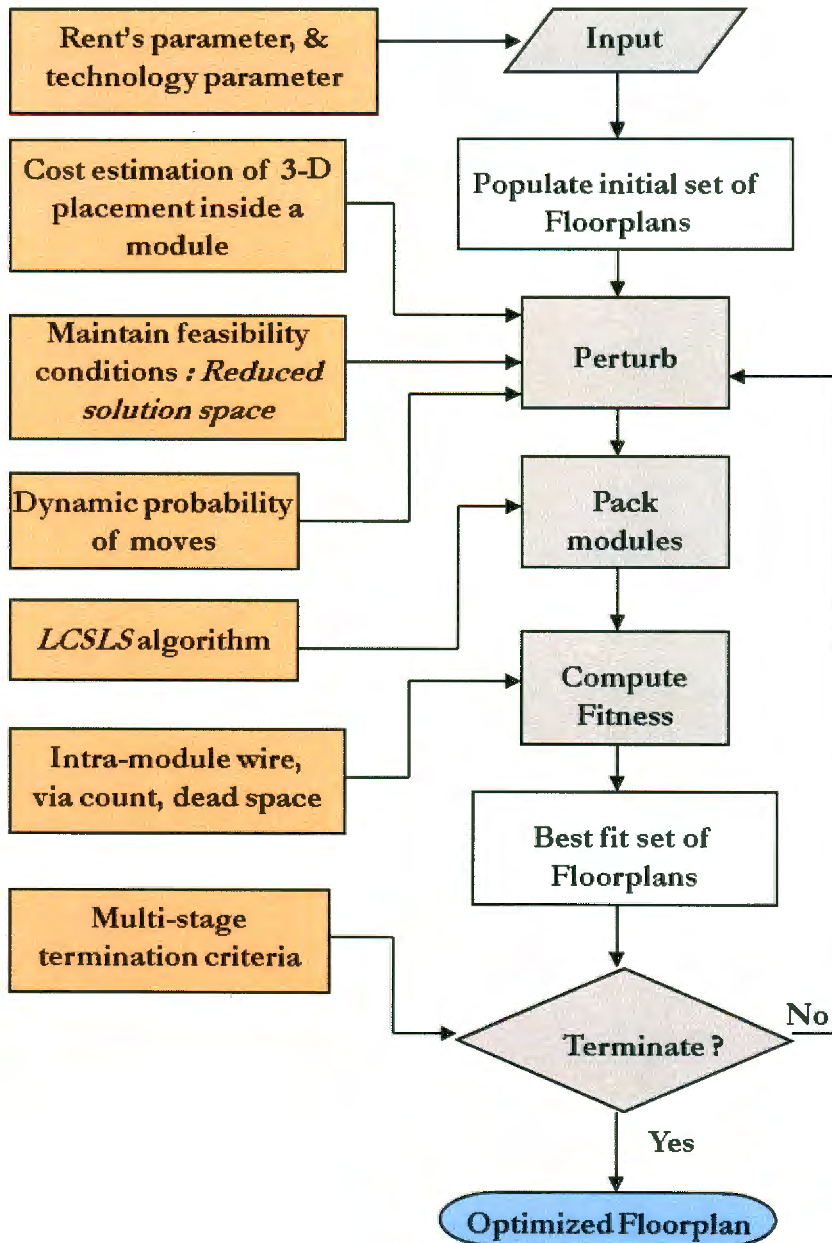


Figure 4.6: Flowchart of the proposed placement-aware 3-D floorplanning using vertical constraints (3-D FVC). The grey shaded portions have been modified from the initial algorithm. The left most rectangular boxes in orange shade are the new methods.

4.9 EXPERIMENTAL RESULTS

The proposed placement-aware 3-D floorplanning using vertical constraints (3-D FVC) was implemented in C++/STL. All experiments were performed on a Sun V490 server (*4xDual Core Sun SPARC IV+ CPUs, each running at 1.35GHz speed and total 32GB RAM*). The algorithm is designed to run on a single core only. Our 3-D FVC related experimental data are an *average* of 20 runs of each benchmark.

We used the two largest MCNC benchmarks (*ami33 and ami49*) and three largest GSRC benchmarks (*n100, n200 and n300*) for our experiments. The number in each benchmark's name denotes the total number of modules in the floorplan.

4.9.1 Experimental Setup

For a module's intrinsic wirelength estimation, we assigned different Rent's parameters *(k,p)* to different modules in the floorplan. Different values of Rent's parameters were taken from [77], assuming that the modules represent different types of circuits, such as logic, SRAM, microprocessor, control circuitry etc. The Rent's parameters were randomly chosen from [77] for different modules of floorplanning benchmarks. These parameters for each module of benchmark circuits have been given in Appendix C. The Rent's parameters for different types of circuits differ due to difference in the routing topology, complexity of wiring, and different number of I/O terminals.

The choice of technology node is limited by the number of nets connecting different modules, the number of gates inside modules and a need to stay within an acceptable range of Rent's parameters that describe the relation between gate count and I/O pins of a

module. We used a 0.25μm technology node for ami33 and ami49 because they are old benchmarks. In addition we have chosen a 0.25μm technology node for n100, a 0.13μm technology node for n200 and a 65 nm technology node for n300. This is because the aggregate area of all modules remains the same in each of these GSRC benchmarks but the complexity of inter-module wiring and the number of modules increase gradually. In other words, the wiring complexity and the number of modules increase while the area remains constant, which can be interpreted as technology scaling. The units of MCNC and GSRC benchmarks are 1 μm and 10 μm respectively. The same units were assumed in CBA [13].

We assume that a 2-input NAND gate cell has an area equivalent to $1500\lambda^2$ (*to estimate the number of gates inside a module*) and the average fanout of the circuit is 3.0. Please note that, for a real design, Rent's parameters (*k,p*) can be preprocessed from the optimized netlist (*after logic synthesis*) of a module [77]. A brief description of the Rent's parameter extraction is given in Appendix C. Furthermore, average fanout and the total number of gates can be obtained from logic synthesis.

4.9.2 Comparison of 3-D Packing Algorithms with Vertical Constraints

The packing algorithm translates the topological floorplan representation into a geometric floorplan, i.e. it assigns the (x, y) coordinates of the lower left corners of modules and determines the chip dimensions. In real life this assignment of x-y coordinates can be identified as floorplanning. However, in the floorplanning CAD tool, the module packing algorithm only acts as a decoder of the data-structure (i.e. floorplan representation) that translates the floorplan representation into the real geometric floorplan. However the floorplanning tool requires many more operations to search for an optimal floorplan than

TABLE 4.2: RUNTIME COMPARISON OF 3-D PACKING WITH VERTICAL CONSTRAINTS

Circuit	3D-FVC runtime using 3DCG (s)	3D-FVC runtime using LCSLS (s)	# of vertically aligned sub-modules (count)
ami33	54	23	28
ami49	98	52	18
n100	304	139	40
n200	851	263	50
n300	2001	445	48
Total	3308	922	184

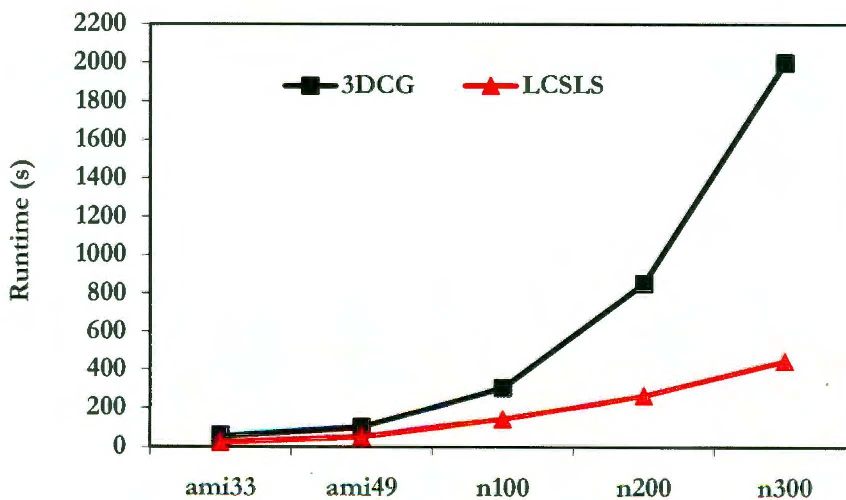


Figure 4.7: Runtime of 3-D FVC (with increasing problem size) using various packing algorithms with vertical constraints. LCSLS is faster than 3DCG.

simply decode the data structure to a geometric floorplan, as it was shown in Section 4.8. Thus, the packing algorithm is a part of the overall floorplanning CAD tool.

We compare the effect of 3-D packing with vertical constraints algorithms (*Sections 3.5 and 3.6*) on the runtime of 3-D FVC. Table 4.2 shows the runtimes of each benchmark obtained using 3DCG and LCSLS algorithms. The LCSLS algorithm is approximately 2x-5x faster than the 3DCG algorithm on these benchmarks. When the problem size increases, the difference in runtime is larger and this trend can be easily observed in Figure 4.7. It also verifies the asymptotic complexity analysis of 3DCG and LCSLS algorithms presented in section 3.6. Based on this observation we have performed all the remaining experiments using the LCSLS packing algorithm. During our experiment we have observed that the 3DCG and LCSLS algorithms produce identical floorplans for a given Grouped Sequence Pair, and a set of vertical constraints. This observation was made by comparing the Latex output files generated by 3DCG and LCSLS algorithms during a series of floorplanning experiments. We have not shown floorplans obtained by the 3DCG and LCSLS algorithms because it will be impossible to visually distinguish two identical floorplans (because there is no difference).

4.9.3 Comparison of Area and Wirelength Minimization without Vertical Constraints

In this sub-section, we compare our results with two state-of-the-art algorithms, CBA [13] and 3-D STAF [65]. The thermal optimization in CBA and 3-D STAF is disabled and their results are taken directly from [13] and [65]. For a fair comparison, module splitting and vertical constraints were disabled in 3-D FVC. We exclude via height in inter-module

wirelength (HPWL) calculation, similar to CBA and 3-D STAF [78]. We report footprint area, HPWL, via count and runtime. If a net spans from layer 1 to layer 4, then we count three vias. The same method was used in CBA and 3-D STAF [78].

Compared to CBA, 3-D FVC produces a 12.6% smaller footprint area, 24.8% smaller wirelength and 16.9% fewer via count, as shown in Table 4.3. These parameters are also better than or comparable to those generated by 3-D STAF.

Table 4.4 shows a runtime comparison among CBA, 3-D STAF and 3-D FVC. Please notice that the runtime of CBA taken from [13] was obtained using a 750 MHz CPU, the runtime of 3-D STAF [65] was reported using a 3 GHz CPU and our CPU speed is only 1.35 GHz. Therefore it is only an approximate comparison. We leave it to readers to judge the performance. Using linear scaling of runtimes with respect to the 3 GHz CPU speed, it is clear that 3-D FVC is faster and it also scales well with increasing problem size compared to CBA and 3-D STAF as shown in Figure 4.8.

TABLE 4.3: COMPARISON OF FOOTPRINT AREA, WIRELENGTH AND VIA COUNT OPTIMIZATION WITHOUT VERTICAL CONSTRAINTS

Circuit	CBA† [13]			3D-STAF† [65]			3D-FVC without module splitting		
	Footprint (mm2)	HPWL (mm)	Via (count)	Footprint (mm2)	HPWL (mm)	Via (count)	Footprint (mm2)	HPWL (mm)	Via (count)
ami33	0.35	22.5	93	0.35	22.0	122	0.36	20.8	88
ami49	14.90	446.8	179	13.49	437.5	227	10.9	449	378
n100	5.29	1005	955	5.9	913	828	5.20	678	717
n200	5.77	2103	2093	5.9	1686	1729	5.42	1530	1495
n300	8.90	3150	2326	9.7	2379	1557	8.89	2380	2014
Aggregate relative to CBA				+0.37%	-12.2%	-21%	-12.6%	-24.8%	-16.9%

† Thermal optimization was disabled in CBA and 3-D STAF whereas module splitting and vertical constraints were disabled in 3-D FVC for a fair comparison. The results of CBA and 3-D STAF were taken directly from their original sources [13] and [65] respectively.

TABLE 4.4: RUNTIME COMPARISON WITHOUT VERTICAL ALIGNMENT

Circuit	Actual runtime on different processor speed		
	CBA at **750 MHz** (s)	3D-STAF at **3.0 GHz** (s)	3D-FVC at **1.35 GHz** (s)
ami33	23	52	10.3
ami49	86	57	36.1
n100	313	68	60.5
n200	1994	397	125.2
n300	3480	392	209.7
Total	5896	966	442

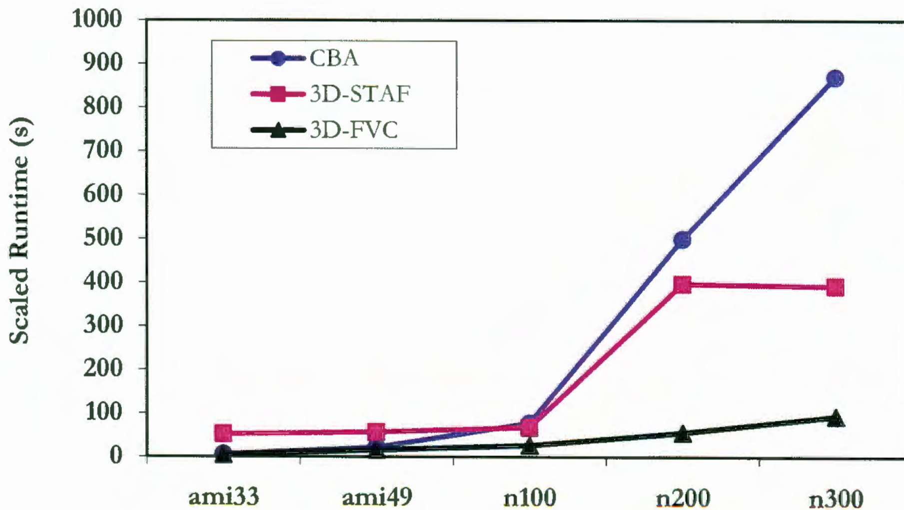


Figure 4.8: Runtime comparison of various 3-D floorplan algorithms with increasing problem size. Thermal optimization and vertical alignment were disabled. All runtimes have been scaled linearly to 3 GHz CPU speed.

4.9.4 Impact of Placement-Aware 3-D Floorplanning with Vertical Constraints on System Level Wirelength

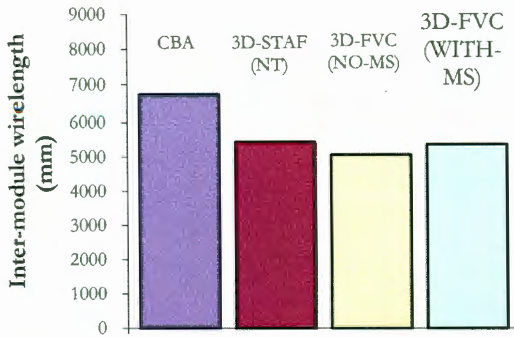
We present the results of 3-D FVC with module splitting on MCNC and GSRC benchmarks. Table 4.5 shows the footprint area, inter-module wirelength (HPWL), total wirelength, via count, runtime, and reduction in intra-module wirelength due to placement-aware module splitting. When placement aware module splitting is enabled, 3-D FVC reduces the system level total wirelength (*inter-module wire + intra-module wire*) by 9.88% (*559628 vs. 504356 mm*). The reduction in intra-module wirelength is ~10X compared to the inter-module wirelength (*5539 vs. 55705 mm*). Furthermore, the runtime is comparable to or better than other state-of-the-art algorithms.

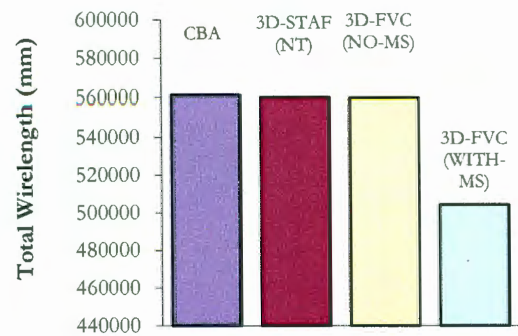
The inter-module wirelength goes up slightly upon activation of module splitting in 3-D FVC because the cost of module splitting is activated in the cost function (*see eqn 4.3 in Section 4.7*), i.e. ΔW_{intra} changes from zero to a positive number. Therefore the EA engine sees the introduction of an additional parameter in the cost function for combinatorial optimization. If there is a large reduction in intra-module wirelength at the expense of a minor increment in inter-module wirelength, the optimization engine accepts it as a better solution. 3-D FVC purposely introduces additional vias (*intra-module vias*) to further reduce wirelength. However, the increase in via count is still under the projected via-density [10] by the year 2010 onwards (please see Figure 1.7 in chapter 1). Please note that this projected via density is the basis of our assumption that vias will not be a strong limiting factor in future 3-D ICs as explained in Section 4.1. Upon activation of module splitting, we observe an increment in footprint area which may be due to the following reasons: *a)* area and

wirelength optimization generally follows a trend similar to a banana curve; *b)* to satisfy the vertical constraints, the LCSLS packing algorithm laterally shifts sub-modules, which may result in area overhead.

In this experiment, the values of tuning parameters were chosen for various benchmarks as follows *a)* for ami33, $\alpha = 1$, $\beta=17$, $\gamma_1 = 2000$, $\gamma_2 = 5000$; b) for ami49, $\alpha = 1$, $\beta=12$, $\gamma_1 = 5000$, $\gamma_2 = 15000$; c) for n100, $\alpha = 1$, $\beta=10$, $\gamma_1 = 2000$, $\gamma_2 = 5000$; *d)* for n200, $\alpha = 1$, $\beta=8$, $\gamma_1 = 2000$, $\gamma_2 = 4000$; and *e)* for n300, $\alpha = 1$, $\beta=15$, $\gamma_1 = 2000$, $\gamma_2 = 15000$. As discussed in Section 4.7, the floorplanning results may change if different tuning parameters are used in the fitness function. A set of ranges for the tuning parameters have been also suggested for designers in Section 4.7 which is based on the sensitivity analysis presented in Appendix D.

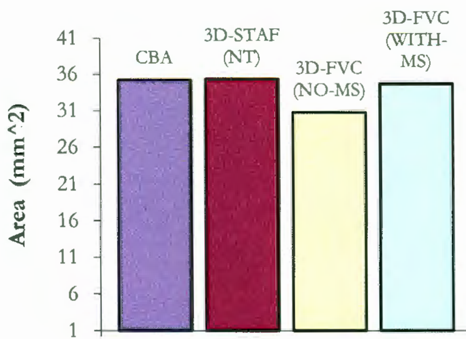
A comparison of various parameters obtained from different floorplanners is presented in Figure 4.9 which shows that our footprint area, inter-module wirelength, and inter-module vias remain smaller or comparable to CBA and 3-D STAF's results (*without thermal optimization*; *Table 4.3*). Please note that prior to module splitting, the intra-module wirelength is the same for any floorplanner which does not split modules. Thus we statistically compute the intra-module wirelength of all 2-D modules and use it to estimate the total wirelength of CBA, 3-D STAF and 3-D FVC by adding their respective inter-module wirelengths. In addition, inter-module wirelength is a small fraction of the total wirelength. Therefore Figure 4.9(b) shows similar total wirelength for CBA and 3-D STAF. However, 3-D FVC (with MS) reduces total wirelength by ~9.8% compared to both floorplanners. A 3-D-floorplan of ami49 with split modules and satisfying all placement-aware constraints is shown in Figure 4.10.

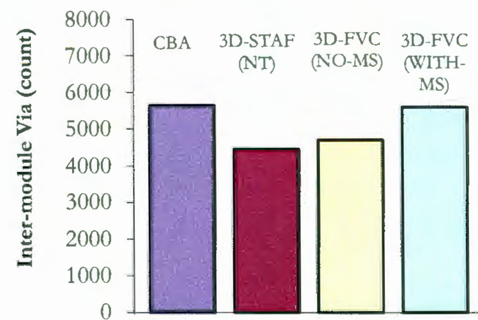




(a)

(b)





(c)

(d)

Figure 4.9: Comparison of (a) inter-module wire, (b) total wirelength (c) footprint area, and (d) inter-module via count, obtained using different floorplanning tools. Total wirelength is the sum of inter and intra module wirelength. Vertical constraints are applied only when module splitting (MS) is activated. 3-D FVC (with MS) reduces total wirelength by 9.8%. The bar charts represent the aggregate data of all the benchmarks. Please note that each chart has a different y-scale.

TABLE 4.5: EFFECT OF 3-D FVC WITH MODULE SPLITTING ON THE SYSTEM LEVEL TOTAL WIRELENGTH ON A 4-LAYER 3-D FLOORPLAN.

Circuit	Footprint (mm^2)	Inter-module wirelength (mm)	System level total wirelength (mm)	Inter-module via (count)	Intra-module via (count)	Runtime at 1.35 GHz CPU (s)	Reduction in intra-module wirelength (mm)
ami33	0.44	26.7	1611	133	561	29.0	-485
ami49	12.4	504	113448	448	9867	56.7	-30000
n100	5.95	738	32102	972	3531	165.4	-7370
n200	5.99	1590	69726	1776	4689	301.9	-6050
n300	9.91	2500	287469	2269	4186	506.8	-11800
Aggregate	34.69	5358.7	504356	5589	22834	1059.8	-55705

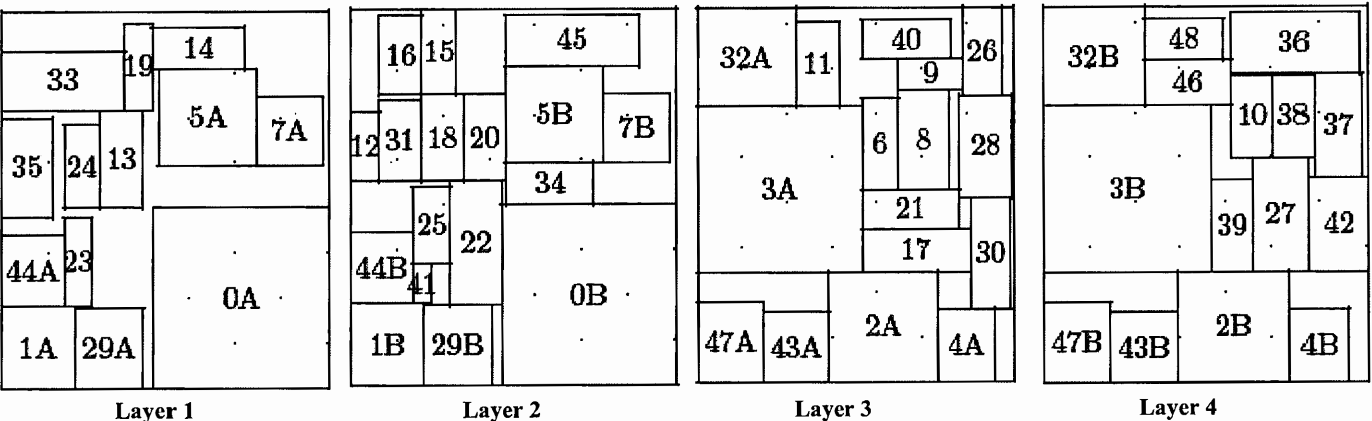


Figure 4.10: A 4-layer 3-D floorplan of ami49. Please notice that parts of module 0 (0A, 0B), 1, 2, 3, 4, 5, 7, 29, 32, 43, 44 and 47 have been placed in consecutive device layers. The planar locations of sub-modules are the same.

TABLE 4.6: EFFECT OF FEASIBILITY CONDITIONS ON THE SOLUTION QUALITY OF 3-D FVC

Circuit	3D-FVC ***without*** feasibility conditions				3D-FVC ***with*** feasibility conditions			
	Footprint (mm^2)	Inter-module wirelength (mm)	Reduction in intra-module wirelength (mm)	No. of split modules (count)	Footprint (mm^2)	Inter-module wirelength (mm)	Reduction in intra-module wirelength (mm)	No. of split modules (count)
ami33	0.40	24.7	-311	8	0.44	26.7	-485	15
ami49	12.4	534	-29600	9	12.4	504	-30000	9
n100	5.68	709	-5710	16	5.95	738	-7370	25
n200	5.53	1560	-2680	9	5.99	1590	-6050	25
n300	9.43	2510	-7780	17	9.91	2500	-11800	25
Aggregate relative to 3D-FVC *without* feasibility conditions					+3%	+ 0. 3%	- 20%	+ 68%

4.9.5 Effect of Feasibility Conditions on the Solution Quality of 3-D FVC

To study the benefits of feasibility conditions, we performed experiments in two different cases *a)* we *disabled* the feasibility conditions and removed the restricted set of associated moves (of Section 4.6) from the 3-D FVC, and *b)* we used the 3-D FVC algorithm described in Section 4.8 which is our placement-aware 3-D floorplanning tool i.e. feasibility conditions were *activated* in this case. Both of these cases have the same set of tuning parameters and the same cost function (as defined in Section 4.7). Next we compared the results of experiments obtained using these two cases. Table 4.6 shows the comparison of footprint, inter-module wire, reduction in intra-module wirelength due to splitting and number of split modules. Please note that 3-D FVC *with feasibility conditions* (i.e. case *b*) splits 68% more modules and reduces intra-module wirelength by 20% at the expense of a 3% increment in footprint compared to 3-D FVC *without* feasibility conditions (i.e. case *a*). The inter-module wirelength remains comparable. Runtime increases by 10% because packing takes longer due to an increased number of split modules/vertical constraints. Thus feasibility conditions split more modules and further reduce wirelength. As a result, it introduces an additional number of intra-module vias which is still under the projected via density (by IBM) as previously discussed in Sub-section 4.9.4.

Since our approach introduces intra-module vias to harness the advantages of 3-D integration, our approach is limited by the via density and the penalty associated with by the TSV technologies. If the TSV technology does not allow dense TSV placement within 3-D ICs, then our approach will not work. Fortunately the continuous advancement of TSV technology favors our assumptions, and TSV size continues to shrink [10][69]. Furthermore

our algorithm is limited by the optimization of area, system-level wirelength, and via count. Our algorithm is based on a stochastic search method (evolutionary algorithm) in which different solutions are obtained for different experimental runs. Additionally, using different tuning parameters for the cost function (described in Section 4.7) changes the solution quality. This is a known issue in the combinatorial optimization based on stochastic search methods (such as SA and EA) [104].

5.1 INTRODUCTION AND MOTIVATION

In this chapter, vertical alignments between modules assigned to different device layers in 3-D ICs are considered at the floorplanning stage. Vertical alignments are important in memory intense design, mixed signal, and future comprehensive systems due to delay, substrate noise, and power reduction, to name just a few. In memory intense design such as a microprocessor consisting of a CPU and L2 cache memory will be required to be vertically aligned for low latency as shown in Figure 5.1. The alignment of blocks in different device layers is desired for aligning modules on the same bus (*bus driven 3-D floorplanning*) e.g. SIMD (Single Instruction, Multiple Data) and MIMD (Multiple Instruction Multiple Data). The SIMD performs the same type of operation on multiple data in which a common data pool is connected to multiple processing units. The MIMD architecture has a number of processing units that function independently using the same data pool. MIMD machines with shared memory have processors which share a common, central memory. In a simple approach, all processors are attached to a bus which connects them to memory. These constraints will be defined based on the design information which is available to designers. Furthermore it might be desired to control the relative position of certain blocks to reduce temperature and noise inside the 3-D chip. One approach could be to use a thermal and noise model to for simultaneous optimization during floorplanning. However, in the presence of macro blocks, the power and temperature profile of macro blocks could be easily available to designers. Similarly the knowledge of noisy digital blocks and noise

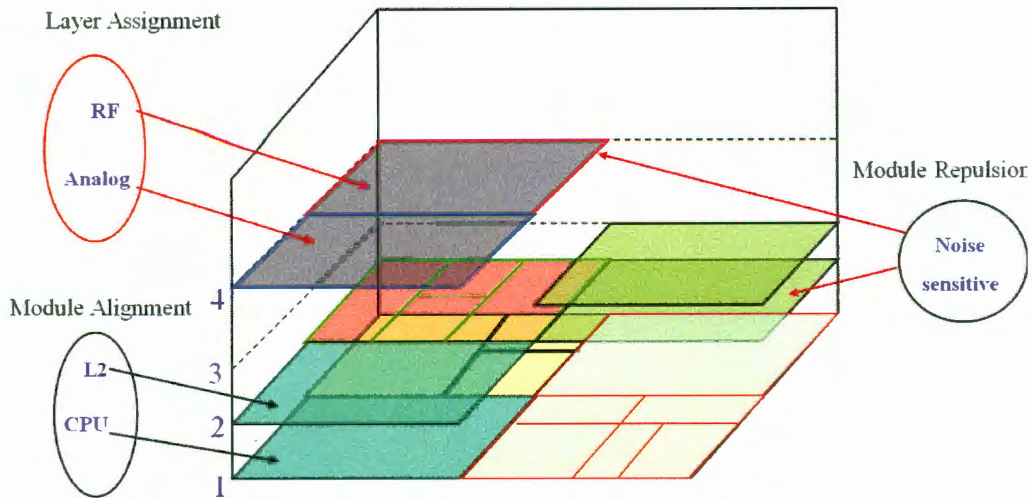


Figure 5.1: Example of different placement constraints in 3D floorplanning.

sensitive analog blocks will be available to designers in the case of macro blocks. It might also be necessary to restrict certain group of modules within a particular set of device layers due to different types of substrates (such as bulk-Si or Silicon-on-Insulators), different types of technology nodes, or due to thermal constraints. For example, in a microprocessor consisting of a CPU logic and an L2 cache block, the CPU is generally hotter (due to ALUs and control logic) than L2 cache memory. In this case, the CPU block can be placed closer to the heat sink whereas L2 cache memory can be placed above the CPU and away from the heat sink. Finally, certain groups of modules might be required to be spatially separated due to the noise sensitivity of the modules as shown in Figure 5.1.

The advantages of designer specified constraints are that designers get a control over the optimization process. Based on the design information (either at block level power, thermal or noise analysis, or information from third party designers for macro blocks), designers

might have want to impose certain constraints prior to the optimization process. Furthermore, in the case of design reuse, designers can use their prior experience and knowledge from the previous design. Finally the designer specified constraint also reduces the search space of the floorplan. In case of optimization through the cost function of the floorplanner, an accurate and fast noise, thermal, and power model is required, and these models will add additional dimensions in the solution space that will also increase runtime. Since the floorplanning algorithms are based on stochastic search methods, it might not be possible to reach the desired solution quickly. Furthermore, it does not allow any designers to control the optimization process in terms of the vertical module alignment, module repulsion, and layer assignment constraints shown in Figure 5.1. However, if designers do not have sufficient information about the circuit blocks, optimization through the cost function might be desired. In this problem we focus on designer-specified sets of placement constraints only.

5.2 PROBLEM FORMULATION

We present a 3-D floorplanning with module alignment (3-D FMA) algorithm that considers *a)* vertical alignment of modules located in different device layers (*vertical constraint*), *b)* layer assignment for modules based on technology node or substrate type information (*layer assignment constraint*), and *c)* places certain set of modules away from each other (*module repulsion constraint*). Figure 5.1 shows an example of these constraints. In this algorithm, we allow designers to specify these constraints as input information.

Consider a set of n rectangular modules where each module M_i has a fixed area A_i, width W_i and height H_i connected by m nets. Let L be the total number of fixed active layers. Let (x_i, y_i, z_i) denote the lower left corner of the module M_i, where $1 \leq z_i \leq L$ and z_i belongs to the set of natural numbers. A *cluster* is defined as a set of floorplan modules. A set of clusters is composed of *several* independent *clusters*. Let $G = \{g_1, g_2, g_3 \ldots, g_k\}$ be a set of clusters. Thus each cluster $g_i \in G$ contains a set of floorplan modules. Modules inside each cluster $g_i \in G$ need to be vertically aligned by having the same lower left corner (x_i, y_i) but different z_i. Similarly, let $R = \{r_1, r_2, r_3, \ldots, r_k\}$ be a different set of clusters, and a particular cluster $r_i \in R$ contains a group of modules. Modules inside each cluster $r_i \in R$ need to be placed apart from each other by having different x_i and/or y_i such that they do not overlap. Furthermore, let D_i be a set of allowed device layers for a module M_i where it can be legally placed. A 3-D floorplan with module alignment is an assignment of (x_i, y_i, z_i) for each M_i and such that, all members of G satisfy *vertical constraint*, all members of R satisfy *repulsion constraints*, and each M_i satisfies *layer assignment constraint*. We seek a solution to the following problem:

3-D Floorplanning with Module Alignment (3-D FMA): *Given a set of n rectangular modules, with areas and aspect ratios, connected by m nets in L device layers, a set of vertical alignment constraints, module repulsion constraints, and layer assignment constraints, find a 3-D floorplan that satisfies all these constraints and minimizes chip area, inter-module wirelength, and the number of inter-module vias.*

5.3 COMBINATORIAL OPTIMIZATION AND THE COST FUNCTION

We use the evolutionary algorithm described in Chapter 4, as the main optimization engine for 3-D FMA. We use the *vertical constraints* of the grouped sequence pair (GSP) derived in Chapter 3 to explore within the feasible solution search space. However, a penalty function is incorporated in the cost function to satisfy the repulsion constraints. It is due to the fact that in all cases of feasibility and infeasibility conditions (derived in Chapter 3), it is possible to separate modules from each other. Finally, we use special operators (*restricted perturbation of the solution search space*) to satisfy the layer assignment constraints, and to maintain the feasibility of the solution.

Let a module i has the lower left X-coordinate x_i, and width w_i. Similarly another module j has the lower left X-coordinate x_j, and width w_j. Thus the pair wise penalty Δx_{ij} for these modules is calculated as:

$$\Delta x_{ij} = \frac{1}{|x_i - x_j| + \varepsilon} + \frac{1}{|(x_i + w_i) - x_j| + \varepsilon} + \frac{1}{|x_i - (x_j + w_j)| + \varepsilon} + \frac{1}{|(x_i + w_i) - (x_j + w_j)| + \varepsilon} \tag{5.1}$$

where ε is an infinitesimally small positive real number. Please note that each term of eqn(5.1) represents the distance (along the X-axis) between two corners of the two modules. In addition, if we detect that any corner of a module vertically overlaps with another module under MR constraint (i.e. the x coordinates of the two modules are the same across different device layers), we heavily penalize it by artificially increasing the value

of Δx_{ij}. Similarly Δy_{ij} is also calculated for the Y-coordinate of the two modules. However Δz_{ij} which denotes the vertical distance across the two device layers containing module *i*, and module *j* is calculated as shown in eqn (5.2):

$$\Delta z_{ij} = \frac{1}{|Layer(i) - Layer(j)| + 1} \tag{5.2}$$

where *Layer(i)* denotes the device layer ID of a module *i*.

Thus the repulsion penalty function, *Penalty(R)* is computed as eqn (5.3):

$$Penalty(R) = \sum_{k=1}^{size(R)} \sum_{i=1}^{r_k - 1} \sum_{j=i+1}^{r_k} \left(\Delta x_{ij} + \Delta y_{ij} + \Delta z_{ij} \right) \tag{5.3}$$

where *size(R)* is the total number of groups in the repulsion constraint set R, and r_k is the number of modules in a repulsion group $k \in R$.

Finally, the cost function (Fitness) for the EA engine is designed as shown in eqn. (5.4):

$$Cost = \alpha.DS + \beta.WL + \gamma.VC + \chi.Penalty(R) \tag{5.4}$$

where ***DS*** is dead space, ***WL*** is inter-module wirelength, and ***VC*** is inter-module via count. In addition, ***Penalty(R)*** is the sum of the penalty for violations of module repulsion constraints. The constants $\alpha, \beta, \gamma,$ and χ are tuning parameters that can be changed by the user to fine tune the solution quality. In the floorplan optimization, we minimize the weighted sum of the cost function given by eqn. (5.4). Please note that eqn. (5.4) is similar

to eqn. (4.2) which is used for area, inter-module wirelength, and via count minimization. The only difference is that the penalty function has been added to eqn. (4.2) to obtain eqn. (5.4). For the floorplanning benchmarks that are frequently used by academic researchers, we have observed the following ranges of tuning parameters: *a)* α is used between 0.5 to 2.0, *b)* β is used between 1 to 10, *c)* γ is used between 1000 to 5000, and d) χ is used between 1.0 to 2.0. The ranges of these tuning parameters are chosen based on the sensitivity analysis of the cost function presented in Appendix D.

The values of tuning parameters can noticeably change the quality of floorplanning solutions. For example, if α is chosen very high, then the floorplanner will put more emphasis on minimizing area. Similarly, if β is kept high and α is kept low, the floorplanner will put more emphasis on wirelength minimization. If β is kept zero then the floorplanner will not optimize wirelength at all. If γ is kept in its lower range then the floorplanner will insert more inter-module vias in order to minimize wirelength. In contrast, if γ is kept in its upper range, inter-module via height will be minimized, and inter-module wirelength might go up. If χ is kept zero, the floorplanner may not put more emphasis on separating modules under MR constraints. However, if χ is non-zero, then floorplanner puts a penalty in the cost function in order to guide the floorplanner to separate modules under MR constraints.

The values of tuning parameters used for experiments have been given in Section 5.5.

5.4 PERTURBATION OF THE SOLUTION SPACE

We randomly generate an initial set of floorplan solutions at the beginning that do not satisfy any of the user specified constraints. Next, we adjust the locations of modules under vertical constraints, repulsion constraints and layer assignment constraints. Then we balance the number of modules in each device layer.

To satisfy the vertical constraints of different sets of modules, we restrict the moves such that we only search within feasible solution search space by applying the feasibility condition introduced by the theorems of section 3.4. It was stated in theorem 1, *"from a software implementation point of view, theorem 1 can be satisfied by checking if either* $\{\psi_1^+, \psi_2^+\}$ *or* $\{\psi_1^-, \psi_2^-\}$ *are connected by an edge, i.e. modules in those node pairs are in the same order."* Thus we satisfy this condition by simply keeping the same orders of *constrained groups* of modules under vertical constraints in each layer, either in their positive constrained sequences or in their negative constrained sequences. For example, let us assume that modules inside each of the following three groups, $u = \{u_1, u_2, u_3, u_4\}$, $v = \{v_1, v_2, v_3, v_4\}$, and $w = \{w_1, w_2, w_3, w_4\}$ are under vertical constraints in a 4-layer 3-D IC. In this case, we can create constrained sequence pairs such that either all positive constrained sequences $\{\psi_1^+, \psi_2^+, \psi_3^+, \psi_4^+\}$ or all negative constrained sequences $\{\psi_1^-, \psi_2^-, \psi_3^-, \psi_4^-\}$ of all device layers have the same order of modules belonging to different groups. Continuing our example, if all the positive (or negative) constrained sequences of different device layers are as $\{u_1, v_1, w_1\}$, $\{u_2, v_2, w_2\}$, $\{u_3, v_3, w_3\}$, and $\{u_4, v_4, w_4\}$, the solution will always be feasible. This order can be changed, and as long as the corresponding changes in all device layers are the same, it will always insure feasible solutions. For example, $\{u_2, w_1, v_2\}$, $\{u_1, w_3, v_1\}$, $\{u_3, w_2, v_4\}$ and $\{u_4, w_4, v_3\}$ either in

the positive or in the negative constrained sequences will also produce feasible solutions. Please note that each set represents the positive (or negative) constrained sequence of a particular device layer. Furthermore, it does not matter which modules go to which device layer. It only matters that their group order be the same. In this method we sacrifice a fraction of feasible solution space but avoid checking of feasible solutions. For example, let us assume that {$\mathbf{a_1}$, $\mathbf{b_1}$, c_1} and {$\mathbf{b_2}$, $\mathbf{a_2}$, c_2} are the two *positive* constrained sequences of two device layers. In these two positive constrained sequences, only the order of {$\mathbf{a_1}$, $\mathbf{b_1}$} and {$\mathbf{b_2}$, $\mathbf{a_2}$} are different which makes it a different example than the previous one in this paragraph. If the two *negative* constrained sequences of the two device layers have either *a)* {a_1, b_1} and {a_2, b_2,}, or *b)* {b_1, a_1} and {b_2, a_2,} orders of modules, the solution will be feasible. Thus the restriction might not allow searching all the feasible floorplan solutions and in some cases a few good solutions might never be considered. In the worse case, i.e. *if there is only one good solution* which is not reachable by our restricted moves then the floorplanner might not find it. However, due to enormous solution space such as $n^{k-1}(n!)^2/(k-1)!$ for a *k*-layer 3-D chip containing *n* modules [45], the chances of reaching the worst case (*i.e. the best solution is not reachable by the floorplanner due to the restricted moves*) situation are negligible.

Next we use the following moves - *insert, swap, invert, exchange, change group, and rotate* that are the same moves as presented in chapter 4 (please see section 4.6). In addition, we have introduced the following set of new restricted moves (*special operators*) to perturb the solution search space:

- ***ValignExchange:*** In this move, we only exchange modules within a vertically aligned group. For example, if the group $g = \{m_1, m_2, m_3\, m_4\}$ has all its modules under vertical constraints and located in four different device layers, then we only perform inter-layer exchange of two randomly selected modules from this group.

- ***ValignChangeGroup:*** This perturbation is similar to *ValignExchange* except that it moves a module m_i of a group g from one device layer to another layer. Please note that this move can only be effective if the number of modules present in a vertically constrained group is smaller than the total number of device-layers in a 3-D chip.

- ***ValignPerturbFeasibilityOrder:*** This move changes the order of modules within the *constrained positive sequences* while keeping their orders the same in all device layers. For example, let us say that we swap two modules from two different vertically constrained groups, such that their relative position changes in one device layer. Then we perform the *swap* operation in all the remaining device layers between modules from these two constrained groups, such that the resultant order of modules' groups will be the same across all device layers. Similarly, *insert* and *invert* operations are performed on the *constrained positive sequences* of all device layers. The same operations can also be performed on the *constrained negative sequences* of all device layers.

The *ValignPerturbFeasibilityOrder* move explores the different feasibility configurations for module alignment (MA) constraints. At the same time, it also creates large perturbation in the search space because the order of modules (under MA constraints) in each device layer

needs to be rearranged to maintain the feasibility of a solution. The larger the probability of the *ValignPerturbFeasibilityOrder* move, the more frequently the entire floorplan of all device layers will change with a large perturbation. The large perturbation may be desirable at an early stage of the floorplan optimization. However, frequent use of this move may not lead to a good solution, or the solution might not converge. Other two perturbations, *ValignExchange* and *ValignChangeGroup* also search in the feasible solution space. However the perturbations created by these two moves only perturb two device layers instead of all device layers. We sometime perturb the layout of all the device layers in just one move, and rest of the time we only perturb the layout of two device layers. Thus the combination of these three moves helps in creating drastic as well as moderate perturbation while maintaining the feasibility of vertical alignment of modules.

The probabilities of moves are changed in three stages (similar to those discussed in Chapter 4). Thus, for each move we provide three probabilities corresponding to the three stages of the move. The probabilities are *a)* for *insert* (0.14, 0.14, 0.22), *b)* for *swap* (0.14, 0.20, 0.28), *c)* for *invert* (0.24, 0.18, 0.15) *d)* for *exchange* (0.13, 0.09, 0.05) *e)* for *ChangeGroup* (0.13, 0.09, 0.05), *f)* for *ValignExchange* (0.06, 0.1, 0.02), *g)* for *ValignChangeGroup* (0.04, 0.03, 0.01), *h)* for *ValignPerturbFeasibilityOrder* (0.07, 0.09, 0.02) , and *i)* for *rotate* (0.05, 0.08, 0.2). There is no straightforward mathematical theory behind the decision about the probabilities of moves. The values of these probabilities were chosen based on the initial experiments performed by varying the probabilities of various moves, and studying their effects on various floorplan benchmarks. A simple logic is to assign high probabilities at the early stage of the optimization process to those moves which create large perturbations

because initial solutions are random and they have not converged. At the later stage, high probabilities are assigned to those moves which create small perturbations because at the later stage floorplan solutions start to converge to an optimal solution.

5.5 EXPERIMENTAL RESULTS

We implemented 3-D FMA, the proposed 3-D floorplanning with module alignment, in C++/STL. All experiments were performed on a Sun V490 server (*4xDual Core Sun SPARC IV+ CPUs, each running at 1.35GHz speed and total 32GB RAM*). The algorithm is designed to run on a single core only. Our 3D-FMA related experimental data are an *average* of 20 runs of each benchmark.

We used the two largest MCNC benchmarks (*ami33 and ami49*) and three largest GSRC benchmarks (*n100, n200 and n300*) for our experiments. The number in each benchmark's name denotes the total number of modules in the floorplan. Since the MCNC benchmarks are old and small in problem size, we present our experiments for MCNC and GSRC separately. We used the following tuning parameters for various floorplan benchmarks: *a)* for *ami33*, $\alpha=1$, $\beta=20$, $\gamma=2000$, $\chi=2$, *b)* for *ami49*, $\alpha=1$, $\beta=12$, $\gamma=2000$, $\chi=2$, *c)* for *n100*, $\alpha=2$, $\beta=10$, $\gamma=2000$, $\chi=2$, *d)* for *n200*, $\alpha=1$, $\beta=10$, $\gamma=2000$, $\chi=2$, and *e)* for *n300*, $\alpha=1$, $\beta=12$, $\gamma=2000$, $\chi=2$. These values were chosen based on the sensitivity analysis presented in Appendix D which suggests a certain range for each tuning parameter for the optimal solution.

5.5.1 Effect of Module Alignment on MCNC Benchmarks

TABLE 5.1: EFFECT OF DIFFERENT PLACEMENT CONSTRAINTS ON A 4-LAYER 3-D FLOORPLAN USING MCNC BENCHMARKS

circuits	# of constraints			Footprint	HPWL	Via	Runtime	
	MA	MR	LA	(mm^2)	(mm)	(count)	Actual (s)	Relative
ami33	0	0	0	0.36	21.0	89	16.3	1.00
	3	0	0	0.46	27.2	91	21.4	1.30
	3	2	0	0.45	26.4	93	21.9	1.34
	3	2	6	0.46	27.8	94	22.9	1.40
ami49	0	0	0	10.9	424	473	45.7	1.00
	3	0	0	14.5	490	439	50.0	1.09
	3	2	0	14.1	512	465	53.7	1.17
	3	2	6	14.5	499	450	58.7	1.28

The ami33 and ami49 circuits from MCNC benchmarks only contain a small number of modules. Thus we keep the number of constraints the same on these benchmarks. First we perform experiments without any constraints as a baseline. Then we use only vertical constraints for module alignment (MA). Next we apply vertical constraints for module alignment (MA) and module repulsion constraints (MR). Finally we apply MA, MR, and layer assignment constraints (LA) simultaneously.

We report footprint area, inter-module wirelength (HPWL), via count (i.e. number of TSVs), and runtime on 4-layer 3-D floorplanning. Table 5.1 shows the comparison of these parameters. From the table, it is clear that 3-D FMA optimizes area, HPWL, and via count in the presence of various constraints. The runtime increases between 9% – 40% when the different types of constraints are combined for simultaneous optimization. When the MA constraints are applied, the footprint area increases on average by 30% and the HPWL increases by 15% – 30%. Via count remains approximately the same. Please

observe that the effect of MA constraints on the footprint area and HPWL dominates over the penalty caused by MR and LA constraints.

5.5.2 Effect of Increasing the Number of Module Alignment Constraints on 3-D Floorplanning using GSRC Benchmarks

In this subsection we use GSRC benchmarks and vary the number of module alignment (MA) constraints to observe its effect on the solution quality and runtime. Table 5.2 shows the varying numbers of MA constraints, and their effect on area, HPWL, via count, and runtime for n100, n200, and n300 benchmarks. We increase the number of MA constraints from 5 to 15. As a result, the footprint area increases on an average by 11.5% – 32.7%, HPWL increases by 4.5% – 15.5% whereas via count approximately remains the same. The runtime penalty is between 1.35x to 1.65x for module alignment constraints varying between 5 to 15. The area increases due to the module alignment that is achieved by laterally shifting modules during the LCSLS packing algorithm. Due to the alignment of modules, the wiring may need to take longer routing path and it results in increased HPWL. The effect of module alignment has no effect on via count.

5.5.3 Composite Effect of Increasing the Number of Various Constraints on 3-D Floorplanning using GSRC Benchmarks

In this experiment, we simultaneously apply MA, MR, and LA constraints, vary the number of these constraints, and observe their effects on the quality of floorplan solutions. Table 5.3 presents the comparison of footprint area, HPWL, via count, and runtime. The number of MA constraints is varied between 5 to 15, the same as previous subsection 5.5.2. We observe that the footprint area and HPWL increments are approximately the same as

TABLE 5.2: EFFECT OF INCREASING THE NUMBER OF MODULE ALIGNMENT CONSTRAINTS ON A 4-LAYER 3-D FLOORPLAN USING GSRC BENCHMARKS

Circuits	# of MA constraints	Avg. # of modules per MA group	Footprint (mm^2)	HPWL (mm)	Via (count)	Runtime	
						Actual (s)	Relative
n100	0	0.0	5.18	674	713	81.2	1.00
	5	3.0	5.84	716	700	102.5	1.26
	10	2.8	6.49	765	704	112.0	1.37
	15	2.6	6.78	811	707	125.9	1.55
n200	0	0.0	5.45	1530	1510	172.1	1.00
	5	3.0	6.12	1590	1518	233.3	1.35
	10	2.8	6.67	1660	1521	262.8	1.52
	15	2.8	7.16	1720	1512	268.4	1.55
n300	0	0.0	8.94	2390	2031	331.2	1.00
	5	3.0	9.80	2480	2020	521.1	1.57
	10	2.8	10.70	2570	2012	579.8	1.75
	15	2.8	11.50	2700	2013	602.5	1.81

TABLE 5.3: COMPOSITE EFFECT OF INCREASING THE NUMBER OF DIFFERENT CONSTRAINTS ON A 4-LAYER 3-D FLOORPLAN USING GSRC BENCHMARKS

Circuit	# of constraints			Footprint (mm^2)	HPWL (mm)	Via (count)	Runtime	
	MA	MR	LA				Actual	Relative
n100	5	2	8	5.77	714	701	106.8	1.00
	10	5	8	6.42	751	712	137.9	1.29
	15	8	8	6.78	823	701	148.4	1.38
n200	5	2	8	6.11	1580	1522	254.6	1.00
	10	5	8	6.68	1640	1518	276.6	1.08
	15	8	8	7.19	1720	1534	300.0	1.17
n300	5	2	8	9.88	2500	2045	491.1	1.00
	10	5	8	10.70	2600	2032	569.6	1.15
	15	8	8	11.15	2680	2018	602.2	1.22

previous subsection's result (presented in Table 5.2). This indicates that the effects of module repulsion (MR) and layer assignment (LA) constraints on the solution quality are negligible (on average within 1% of the results in Table 5.2) compared to MA constraints. It is due to the fact that MR constraints are satisfied by a penalty function, and they do not require any changes in the physical floorplan. The layer assignment constraints are satisfied by the restricted moves (special operators) during the perturbation of the solution search space. Thus it also does not require additional changes on the geometric floorplan whereas MA constraints involve lateral shifting of modules on the geometric floorplan for the vertical alignment.

5.5.4 Runtime Comparison of 3-D FMA with LTCG based 3-D Floorplanning Algorithm

In this sub-section we compare the runtime of our proposed 3-D FMA algorithm with the layered TCG (LTCG) based 3-D floorplan algorithm [67] for vertical module alignment. However, the comparison of 3-D FMA with LTCG is approximate due to the following differences *a)* LTCG performs floorplanning of soft modules whereas 3-D FMA performs floorplanning of hard modules, *b)* The problem formulation of LTCG is such that it vertically aligns "m" modules out of a set of "k" modules where $k > m$. In contrast, the number of modules for vertical alignment in each group is fixed in 3-D FMA, i.e. $m = k$ which is more strict constraint than LTCG's formulation *c)* LTCG [67] optimizes only area for soft modules and maintains vertical module alignment whereas 3D-FMA minimizes area and inter-module wirelength and via count while maintaining the vertical module alignment. Therefore we only compare runtime with LTCG in Table 5.4. In addition we

also report the total number of MA constraints for 3-D FMA, and LTCG based floorplanner. Please note that the runtime of LTCG was reported using a 3.2 GHz CPU [67] while our CPU's speed is only 1.35 GHz. Considering our slower CPU speed, 3-D FMA is faster than LTCG (please see Table 5.4). Figure 5.2 shows a runtime comparison of LTCG and 3-D FMA with the actual runtime, and linearly scaled runtime.

5.5.5 An Example of a 4-Layer 3-D Floorplan with Various Constraints

Figure 5.3 shows a 4-layer 3-D floorplan obtained using 3-D FMA. The module groups for MA constraints are {0, 3}, {1, 2, 5, 32}, and {10, 17, 20, 48}. Please observe in Figure 5.3 that modules within each MA group have been placed in different device layer and they are vertically aligned. Similarly module groups under MR constraint are, {4, 33} and {11, 28}. Please note that modules within each group are placed away from each other, and they are not vertically aligned. Finally, the group of modules under the layer assignment constraints in Layer 1 is {3, 33, 40}, in Layer 2 is {0, 7, 32}, in Layer 3 is {12}, and in Layer 4 is {13}. Please observe that modules under the LA constraint have been placed in their specified device layers. Thus 3D-FMA produces feasible solution satisfying these constraints.

The 3-D FMA algorithm can optimize area, inter-module wirelength, and via count while simultaneously satisfying designer specified set of constraints. These constraints are useful in bus driven 3-D design, and heterogeneous 3-D integration. The tradeoffs (in terms of footprint area and wirelength) associated with the MA, MR and LA constraints can be minimized by modifying the tuning parameters of the cost function as discussed in Section 5.3. An approximate runtime comparison with an LTCG based 3-D floorplanner [67] shows that 3-D FMA is faster and scales well with increasing problem size.

TABLE 5.4: RUNTIME COMPARISON OF 3-D FMA WITH LTCG BASED 3-D FLOORPLANNER

Circuits	LTCG @ 3.2 GHz CPU		3-D FMA @ 1.35 GHz CPU	
	Number of MA constraints [67]	Runtime [67] (s)	Number of MA constraints	Runtime (s)
ami33	2	28	3	21.4
ami49	3	63	3	50.0
n100	5	548	5	102.5
n200	N/A	N/A	5	233.8
n300	N/A	N/A	5	521.1

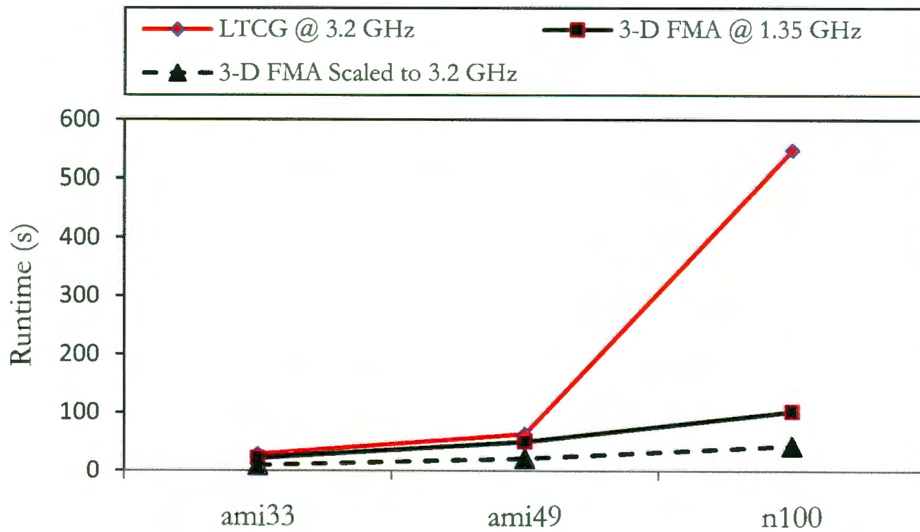


Figure 5.2: Runtime Comparison of 3-D FMA with LTCG

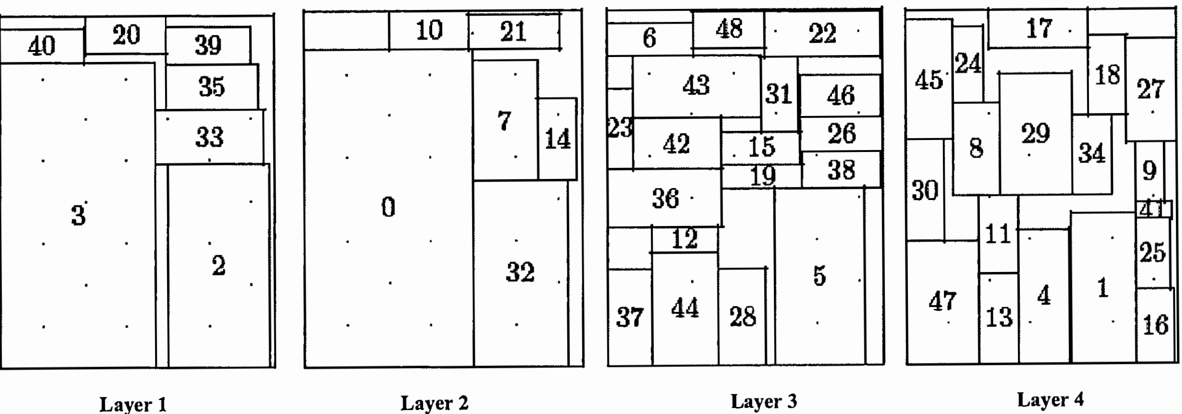


Figure 5.3: A 4-layer 3-D floorplan of ami49 obtained using 3D-FMA. Module groups under MA constraints are {0, 3}, {1, 2, 5, 32}, and {10, 17, 20, 48}. MR constraint groups are {4, 33} and {11, 28}. Layer assignment constraints in Layer 1 = {3, 33, 40}, Layer 2 = {0, 7, 32}, Layer 3 = {12}, and Layer 4 = {13}.

6.1 TSV FABRICATION TECHNOLOGIES

In this Section we will discuss the detailed fabrication technologies of through silicon vias that provide vertical interconnection across different device layers. As briefly discussed in Chapter 1, there are two types of TSV technologies depending on the order of their formation: *a)* via-first, and *b)* via-last.

In the via-first process, the majority of the inter-layer vias can be formed during the wafer bonding process (i.e. right after the wafer alignment process) but before the top wafer thinning process. The drilling of TSVs is performed by one of the two available processes *a)* laser drilling, and *b)* dry etching or Bosch etching. Laser drilling creates holes in the wafers without any micro cracks. Similarly, dry etching creates deep, steep-sided holes and trenches in wafers. The formation of TSVs consist of an electrical isolation layer such as

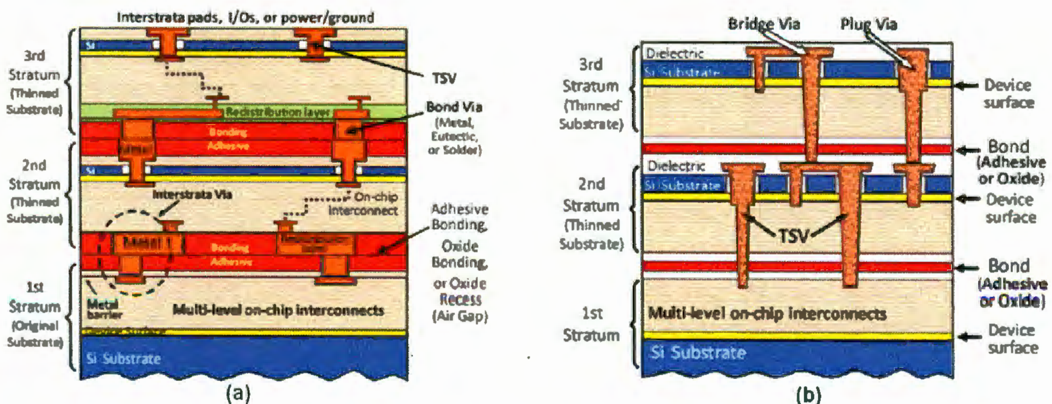


Figure 6.1: TSVs in 3-D ICs using (a) via-first, and (b) via-last methods. [107]

SiO_2 or other dielectrics, a liner or barrier layer (made of titanium, tantalum, TiN, or TaN), and via metal filling (made of copper, tungsten, or highly doped polysilicon) [107]. After the formation of the metal filled holes, the wafers are thinned and bonded together. The thinning of wafers is done by grinding and chemical mechanical polishing (CMP) [9]. In the via-last process, TSVs are formed after the wafers are aligned and bonded, and the top wafer is thinned. Similar to the via-first process, TSVs formed by the via-last method consist of an electrical isolation layer, a liner, and via metal fill [107]. The wafers are attached either by adhesive-to-adhesive bonding or oxide-to-oxide bonding as shown in Figure 6.1.

In the via-first technology, TSVs only pass through the thinned Si-substrate and the rest of the connection is carried through local interconnects and bond vias (vias made at the interface of the bonding layer) as shown in Figure 6.1(a). In via-last technology, TSVs pass through the thinned Si-substrate as well as all metal layers (because wafers are already aligned and bonded) as shown in Figure 6.1(b). As a result, the TSV height in the via-first technique is shorter than those formed using the via-last technique. To mechanically support the long TSVs formed using the via-last technique, the diameter of the TSVs are kept longer than the diameter of TSVs formed using the via-first technique [107].

We discussed the serious 3-D IC yield loss issue posed by TSV failure (see sub-section 1.5.6). For a quick recap TSVs suffer from thermo-mechanical stress caused by a difference in the Coefficient of Thermal Expansion (CTE) of copper TSVs and the surrounding dielectric. As a result, for a given temperature, copper expands more than its surrounding dielectric, resulting in thermo-mechanical stress. In this chapter we will focus more on the

TSV induced yield issue. Prior to that, let us look into other alternative via technologies that are available for vertical interconnections in 3-D ICs.

6.2 ALTERNATIVE VIA TECHNOLOGIES: WIRELESS VIAS

In addition to the through silicon vias, other alternative via technologies such as AC coupled interconnects (ACCI) have been proposed by [82],[83],[84]. For a full swing digital signal, its edges carry the digital information and its DC component carries no information [82]. An ACCI transmits digital information on the edges of a signal. These ACCI are circuit based solutions in which a wireless link across multiple device layers is established through transmitter (Tx) and receiver (Rx) circuits. Two types of ACCIs have been proposed by researchers as follows:

- **Capacitive ACCI:** In this topology, the transmitter and receiver circuits communicate with each other using a capacitive interface as shown in Figure 6.2(a). A voltage-mode driver transmits a signal which is converted into voltage pulses after passing through the coupling capacitor. The voltage pulses are reshaped to a full swing digital signal by the receivers.

- **Inductive ACCI:** In this topology, the transmitter and receiver circuits are interfaced by two spiral inductors which are formed across the two device layers as shown in Figure 6.2(b). These two spiral inductors construct a transformer to communicate with the Tx and Rx circuits. In an inductive ACCI, a current mode driver transmits a signal which is converted into current pulses after passing through the inductor pair. The current pulses are reshaped to a full swing digital signal by the receivers.

Since wireless vias are circuit based solutions, they do not suffer from thermo-mechanical stress which makes them a promising alternative to TSVs. Capacitive coupling based vias can only be used in two layers (face-to-face) and they are prone to low frequency noise. However, inductive vias are immune to low frequency noise and can be extended to more than two layers but they consume more power [82] (*14.5mW per via for 5Gbps data transfer at 180nm technology node*) compared to capacitive vias and TSVs. Both capacitive and inductive vias require additional area to implement Transmitter (Tx) and Receiver (Rx) circuits. A single wireless via needs additional Tx and Rx circuits in two device layers. The additional area overhead estimation will be presented in Chapter 7. The capacitive and inductive ACCI use the same type of Tx and Rx circuits and the coupling capacitors/inductors are formed in the metallization layer. Thus both capacitive and inductive ACCI occupy the same amount of Si area. Circuit schematics of Tx and Rx circuits for an inductive ACCI are shown in Figure 6.3. As a proof of concept, researchers have also fabricated a 3-D test chip using inductive ACCI and 0.35μm bulk-CMOS process as reported in [85]. They were able to achieve 2.5Gbps speed and performed for a 2^7 -1 pseudo random binary sequence with no errors for more than 2.5^{13} bits, after which they stopped the measurement

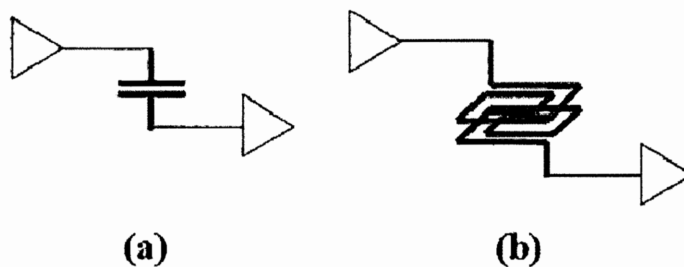
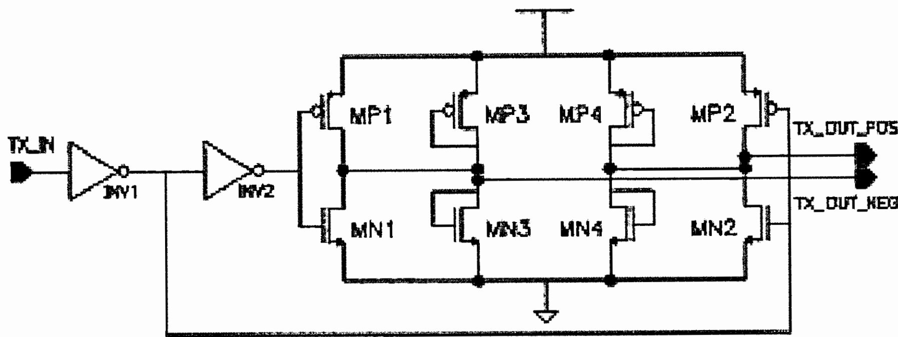
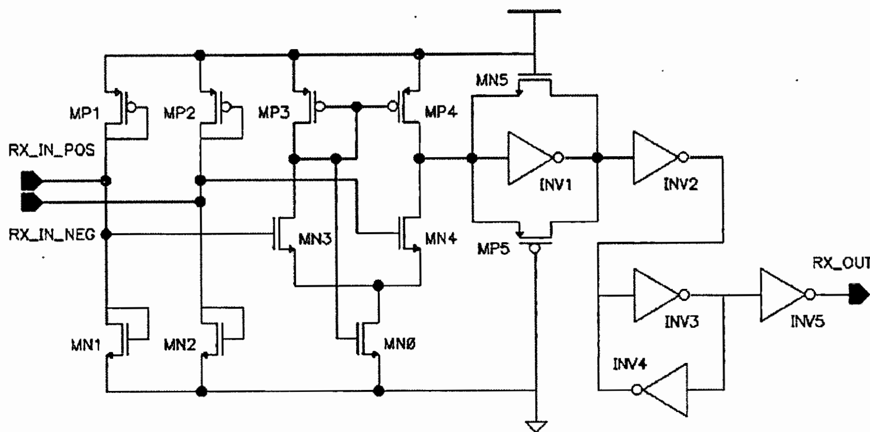


Figure 6.2: Concept of ACCI (a) Capacitive ACCI (b) Inductive ACCI. [82]



(a)



(b)

Figure 6.3: Schematic of (a) transmitter, and (b) receiver circuits of a wireless via using inductive coupling [82].

due to time constraints of the measurement. Furthermore, a new pulse based circuit technique using a 90 nm technology node to raise the aggregated data rate up to 1Tb/s, with high reliability (BER<10^{-13}) has been presented in [95]. It reduces the pulse width in the transmitter which in turn reduces the power consumption, because transmitter consumes power only when the pulse current flows. These inductive vias are based on standard CMOS techniques that do not require new process development. Unlike TSVs, they do not need ESD protection and experimental results show that inductive ACCI

exhibit good misalignment tolerance (+/- 3μm tolerance within 5% increase in power consumption) [95]. The two misaligned spiral inductors of inductive ACCI are still able to transmit signal using a weaker magnetic field than a perfectly aligned pair of inductors. However, the weak magnetic field is increased by increasing the current (hence the increase in power consumption). Thus good misalignment tolerance is achievable in inductive vias.

6.3 CARBON NANOTUBE BASED INDUCTORS FOR WIRELESS VIAS

Researchers [86],[87] have recently reported the following properties of Carbon Nanotube (CNT) based inductors:

- The radius (r) of a CNT is several nm compared to the radius of a copper wire which is about several μm. Therefore the magnetic field (*H*) induced by current (*I*) in a CNT is about one thousand times larger than that induced by a copper wire. $H \approx \frac{I}{2\pi r}$

- The relation between magnetic field *(H)* in the inductor and the inductance *(L)* of the inductor is $\int \frac{\mu_0 H^2}{2} dV \approx \frac{LI^2}{2}$. Thus large magnetic field results in the large inductance.

- CNTs can be bent with small curvature. Therefore an inductor made using CNTs has a smaller footprint than a copper inductor.

Furthermore, the large magnetic field induced by a small current in a CNT was experimentally measured using a magnetic force microscope and reported in [86],[87]. Since the on-chip CNT based inductor has a smaller footprint than a copper based inductor, denser packing of inductive vias is possible. In addition, recent research works have shown

promising results for use of CNTs as passive inductors in analog circuits such as LNA [88] and decoupling capacitors [89]. Thus in our work, we assume that high density inductors with smaller footprint areas are feasible in 3-D ICs.

6.3 YIELD AS A FUNCTION OF TSV FAILURE

We consider the yield problem in a 3-D system-on-chip in the presence of defects in through silicon vias. To operate the chip, we assume that all TSVs need to be fully functional. The objectives for solving this problem are as follows:

- Find new strategies such that all connections between device layers inside a 3-D chip can be established.
- Develop a model to estimate yield for a given TSV failure rate.

One solution to the 3-D yield problem (caused by thermo-mechanical stress) would be to replace all TSVs with inductive vias in a 3-D chip. However, the power penalty associated with them might be unacceptable. Therefore in this work we focus on minimizing power consumption in wireless vias while solving the yield problem in 3-D ICs. In the next chapter 3-D IC yield improvement will be discussed in detail.

Let us consider via redundancy within 3-D chips. It is a well known technique for yield enhancement. For a chip to be functional, all connections have to be satisfied. Thus we propose redundant via insertion such that every connection through TSVs can be established inside a chip. The objectives are to find new strategies for yield enhancement and to develop models to estimate yield for a given via failure rate.

A simple approach for yield enhancement would be to use redundant TSVs in parallel with the primary TSVs which is a common practice in traditional 2-D ICs. We studied the same technique for 3-D IC yield improvement by inserting a redundant TSV in parallel with a primary TSV and connecting them directly with a wire. Please note that in this configuration, there is no MUX inserted. We presented this study in [35],[36] and reported that for 10K primary TSVs and a 1% defect rate, the obtained functional yield was 35% which was far below the acceptable range of yield. We further analyze the number of redundant TSVs per primary TSV required in order to achieve an acceptable yield. Let us assume that there are "*r*" redundant TSVs connected in parallel with one primary TSV. The functional yield probability can be analytically expressed as:

$$Y_f = \left(1 - \left(P_d\right)^{r+1}\right)^{ViaCount} \tag{7.1}$$

Where P_d is the probability of defect ($0 < P_d < 1$), *ViaCount* is the total number of TSVs in a chip, and Y_f is the functional probability of yield ($0 < Y_f < 1$). Solving eqn (7.1) to find the

TABLE 7.1: NUMBER OF TSVS REQUIRED TO BE CONNECTED IN PARALLEL WITH EACH PRIMARY TSV IN 3-D CHIPS TO OBTAIN 90% FUNCTIONAL YIELD

Defect rate	ViaCount:10K	ViaCount: 20K	ViaCount:90K	ViaCount: 1M
	$r =$	$r =$	$r =$	$r =$
1	2	2	2	3
2	2	3	3	4
3	3	3	3	4
4	3	3	4	4
5	3	4	4	5
6	4	4	4	5
7	4	4	5	6
8	4	4	5	6
9	4	5	5	6
10	4	5	5	6

number of redundant TSVs required to be connected in parallel with each primary TSV for an acceptable/desired value of functional yield is obtained as follows:

$$r = \left\lceil \frac{\log\left(1-\left(Y_f\right)^{\frac{1}{ViaCount}}\right)}{\log\left(P_d\right)} - 1 \right\rceil \tag{7.2}$$

From Table 7.1, it is obvious that for a reasonable yield of 90%, 2 to 5 redundant vias per primary TSV are required for defect rate of 1 to 10%, and via count of 10K to 90K in a 3-D chip. However, even these redundant TSVs will also suffer from thermo-mechanical stress and will be prone to failure. They will also consume more routing resources in the vertical direction, and may cause congestion because electrical connections through TSVs pass through all metal layers as well as thinned substrates in the 3-D stack. Thus adding so many redundant vias will increase the thermo-mechanical stress, consume additional Si-

area, and will also create routing congestion in the vertical direction. Furthermore, putting two or more redundant vias very close to each other is not desired because the thermo-mechanical stress shifts from TSVs to the thinned substrate which can permanently damage the device layer [27],[28]. Thus it may not provide an efficient solution. Therefore we consider redundancy solutions that require less than 100% redundant vias.

To reduce the number of redundant vias, an alternative approach would be to make connections with redundant vias reconfigurable, i.e. a redundant via can be connected in place of a failed primary TSV in its neighborhood depending on the failure of a TSV. We can achieve the re-configurability using MUX-logic. To minimize the effect of thermo-mechanical stress, an alternative approach would be to use redundant wireless vias in addition to the primary TSVs. The advantages are that these wireless vias will not fail due to stress, and they will save routing resources in the vertical direction. We only consider inductive vias [82] because they can be used for more than two layers.

To minimize the impact of process variation (such as CMP variation), TSVs are preferred to be arranged uniformly. Thus we assume that: *a)* through silicon vias are uniformly distributed in rows and columns, *b)* the probabilities of defects in TSVs are uniform, *c)* wireless vias are 100% functional due to the absence of thermo-mechanical stresses, *d)* 3-D integration is achieved by die-to-die (DTD) and die-to-wafer (DTW) methods using known-good-dies only. For simplicity, we consider the insertion of redundant vias in two layer 3-D ICs only. We will show in Chapter 8 that it can be easily extended for more than two device layers.

7.1 REDUNDANCY LATTICES

In order to elaborate our proposed redundancy, we present different types of redundancy lattices that are used to construct various redundancy configurations in a device layer. Since the redundancy will be achieved using MUX-logic to re-route the failed TSVs, we propose lattices that use 2:1, 4:1, or 8:1 MUXes only because going beyond an 8:1 MUX might have an unacceptable area/delay penalty. The redundant via arrangement topologies have following different lattices:

- ***Quad Lattice (QL)***: a redundant via is located at the center of a square and each corner has primary TSVs as shown in Figure 7.1(a). Primary TSVs are those vias which are originally introduced during the 3-D IC design. The redundant via can be re-routed using a 4:1 MUX to connect any one of the four TSVs. Please note that the redundant via can either be a wireless via or a physical TSV.

- ***Octal Lattice (OL)***: A redundant via is inserted at the center, and eight primary TSVs are located around it (Figure 7.1(b)). An 8:1 MUX is used to connect any one of the TSVs..

- ***Dual Lattice (DL)***: Two primary TSVs are covered by one reconfigurable redundant via as shown in Figure 7.1(c). The redundant via re-routing is done using a 2:1 MUX.

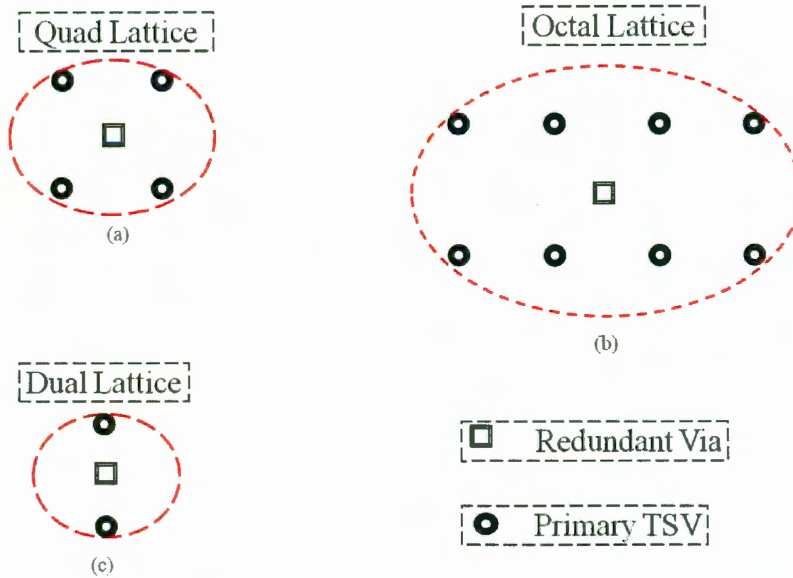


Figure 7.1: Proposed lattice structures for redundant via insertion in a device layer of a 3-D IC (a) Quad Lattice, (b) Octal Lattice, and (c) Dual Lattice. The redundant via can either be a wireless via or a physical TSV.

7.2 REDUNDANCY EVALUATION FACTORS

To characterize the redundant via configurations, we define the following evaluation factors:

- **Coverage factor (RT):** It is the number of primary TSVs covered by a redundant via. It also indicates the sizes of MUXes used. For example, in Figure 7.2, the redundant via of the shaded lattice at the center covers four primary TSVs within the lattice. Hence RT = 4.

- **Redundancy factor (TR):** It is a number of redundant vias covering a primary TSV. In Figure 7.2, the primary TSVs of the shaded lattice at the center are covered by two redundant TSVs. Thus TR = 2.

- **Lattice overlap factor (LO)**: It is the number of lattices that partially overlap with a particular lattice. In Figure 7.2, the shaded lattice at the center overlaps with four lattices (*shown by dashed circles*) and therefore LO = 4.

- **TSV overlap factor (OV):** A set of numbers that indicates the number of primary TSVs which are common between two overlapping lattices. In Figure 7.2, the shaded lattice at the center and any neighboring lattices (*shown by dashed circles*) have exactly one common primary TSV. Thus OV =1. However, OV can be more than one number in a given layout. For example, if we analyze the grey shaded lattice at the center of Figure 7.4, OV is 3 for its overlapping lattices in adjacent rows (i.e. neighboring lattices above and below the grey shaded lattice), but OV is 4 for its overlapping lattices in the same row (i.e. lattices to the left and right of the grey shaded lattice). Thus OV = {3, 4}.

The coverage factor (RT) indicates the sizes of MUXes used. If RT increases, then theoretically yield is expected to improve because it increases the reachability of a redundant TSV by primary TSVs. Similarly, increasing the redundancy factor (TR) should increase the yield because it enhances the probability of repairing a failed primary TSV. Furthermore, the lattice overlap factor LO should also increase the functional yield because it enhances the functional probability of all TSVs within a lattice. TSV overlap factor (OV) is used to distinguish the layout of different redundancy configurations in a device layer. Its effect on yield is already included in LO.

7.3 REDUNDANCY CONFIGURATION IN A DEVICE LAYER

Using the lattice structures introduced in section 7.1, we evaluate the different arrangements of primary and redundant vias in a device layer of 3-D chips for yield improvement. These configurations are divided into two categories that will be explained in sub-sections 7.3.1 and 7.3.2 respectively.

7.3.1 Wireless Via Redundancy Configuration

- ***Quad Wireless Plus Configuration (QWP):*** In this case, RT = 4, TR = 2, LO = 4, and OV = 1. It is possible to repair all four failing TSVs within a lattice in this configuration. A quad wireless plus configuration is shown in Figure 7.2, where all edges are covered by regular wireless vias while the remaining rows are covered by wireless vias in alternate columns. This configuration makes sure that each primary TSV, excluding the corner ones, can be re-routed by at least two different wireless vias. In this configuration, each lattice interacts with four neighboring lattices as shown by the shaded lattice in Figure. 7.2. Excluding the lattices at the edges, only one TSV is shared with any neighboring lattices in this configuration. Due to the redundant vias in *alternate* columns, there are vacant sites (*where wireless vias could be inserted*) as shown in Figure 7.2.

- ***Octal Wireless* (OW*) Configuration:*** Here, RT = 8, TR = 2, LO = 4, and OV = 2. The interaction of lattices in this configuration is shown in Figure 7.3. This configuration can be useful in saving area in case of a smaller defect rate.

- ***Octal Wireless Plus Configuration (OWP):*** Here, RT = 8, TR = 4, LO = 10, and OV = {3, 4}. The interaction of an *Octal* Lattice with its neighboring lattices is shown in Figure 7.4. Furthermore, there are vacant sites between any two consecutive redundant vias in a row as shown in Figure 7.4. Please note that if all eight TSVs within a lattice are failing, it is still possible to repair all of them simultaneously.

7.3.2 Physical Via Redundancy Configuration

- ***Dual TSV (DT) Configuration:*** Here, RT = 2, TR = 2, LO = 2, and OV = 1. Two primary TSVs are covered by one redundant TSV as shown by the shaded lattice in Figure 7.5.

- ***Quad TSV Complete (QTC) Configuration:*** Here, RT = 4, TR = 4, LO = 8, and OV = 2. It is constructed using redundant TSVs and *Quad Lattices*. The layout in a device layer is similar to Quad Wireless Plus *except that redundant TSVs are inserted in consecutive columns instead of alternate columns in each row, i.e. they are also inserted on the vacant sites between any two redundant vias in a row*. The examples of vacant sites are been shown in Figure 7.2.

- ***Octal TSV Plus (OTP) Configuration:*** In this case, RT = 8, TR = 4, LO = 10, and OV = {3, 4}.This is similar to the *Octal Wireless Plus* configuration (see Figure 7.4), except that its redundant wireless vias are replaced by redundant TSVs.

- ***Octal TSV Complete (OTC) Configuration:*** Here, RT = 8, TR = 8, LO = 20, and OV = {3, 4, 6}. It uses Octal Lattices with redundant TSVs. Its layout is similar to the Octal Wireless Plus configuration (Please see Figure 7.4) *except that redundant TSVs are inserted in consecutive columns instead of alternate columns in each row, i.e. they are also inserted on the vacant sites between any two redundant vias in a row.* The examples of vacant sites are shown in Figure 7.4. For a lattice in a given row, there are seven overlapping lattices (*including the redundant vias that can be placed on all vacant sites*) in each adjacent row *above* and *below* it, and six overlapping lattices in the same row. Thus LO = 2 x 7 + 6 = 20.

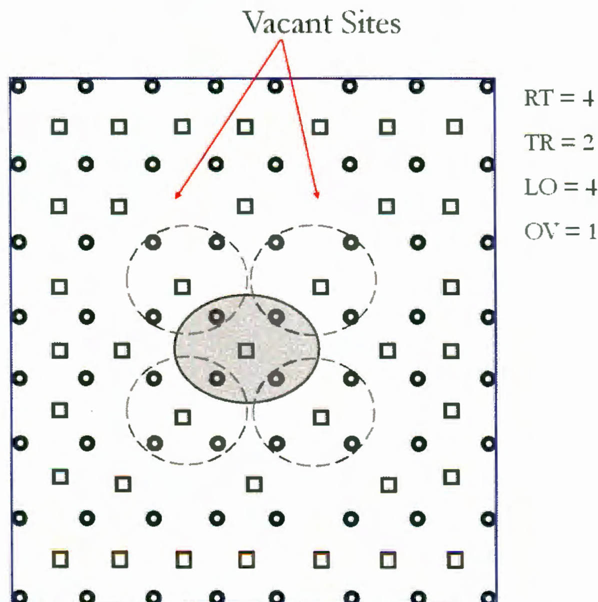


Figure 7.2: Quad Wireless Plus (QWP) configuration. If all the four TSVs within the shaded quad lattice fail, then it can be repaired by the neighboring wireless vias.

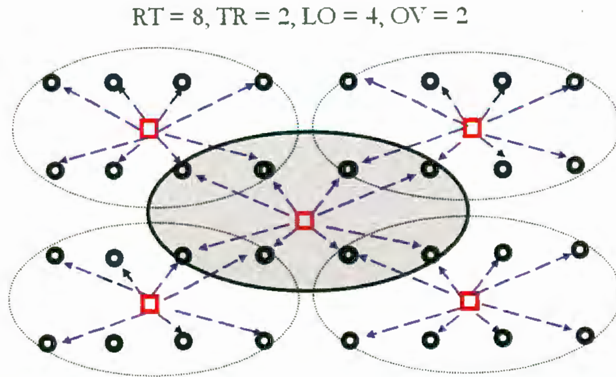


Figure 7.3: Interaction of an Octal Lattice (shaded region at the center) with its neighborhood lattices in an Octal Wireless* Configuration.

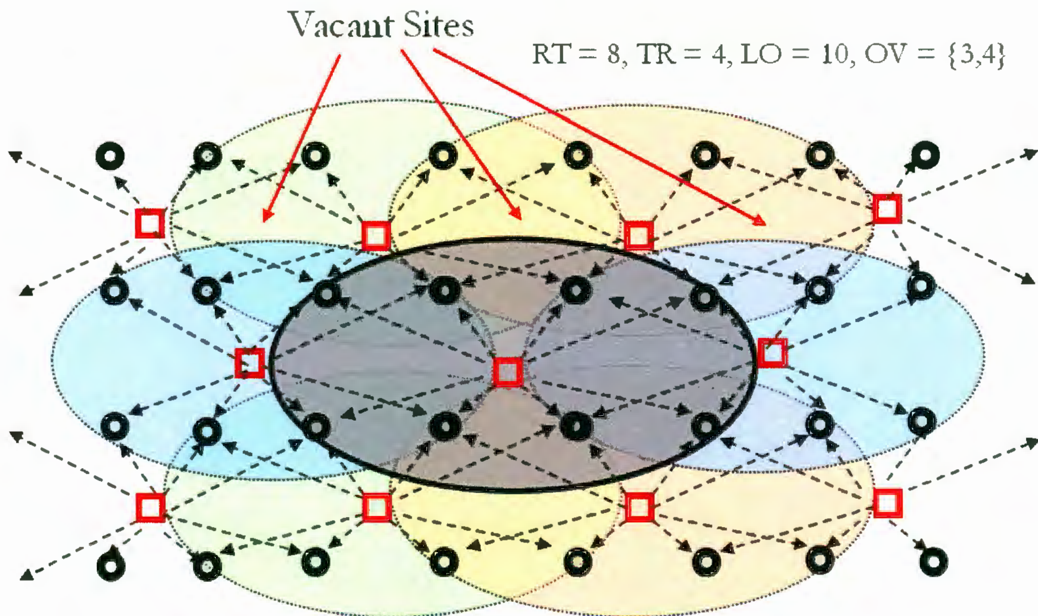


Figure 7.4: Interaction of an Octal Lattice (grey shaded region at the center) with its neighboring lattices in an Octal Wireless Plus configuration.

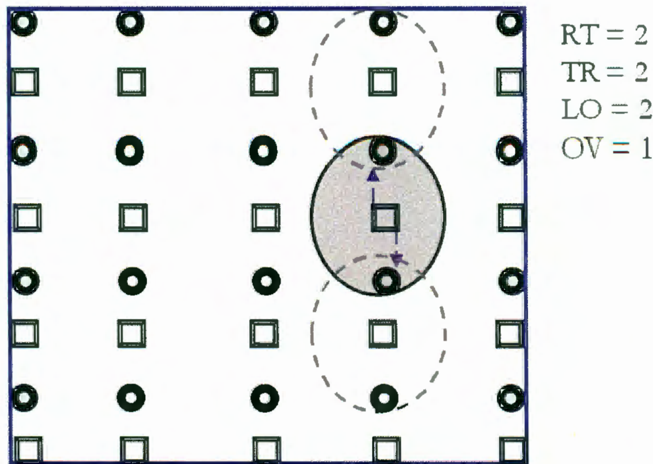


Figure 7.5: Interaction of a Dual (grey shaded) lattice with the neighboring lattices (encircled by dotted lines) in a Dual TSV redundancy configuration.

7.4 YIELD ESTIMATION BY MONTE CARLO SIMULATION

We implemented the proposed redundancy configurations for Monte Carlo simulations in C++/STL and performed simulations by varying the defect rate. We used 100K chips and each chip contained 10K primary TSVs. A chip is treated as failed when a primary TSV fails and its connection cannot be repaired by a redundant via. Defects in TSVs (*both primary and redundant*) are randomly inserted based on the given percentage of defect rate. Wireless vias are assumed to be defect free due to their immunity to thermo-mechanical stress. We used 1 – 10% defects in TSVs for our experiments.

A comparison of 3D IC yields obtained by different redundancy configurations that use redundant TSVs, is shown in Table 7.2. From the table, it can be observed that *Octal TSV Complete* configuration produces the best yield because it has the highest RT=8, TR=8, and LO=20 factors compared to all other redundant TSV configurations. Other redundancy

TABLE 7.2: MONTE CARLO YIELD RESULTS FOR REDUNDANT TSV CONFIGURATIONS

Defect Rate (%)	TSV Redundancy Configuration			
	Octal TSV Complete RT=8,TR=8, LO=20, OV={3,4,6}	Octal TSV Plus RT=8,TR=4, LO=10, OV={3,4}	Quad TSV Complete RT=4,TR=4, LO=8, OV=2	Dual TSV [35] RT=2,TR=2, LO=2, OV=1
1	100	100	97.7	97
2	100	96.5	97.1	84
3	100	84	90	61.1
4	100	67.4	72.1	36.2
5	100	37.8	66.3	15.2
6	100	30.4	45	-------
7	100	7.17	30.9	-------
8	97	6.43	10	-------
9	95.8	2.69	0	-------
10	94.3	2.48	0	-------

TABLE 7.3: MONTE CARLO YIELD RESULTS FOR WIRELESS VIA REDUNDANCY

Defect Rate (%)	Wireless Via Redundancy Configurations			
	Octal Wireless* RT=8,TR=2, LO=4, OV=2	Octal Wireless Plus RT=8,TR=4, LO=10,OV={3,4}	Quad Wireless [36] RT=4,TR=2, LO=2, OV=2	Quad Wireless Plus RT=4,TR=2, LO=4, OV=1
1	99.5	100	100	100
2	96	100	99.8	100
3	96.1	100	99.1	100
4	67.5	100	97.5	99.9
5	43.2	100	94.6	99.8
6	20.7	99	89.6	99.4
7	6.3	98.2	82.2	98.7
8	1.1	97.3	72.7	97.6
9	0	97.1	60.3	95.7
10	0	94.7	46.5	92.6

configurations in Table 7.2 produce an acceptable yield when the defect rate is less than 2%. We suspect that beyond the 2% defect rate, the functional probability of lattices

decrease significantly.

Table 7.3 shows the Monte-Carlo simulation results for the proposed wireless via redundancy configurations. It can be observed that *Octal Wireless Plus* and *Quad Wireless Plus* produce better yield due to higher RT, TR and LO factors compared to the rest of the redundant wireless via configurations. Thus it agrees with our theoretical explanation (please see Section 7.2) of the effects of these evaluation factors on the functional yield.

For each configuration (of Tables 7.2 and 7.3), the *TR:RT* ratio indicates the number of redundant vias as a fraction of the total number of primary TSVs. When this ratio increases, functional yield increases. If the ratio approaches to 1, it indicates that the number of redundant vias is equal to the number of primary TSVs. When the ratio remains the same but LO increases, then yield increases because it increases the functional probability of a redundancy lattice, i.e. the functional probability of all the primary TSVs within a lattice is increased. Similarly, if the ratio remains the same but TR and RT increase simultaneously then yield increases. The TSV overlap factor (OV) is used to distinguish the layout of redundancy configurations in a device layer. Its effect on the functional probability of a lattice is already included in LO.

Based on the results from Tables 7.2 and 7.3, we will focus our study on the most our promising redundancy configurations (*Octal TSV complete*, *Octal Wireless Plus*, and *Quad Wireless Plus*) that produce better yield. We study the scalability of these configurations by increasing the TSV count (i.e. problem size) in a design. Table 7.4 shows the comparison of yield obtained by these three redundancy configurations for TSV counts of 20K, 90K and

1M. It can be observed that with increasing TSV count, yield decreases. However, this decrease is not very steep (except for Quad Wireless Plus for 1M TSVs). Thus these configurations can be used in complex designs with large numbers of TSVs of future 3D ICs. Please note that Quad Wireless Plus and Octal Wireless Plus only use 50% redundant vias compared to Octal TSV Complete which uses 100% redundant TSVs. Quad Wireless Plus has added advantages due to the smaller 4:1 MUX delay compared to Octal Wireless Plus, and Octal TSV complete that use 8:1 MUXes. In the next section we will compute the cost of redundancy in terms of area, delay and power tradeoffs.

TABLE 7.4: COMPARISON OF 3-D IC YIELD OBTAINED BY MONTE-CARLO SIMULATION WITH INCREASING VIA COUNT AND USING THE MOST PROMISING VIA REDUNDANCY CONFIGURATIONS

Defect Rate (%)	Octal TSV Complete RT=8,TR=8, LO=20,OV={3,4,6}			Quad Wireless Plus RT=4,TR=2,LO=4, OV=1			Octal Wireless Plus RT=8,TR=4,LO=10, OV={3,4}		
	TSV count 20K	TSV count 90K	TSV count 1M	TSV count 20K	TSV count 90K	TSV count 1M	TSV count 20K	TSV count 90K	TSV count 1M
1	100	100	100	100	100	100	100	100	100
2	100	100	100	100	100	99.8	100	100	100
3	100	100	100	100	99.8	98.6	100	99.9	99.9
4	100	100	100	99.9	99.4	94	99.9	99.8	99.5
5	100	100	99.9	99.6	99.1	82.3	99.8	99.6	98.9
6	100	99.9	99.7	98.9	95.4	60.4	99.4	99.1	97.5
7	99.8	99.8	99.4	97.5	90	32.4	99.1	98.4	95.4
8	99.7	99.4	98.7	95.3	81	10.6	98.4	97.4	92
9	99.5	99.2	97.7	91.5	68.1	1.4	97.6	95.4	87
10	99.4	99	96.2	85.5	50.8	0	96.5	93.2	80.6

7.5 MODELING AREA, DELAY AND POWER OF REDUNDANT VIAS

For the via configurations proposed in the earlier section, we estimate the penalties in area, delay, and power due to redundancy. We first compute these tradeoffs for each type of redundancy lattice. The actual tradeoff can be obtained by simply multiplying cost values per lattice by the number of lattices present in a design for a given redundancy configuration.

7.5.1 Area Tradeoff

We consider 2 input NAND gates (NAND2) as our basic logic elements in our analysis. The number of NAND gates defines a metric for estimating the area penalty. Each wireless via requires transmitter (Tx) and receiver (Rx) circuitry. We estimate the number of NAND gates using the transistor count in transmitter and receiver circuitry which was earlier shown in Figure 6.3. If a wireless via is chosen for covering a failed physical via, the transistor count in two multiplexers (one for Tx and another for Rx) also needs to be determined. Since the Tx and Rx are standard CMOS based circuits, the number of NAND gate estimates for them are fairly reasonable. Similarly, MUX circuits based on NAND gates are a good estimation. Please refer to Table 7.5 for the number of NAND gates/area needed for each of the configurations. This number determines the area penalty suffered during re-routing for a failed physical via. The exact area can be calculated easily if the technology node is known. For a redundant TSV we have assumed 5x5 μm^2 including the area of the contact pad while calculating the area. Please note that there will be two contact pads required for one TSV (one pad for each device layer). Furthermore, we add the area of the MUX used in a particular redundancy lattice.

TABLE 7.5: AREA PENALTY OF REDUNDANT LATTICES IN TERMS OF THE EQUIVALENT AREA OF A TWO-INPUT NAND GATE

	Redundancy lattices			
	Quad Wireless	Octal Wireless*	Octal TSV	Dual TSV
# of NAND2	20	36	28	4
Area at 180nm (μm^2)	200	360	330	90

TABLE 7.6: DELAY PENALTY AND IT'S SCALING WITH TECHNOLOGY NODE

Logic element	Delay in **ps** for different technology nodes				
	180 nm	**90 nm**	**65 nm**	**45 nm**	**32 nm**
NAND2	26	18.4	13	9.2	6.5
2:1 MUX	123	87.0	61.51	43.51	30.8
4:1 MUX	214	151.3	107	75.7	53.5
8:1 MUX	337.5	238.3	168.5	119.2	84.3

7.5.2 Delay Tradeoff

We calculate the delay penalty by estimating the delay occurred due to redundant MUXes and adding it to the path delay. We use the TSMC 180nm standard cell library [91] for the delay estimation. The library provides delay values for each input pin transition. We averaged these values for each input pin and report them in Table 7.6. The values are scaled for lower technology nodes using the constant field scaling factor $1/\sqrt{S}$ [92]. After re-routing a failed physical via, only one path through these multiplexers would be active which accrues against path delay. The delay for an 8:1 MUX is not reported in the library. We calculate it for the octal wireless configuration obtained using two 4:1 MUXes cascaded

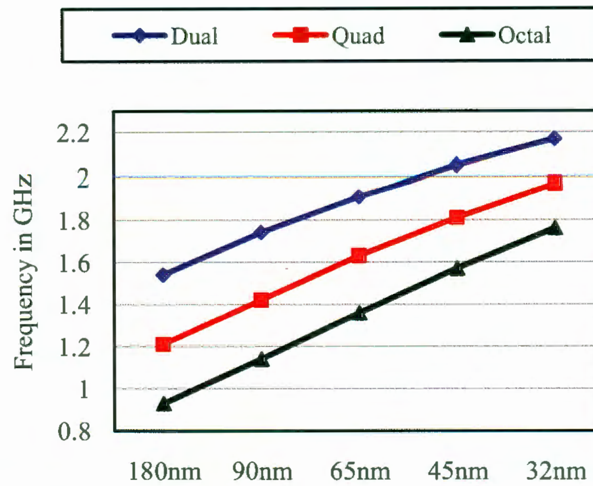


Figure 7.6: Performance Reduction due to via re-routing through MUX logic. The chip's target frequency is 2.5 GHz (ideal case).

with a 2:1 MUX. Also, please note that due to the presence of MUX delay there is clock stretching which reduces the overall performance.

We compare the performance of a 3D chip achieved using different redundancy configurations with respect to a *2.5GHz ideal case target frequency*. From Figure 7.6, it can be seen that the performance penalty decreases as we move to lower technology nodes. At 180nm, there is drastic reduction in performance compared to the target frequency, while for current technology nodes such as 45nm and 32nm, the performance reduction can be acceptable as tradeoff (i.e. since our approach increases the number of functional chips, it is usually better to have slower but functional chips than a large number of failed chips). At 32nm, there is a 13% reduction for Dual lattice, 17% for Quad lattices (*i.e. all the lattices that use 4:1 MUX*) and 29% for Octal lattices (*i.e. all the lattices that use 8:1 MUX*). Octal lattices suffer the maximum delay penalty due to the presence of 8:1 MUX. Because of the large slack, it could be a better choice for slower chips. MUX re-routing involves extra wiring

which also causes additional delay which can be calculated by the formula rcl^2, where, r and c are resistance and capacitance per unit length of a wire, and l is length of the wire.

TABLE 7.7: POWER PENALTY AND ITS SCALING WITH TECHNOLOGY NODE

Logic element	Delay in μW for different technology nodes				
NAND2	28	14	7	3	2
2:1 MUX	55	28	14	7	3
4:1 MUX	110	55	28	14	7
8:1 MUX	166	83	41	21	10

7.5.3 Power Tradeoff

The TSMC 180nm standard cell library specifies values for power in μW/MHz. We assume the operating frequency to be 2.5GHz from [82] where data rates up to 5Gbps can be obtained. Power data for lower technology nodes is obtained by using constant field scaling with a scaling factor of $1/S^2$ [92] Power values are shown in Table 7.7. It can be seen that power dissipation drastically reduces as we move to lower technology nodes. For simplicity, we assume that leakage power is negligible. Please note that the power dissipation in a wireless via is 14.5 mW at 2.5GHz [82] at the 180nm technology node whereas the largest 8:1 MUX consumes only 166 μW. Thus the power consumption in MUX is negligible compared to a wireless via. Please note that even if leakage power is considered, the total power consumption in the MUX will still be negligible compared to a wireless via.

7.6 EFFECT OF REDUNDANCY ON PARAMETRIC YIELD

The delay penalty due to additional Tx/Rx circuits, MUX-logic and signal re-routing may degrade the performance of a 3D chip. In this section, we analyze the impact of redundancy on critical paths and overall chip performance in a bin of 3D chips. We define two types of chips as follows:

- ***Fast Chip*** : A chip without any redundant vias on its critical paths. Here, 3-D chips will operate at its designed speed (assuming other variability issues are taken care of).

- ***Slow Chip*** : A chip with a redundant via on at least one of its critical paths.

We assume that a critical path which spans across two device layers passes through exactly one primary TSV as shown in Figure 7.7. This assumption can be realized using a min-cut partitioning based 3-D placement tool [3]. The placement algorithm minimizes the cut-size of a net spanning across two device layers during placement which is equivalent to

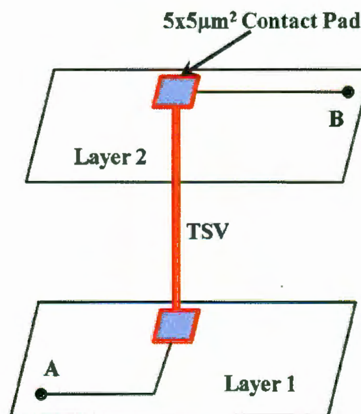


Figure 7.7: A critical path spanning across two device layer using a TSV. We assume that a critical path crosses only once through a TSV.

minimizing the number of TSVs. Our goal is to estimate the number of fast/slow chips. We first use a statistical 3-D wirelength distribution model to estimate the number of global wires in a chip that will be described in the next sub-section.

7.6.1 Estimation of the Total Number of Global Wires

Let us consider $\boldsymbol{N}$ logic gates which are arranged as a uniform 2-D array inside a 2-D chip. It becomes possible to estimate the number of wires for a given length using a wirelength distribution model [71] as follows:

$$\begin{aligned} i(\ell,N,k,p) &= \frac{\alpha k}{2}\Gamma\left[\frac{\ell^3}{3} - 2\sqrt{N}\ell^2 + 2N\ell\right]\ell^{2p-4}; \quad 1 \le \ell \le \sqrt{N} \\ &= \frac{\alpha k}{6}\Gamma\left(2\sqrt{N}-\ell\right)^3 \ell^{2p-4}; \quad \sqrt{N} \le \ell < 2\sqrt{N} \end{aligned} \tag{7.1}$$

where ℓ is the interconnect length in gate-pitch units and α is a function of average fanout (*f.o.*) as shown below:

$$\alpha = \frac{f.o.}{1+f.o.} \tag{7.2}$$

and Γ is given by:

$$\Gamma = \frac{2N(1-N^{p-1})}{\left(-N^p \dfrac{1+2p-2^{2p-1}}{p(p-1)(2p-1)(2p-3)} - \dfrac{1}{6p} + \dfrac{2\sqrt{N}}{2p-1} - \dfrac{N}{p-1}\right)} \tag{7.3}$$

At p = 0.5, Γ becomes indeterminate of the form 0/0 and its value can be determined using L'Hopital Rule [71].

Now, let us consider that the same 2-D chip is designed as an *m*-layer 3-D chip. We obtain the wirelength distribution of each layer and number of TSVs from [1] as:

$$k_{\text{eff}} = km^{p-1} \tag{7.4}$$

$$i_{3D}(\ell) = m \times i(\ell, N/m, k_{eff}, p) \tag{7.5}$$

$$\#TSV = m \times k(1-m^{p-1})(N/m)^{p} \tag{7.6}$$

Similar to [93] we define the lengths of *local*, *semi-global* and *global* wires as follows:

$$\ell_{local} = 1 \le \ell < 0.7\sqrt{N/m} \tag{7.7}$$

$$\ell_{semi} = 0.7 \le \ell < 1.2\sqrt{N/m} \tag{7.8}$$

$$\ell_{global} = 1.2\sqrt{N} \le \ell < 2\sqrt{N/m} \tag{7.9}$$

Using eqn (7.5) and eqn (7.9), we can estimate the total number of global wires inside a 3-D chip by integrating (7.5) over the range of ℓ_{global}. Please note that out of the total number of global wires, many wires will only span within a particular device layer and others will span across two device layers using TSVs.

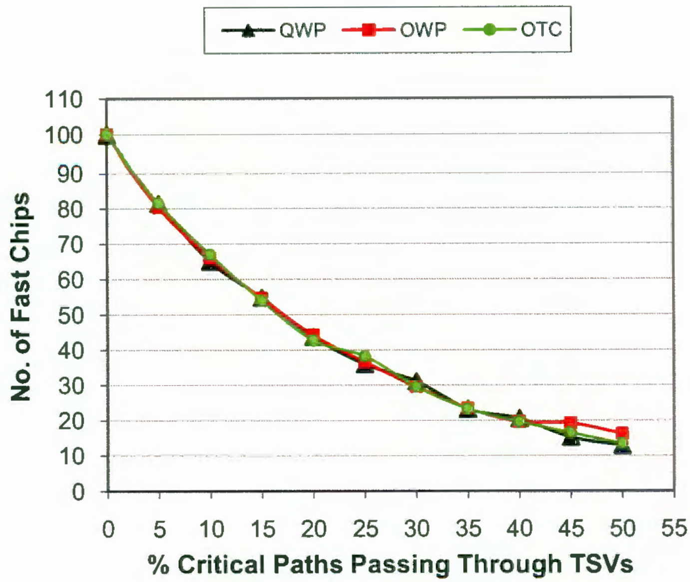
7.6.2 Estimation of the Total Number of Fast Chips

To find the number of fast chips, we performed Monte Carlo simulation on a set of 100K 3-D chips. Each 3-D chip contained 5 million gates in 2 device layers. We used Rent's

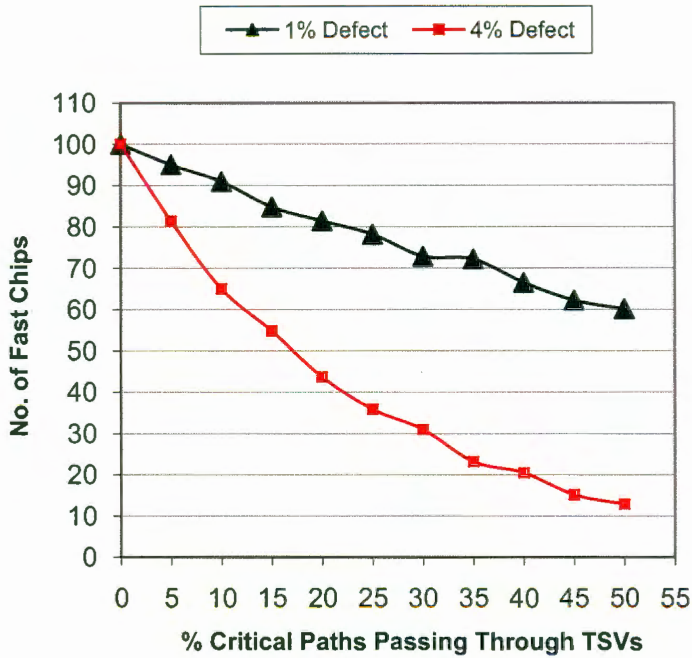
parameters $k = 1.4$ and $p = 0.63$ from [90]. In addition, we assumed that the circuit inside the chip was built using 2-input NAND gates for the uniform logic gate arrays inside each device layer [32]. The average fanout was chosen as 3.0 and the total number of primary TSVs was calculated using eqn (7.6).

The defect rate in TSVs was varied from 1 – 10% and the number of critical paths passing through TSVs was varied between 0 - 50% of the total critical paths. Figure 7.8(a) shows the number of fast chips at a 4% defect rate for Quad Wireless Plus *(QWP)*, Octal Wireless Plus *(OWP)*, and Octal TSV Complete *(OTC)* configurations. It can be observed that the number of fast chips obtained by these three configurations is approximately the same. This is because the functional yields are approximately the same for these configurations for the given problem size. Figure 7.8(b) shows how the number of fast chips varies for different defect rates. Please note that the fast chip count drops sharply with increasing defect rate and increasing number of critical paths passing through TSVs. Its analytical reasoning is given in Chapter 8 (please see section 8.5).

The area penalty in terms of number of NAND2 gates is 1.35% for the Quad Wireless, and 1.2% for the Octal Wireless configuration compared to the total gate count in a chip which is negligible. Since *Octal TSV complete* has 100% TSV redundancy, the area penalty is 2.24e+06μm^2 (including the 5x5μm^2 area for a single TSV in one layer) which is equivalent to 4.5% of total gate count at the 180nm technology node. Thus area penalty is negligible. In the next chapter, we will present analytical models to quickly analyze the yield for a given defect rate when yield numbers are needed as design parameters.



(a)



(b)

Figure 7.8: Comparison between the number of fast chips obtained (a) for 4% defect rate using Quad Wireless Plus (QWP), Octal Wireless Plus (OWP), and Octal TSV Complete (OTC) configurations (b) for 1% and 4% defect rates by Quad Wireless Plus (QWP) configuration.

In Chapter 7 we presented functional yield enhancement methodology based on a via redundancy technique and estimated functional and parametric yield for 3-D ICs. Monte Carlo simulation is an iterative process which computes yield for discrete input values based on the problem size (i.e. TSV count) and defect rate in TSVs. Due to the iterative nature of Monte Carlo simulations, it is time consuming for a large problem size. Furthermore the yield obtained from an MC simulation is just one discrete point in the yield solution space. In this chapter, we present analytical models for functional and parametric yields that eliminate the need for computationally expensive Monte Carlo simulations. We further provide an analytical model for the chip revenue. The analytical models quickly analyze the yield for a given defect rate when yield numbers are needed as design parameters. These yield numbers can be used in yield-aware physical design optimization processes such as floorplanning, placement and routing. Based on the Monte Carlo yield results from Chapter 7, we have chosen to derive analytical models for the three redundancy configurations that provide high functional yield for a large number of TSVs and a wide spectrum of defect rates in 3D designs. These redundancy models are *a)* Quad Wireless Plus configuration (RT = 4, TR=2, LO=4, OV=1), *b)* Octal Wireless Plus configuration (RT =8, TR=4, LO=10, OV={3,4}), and *c)* Octal TSV Complete configuration (RT=8, TR=8, LO=20, OV={3,4,6}). The functional yield is defined by the number of working chips represented as a percentage of the total number of chips in a bin. Thus functional yield = 100 **x** Number of working chips/Total number of chips in a bin.

8.1 NOMENCLATURE

We provide a list of variables that will be used for the derivation of analytical models:

P_d	:	Probability of a TSV to be defective ($0 \leq P_d \leq 1.0$)
P_w	:	Probability of a TSV to be functional
$P_{critical}$	:	Probability of a TSV to be on a critical path
n	:	Number of TSVs in a redundancy lattice
ViaCount	:	Total number of primary TSVs in a chip
ChipCount	:	Total number of 3D chips in a bin.
CritTSV	:	Number of global wires passing through TSVs
GlobalWire	:	Total number of global wires in a chip
Y_f^{QWP}	:	Functional yield of *Quad Wireless Plus* configuration in percentage
Y_f^{OWP}	:	Functional yield of *Octal Wireless Plus* configuration in percentage
Y_f^{OTC}	:	Functional yield of *Octal TSV Complete* configuration in percentage
P_F	:	Price of a fast chip

P_S : Price of a slow chip

m : Number of redundant vias which cover one primary TSV

${}^{n}C_{r}$: Number of possible ways of "r" TSVs being defective from "n" TSVs

$$= \frac{n!}{r!(n-r)!}$$

8.2 ANALYTICAL MODEL FOR QUAD WIRELESS PLUS (QWP) CONFIGURATION

To obtain the analytical model, we first focus on calculating the probability of a quad lattice (shown in Figure 7.1 in Chapter 7) to be functional. Please note that this probability is also dependent on how a lattice is interacting with its neighboring lattices (in terms of lattice overlaps). This lattice interaction for QWP is shown in Figure 7.2, and redundancy evaluation factors are RT = 4, TR=2, LO=4, OV=1. To keep the model simple, we first assume that *if the TSVs within a lattice fail, then the redundant wireless vias in the neighboring lattices are available for the repair.*

The probability of a TSV being *defective* is given by P_d which is dependent on the 3-D integration technology. Due to the unavailability of the statistical data from 3-D technology, we treat P_d as a variable for $0 < P_d \leq 0.1$, i.e. the same range of defects that was used for the Monte Carlo simulation in Chapter 7. Thus the probability of a TSV to be *functional* (i.e. P_w) equals to 1 - P_d. Since there are four primary TSVs in the redundancy lattice of the QWP configuration, n = 4.

Within a lattice, we first consider all the possible combinations of *failed* and *working* TSVs. Next we examine whether for each such combination of *failed* and *working* TSVs it is possible to repair all the failed TSVs by neighboring redundant vias. If a particular combination of *failed* and *working* vias within the lattice is repairable, then the *joint* probabilities of these vias are multiplied by the total number of ways that particular combination can be obtained within a lattice, and the result is added to the functional probability of the lattice. For example, if *one* out of four primary TSVs within a lattice of the QWP configuration has failed, it means that the remaining *three* TSVs within the lattice are working. Thus the joint probability of this combination of *failed* and *working* TSVs would be $(P_d)^1(P_w)^3$. Next the different number of ways for which one out of four primary TSVs in a lattice can fail is ${}^4c_1$. Thus the total probability of this particular combination of *failed* and *working* primary TSVs will be ${}^4c_1 \times (P_d)^1 \times (P_w)^3$. Similarly we consider all the possible combinations of failed and working TSVs with which the failed TSVs can be repaired. The sum of all these combinations gives the functional probability of a lattice. Please recall from Sub-section 7.3.1 that even if all the four primary TSVs of a lattice in QWP fail, it is still possible to repair all of them simultaneously. Therefore all combinations of failed and working TSVs are repairable, and they should be added in calculating the functional probability of the lattice. Thus this probability is calculated by $\sum_{r=0}^{n} {}^nc_r (P_d)^r (P_w)^{n-r}$.

Next we find the probability of all the TSVs in a chip to be working. Since the functional probability of a lattice covers the functional probability of n primary TSVs within the lattice, the functional probability of all the TSVs is calculated by the joint probability of $ViaCount/n$ lattices. Thus the expression for the functional yield of QWP *(represented as a*

percentage of the total number of chips in a bin) is given by eqn (8.1) in which the summation of a series term calculates the functional probability of a lattice, and the power term calculates the functional probability of all the TSVs in a 3-D chip.

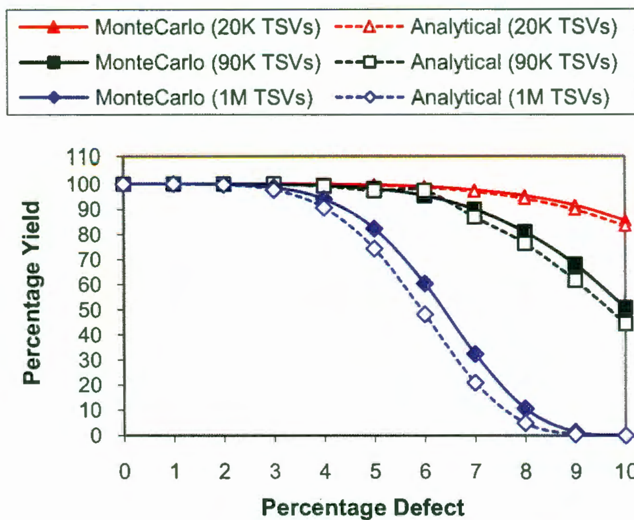
$$Y_f^{QWP} = 100 \times \left(\sum_{r=0}^{n} P_{rd} . {}^{n}c_r \right)^{\frac{ViaCount}{n}} \tag{8.1}$$

where $n = 4$ for *QWP*, and

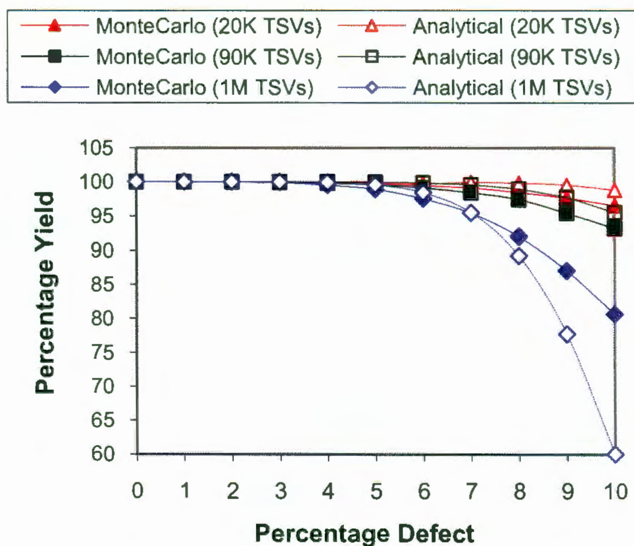
$$P_{rd} = \begin{cases} (P_d)^r (P_w)^{n-r} \left(2P_w - (P_w)^2\right); for \;\; r = \frac{n}{2}+1 \\ (P_d)^r (P_w)^{n-r} \qquad ; otherwise \end{cases} \tag{8.2}$$

Please note that in eqn. (8.2), the $2P_w - (P_w)^2$ term has been introduced to minimize the error because *a)* a primary TSV can be part of two lattices and it can be adjusted by the *logical* OR probability that was introduced in eqn (8.2), and *b)* Our primary assumption was that *if the TSVs within a lattice fail, then the redundant wireless vias in the neighboring lattices are available for the repair.* However, it might be possible that a redundant wireless via might have been used to repair TSVs of another redundancy lattice in a device layer.

A comparison of analytical and Monte Carlo results for functional yield with 20K, 90K and 1M TSVs in a chip is shown in Figure 8.1(a). Please observe that the analytical model matches quite closely with the Monte-Carlo simulation results. We have used ten discrete defect rate points in Figure 8.1(a) to show the yield trend for each of the problem sizes (i.e. TSV counts). However, in real life the problem size as well as the defect rate may not be a



(a)



(b)

Figure 8.1: Comparison of Analytical model results with Monte Carlo simulation results for yield obtained using (a) Quad Wireless Plus configuration, and (b) Octal Wireless Plus Configuration for different numbers of TSVs in 3-D ICs.

discrete point. Thus the trend may shift depending on the problem size and the defect rate. For each of the ten discrete points of defect rate, Monte Carlo simulations were performed independently (i.e. MC simulations were run ten times) which was very time consuming. In

contrast, for the analytical yield results, the analytical model of eqn (8.1) was used to obtain the yield for ten discrete defect rates which was a simple mathematical computation. Thus the analytical model provides fast yield estimation.

8.3 ANALYTICAL MODEL FOR OCTAL WIRELESS PLUS (OWP) CONFIGURATION

In this configuration, n = 8. The method of deriving the analytical model for OWP (RT =8, TR=4, LO=10, OV={3,4}) as shown in Figure 7.4 is similar to the derivation method for the QWP configuration described in Section 8.2. Similar to QWP, all the combinations of *failed* and *working* TSVs are repairable in OWP configuration and therefore they should be added in calculating the functional probability of the lattice. Due to this similarity, the rest of the expression for Y_f^{OWP} will be the same as the expression in the analytical model for the *Quad Wireless Plus configuration* (i.e. Y_f^{QWP}) given by eqn (8.1) and eqn (8.2) in section 8.2. Please note that this similarity might not be present in all other redundancy configurations. Thus any combination of *failed* and *working* TSVs that cannot be repaired by redundant vias should not be included while calculating the functional probability of a redundancy lattice.

A comparison of analytical and Monte Carlo simulation results for the functional yield with 20K, 90K and 1M TSVs in 3D chips is shown in Figure 8.1(b). It can be observed that the analytical model matches quite closely with the Monte-Carlo results. However, for 1M TSVs, the error grows larger beyond an 8% defect rate. We suspect that this is due to exponential growth of the truncation error.

8.4 ANALYTICAL MODEL FOR OCTAL TSV COMPLETE (OTC) CONFIGURATION

As described in Sub-section 7.3.2, the primary and redundant TSVs are placed uniformly in the OTC configuration, which has evaluation factors RT=8, TR=8, LO=20, OV={3,4,6}. For the derivation of the analytical model for OTC, we first calculate the probability of a primary TSV to be functional. For simplicity, we assume that if the *redundant* TSVs for a particular *primary* TSV are working, then they are available for repairing that particular *primary* TSV.

In OTC, a primary TSV is covered by eight redundant TSVs. Let us call this, "**m**" and therefore **m** = 8. For a primary TSV, its probability of working is P_w. The probability of its repair (in case it is failing) is calculated by the join probability of *the primary TSV* to be failing (P_d) **and** the probability of **at least** one out of *m* redundant TSVs to be working. Please note that the term containing the *summation of a series* in eqn. (8.3) calculates the joint probability. Finally we calculate the probability of *all primary TSVs* to be working by incorporating the power term in eqn. (8.3). The final equation for calculating the functional yield of the *Octal TSV Complete* configuration is:

$$Y_f^{OTC} = 100 \times \left(P_{w} + P_d \times \sum_{r=0}^{m-1} (P_d)^r (P_w)^{m-r} \left({}^m c_r\right) \right)^{ViaCount} \quad (8.3)$$

We compared the analytical model's results with the Monte Carlo simulation results for 1 - 10% defect rates. The analytical results are within 1 - 3% error with Monte Carlo results for

20K to 1M TSVs. Thus the analytical model's results match closely with the Monte Carlo simulation results.

8.5 ANALYTICAL MODEL FOR FAST/SLOW CHIPS

In this section, we derive the analytical models for estimating the number of fast/slow chips in a bin of 3-D ICs. The assumptions for defining the fast and slow chips are given in Section 7.6 (in Chapter 7).

We first find the probability of a TSV to be on a critical path (i.e. $P_{critical}$) which is given by the ratio of the total number of TSVs that can be on critical paths (*CritTSV*) to the total number of TSVs in a chip. Here, *CritTSV* is equal to the percentage of the total number of global wires that span across different device layers (i.e. using TSVs), and *CritTSV* is calculated using the statistical 3-D wirelength distribution method explained in Sub-section 7.6.1. Thus the expression of $P_{critical}$ is given by eqn (8.4).

$$P_{critical} = CritTSV / ViaCount \tag{8.4}$$

The probability of a TSV to be *Functional* ***and*** *on Critical Path* = 1 – (the probability of a TSV to be *defective* **and** on a *Critical Path*) = 1 – ($P_d \, P_{critical}$). Please note that in this expression, "1" is the cumulative probability of *a)* the TSV to be functional and on a critical path, and *b)* the TSV to be defective and on a critical path.

Therefore, probability of **all TSVs** to be *Functional* ***and*** *on Critical Path* = $(1 - P_d P_{critical})^{ViaCount}$.

Fast Chips (*FC*) as a percentage of the total number of chips for a given redundancy configuration *RC* is calculated by eqn (8.5):

$$FC_{RC} = \left(Y_f^{RC}\right)\left(1 - P_d \times P_{critical}\right)^{ViaCount} \tag{8.5}$$

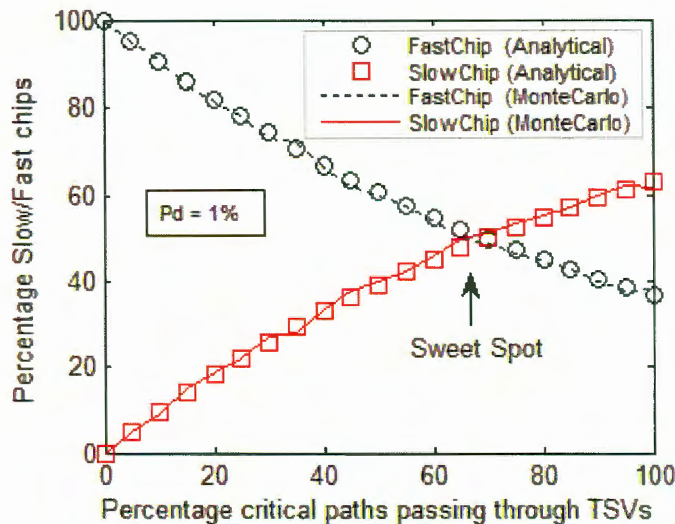
where $Y_f^{RC} \in \left\{Y_f^{QWP}, Y_f^{OWP}, Y_f^{OTC}\right\}$

Due to *ViaCount* as the power term in eqn. (8.5), the number of fast chips may drop exponentially with increasing P_d and/or $P_{critical}$. Furthermore, any increase in P_d will also decrease the functional yield Y_f^{RC} which is also a factor in eqn (8.5). Thus P_d has dual impact on the total number of fast chips.

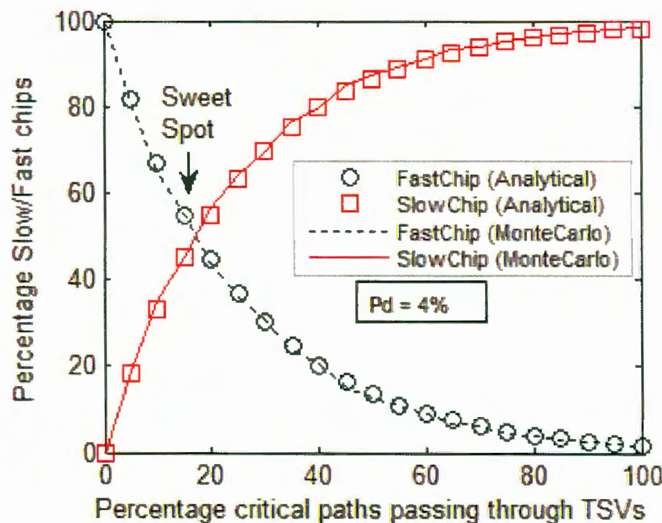
Similarly, Slow Chips (*SC*) as a percentage of the total number of chips for a given redundancy configuration *RC* would be as shown in eqn (8.6).

$$SC_{RC} = \left(Y_f^{RC}\right)\left(1 - \left(1 - P_d \times P_{critical}\right)^{ViaCount}\right) \tag{8.6}$$

Using our analytical model, we calculated the number of fast/slow chips for the input setup presented in Sub-section 7.5.2 in chapter 7 (i.e. 5M gate design). A comparison of Monte-Carlo and our analytical models' results for fast/slow chips (for 1% and 4% defect rates) obtained using the *Quad Wireless Plus* configuration is shown in Figure 8.2. It demonstrates that the analytical model matches closely with the Monte-Carlo results. We have observed similar comparisons for *Octal Wireless Plus* and *Octal TSV complete* configurations as well. Please notice that the *FC* and *SC* curves cross each other at a certain point in Figure 8.2.



(a)



(b)

Figure 8.2: Analytical and Monte Carlo simulation results for Fast/Slow chips for Quad Wireless Plus configuration for (a) 1% defect rate (b) 4% defect rate.

This intersection in Figure 8.2 is a sweet spot below which the number of fast chips produced will be larger than the number of slow chips for a given redundancy configuration. We define a variable P_{sweet} as shown in eqn (8.7).

$$P_{sweet} = \frac{1}{P_d}\left(1-(0.5)^{1/ViaCount}\right) \tag{8.7}$$

Next, we define a variable "*G*" which denotes the highest percentage of global wires that can pass through TSVs and still get a larger number of fast chips. *"G"* is given by eqn (8.8):

$$0 \le G \le \left(\frac{P_{sweet} \times ViaCount}{GlobalWire}\right) \times 100 \tag{8.8}$$

This model will allow a designer to quickly estimate the maximum number of global wires that can pass through TSVs in order to obtain a higher number of fast chips. It can also be incorporated in physical design tools such as floorplanning for 3-D ICs. The detailed description of the analytical models' use is given in Section 8.8. Please note that the sweet spot in Figure 8.2 shifts leftwards with increasing defect rate that will in turn produce fewer number of fast chips.

8.6 ANALYTICAL MODEL FOR CHIP REVENUE

In this sub-section, we present chip revenue estimations obtained from a bin of chips. We assume that the prices of fast and slow chips include the packaging, assembly and bonding costs in addition to design and fabrication costs of 3-D chips in our revenue model. The revenue model is a function of *ChipCount*, *ViaCount*, *defect rate*, *CritTSV*, and *redundancy configuration* and it can be obtained by eqn (8.9).

$$\text{Revenue} = \left(P_F.FC_{RC} + P_S.SC_{RC}\right) \times \frac{ChipCount}{100} \tag{8.9}$$

where P_F is the price of a fast chip and P_S is the price of a slow chip; $P_F > P_S$. The variables FC_{RC} and SC_{RC} are obtained from section 8.5 and they depend on the redundancy configuration. The total chip revenue model provides a tool for quick estimation of chip profitability.

8.7 EXTENSION OF ANALYTICAL YIELD MODELS OF TWO-LAYER 3-D CHIPS TO MULTI-LAYER 3-D CHIPS

The analytical yield models previously described in Section 8.2 to Section 8.6 were derived for 2 layer 3-D ICs only. In this section, we extend our previous analytical models of 2-layer 3-D ICs to analytical models of multi-layer 3-D ICs using the 3-D wirelength distribution model presented by Zhang et al [72].

We assume that TSVs are uniformly distributed in each device layer, the same as assumed in the two-layer 3-D chip's analytical model. Please note that in case of multi-layer 3-D ICs, the heights of TSVs will differ [72] depending upon how far apart certain devices/circuits that require vertical interconnections have been placed in the 3-D stack. For example, in the case of 4-layer 3D ICs, vias' height could be h, $2h$, and $3h$, where h is the vertical distance between two adjacent device layers. Let x_1 be the number of TSVs of height $1h$, x_2 be the number of TSVs of height $2h$ and x_3 is TSV count of height $3h$. Here, we decompose the vias of different heights into multiple vias of $1h$ heights. For example, if a via's height is $3h$, then we count it as three $1h$ vias. Thus the *ViaCount* is calculated as eqn. (8.10).

$$ViaCount = x_1 . 1 + x_2 . 2 + x_3 . 3 \tag{8.10}$$

After the conversion of via count using eqn. (8.10), we have found that analytical models for multi-layer 3-D ICs converge to our models that were derived for 2-layer 3-D ICs in this Chapter. The convergence happens due to the principle of superposition in which a multi-layer 3-D IC is decomposed into an set of identical two-layer 3-D ICs composed of consecutive device layers of the original chip as shown in Figure 8.3. The number of global wires can be obtained using eqn.(7.5) and eqn.(7.9) for an *m*-layer 3-D IC containing *N* logic gates as explained in sub-section 7.6.1 (in Chapter 7). Furthermore, the number of TSVs of different heights x_1, x_2, x_3,...,x_{m-1} (i.e. TSV height distribution) can be obtained from Zhang's 3-D wirelength distribution model [72] as given by eqn. (8.11).

$$x_z = (2m-2z)\frac{\alpha kN(1-N^{p-1}-m^{p-2}+m^{-1}N^{p-1})}{m(m-1)} \tag{8.11}$$

where z = 1, 2, 3, ...,*m*-1; *N* is the total number of gates in the 3-D chip, *k* and *p* are Rent's parameters, and α is the same as defined in eqn. (7.2) in chapter 7.

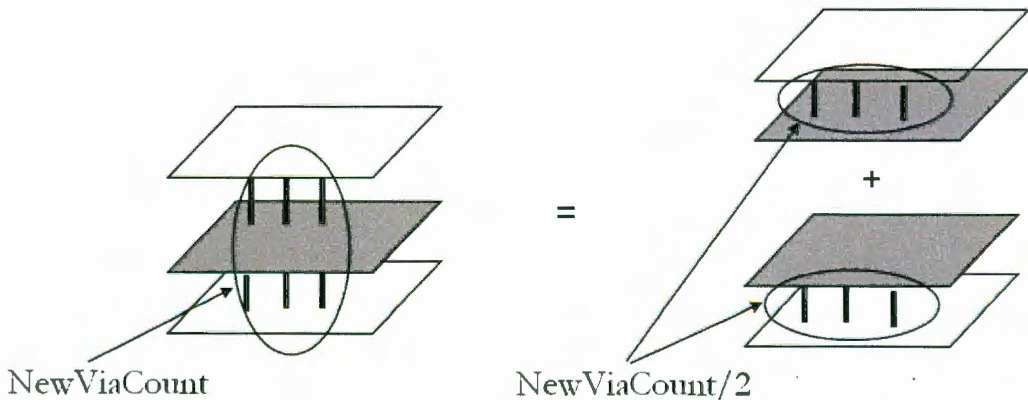


Figure 8.3: Decomposition of a 3-layer chip into a pair of identical two-layer 3-D ICs for yield calculation using superposition.

We assume that *critical path TSVs* also have a height distribution that is proportional to the TSV distribution given by eqn (8.11). This assumption is fairly accurate because these critical path TSVs are a small subset of all TSVs in a 3-D chip. Similar to eqn (8.10), we decompose the critical path TSVs of different heights into critical path TSVs of $1h$ height. Thus the original *CritTSV* is modified as the *new CritTSV* by using eqn (8.12).

$$NewCritTSV = CritTSV\left(\frac{1}{ViaCount} \times \sum_{z=1}^{m-1} z\, x_z\right) \tag{8.12}$$

where *ViaCount* is calculated by eqn (8.10), and x_z is computed using eqn (8.11). Please note that for a 2-layer 3D chip (i.e. $m = 2$), eqn (8.12) converges to *CritTSV*, i.e. the original number of critical path TSVs.

For a multi-layer 3-D IC, eqns. (8.1) to (8.9) are modified by computing *ViaCount* using eqn. (8.10), and replacing *CritTSV* by *NewCritTSV* that is calculated by eqn (8.12). Apart from these two changes, the rest of the expressions remain the same.

8.8 APPLICATION OF YIELD IMPROVEMENT STRATEGIES DURING FLOORPLANNING

As it was discussed in chapter 2, the floorplanning stage guides the placement stage during the physical design of VLSI chips. The proposed yield improvement techniques can be applied during floorplanning to estimate functional yield and chip revenue (Please see the flow chart in Figure 8.3). The analytical yield models presented in Sections 8.3 to 8.7 show that the functional yield depends on TSV defect rate and *ViaCount*. It is also important to note that *ViaCount* is a power term in the functional yield expression (given by eqn (8.1) to

eqn (8.3)) and therefore functional yield may change rapidly with change in *ViaCount*. Thus the floorplanning algorithm can optimize *ViaCount* such that an acceptable range of functional yield can be achieved for a given TSV defect rate by incorporating functional yield in the cost function of the floorplanner. Furthermore, the chip revenue model of eqn (8.9) can be incorporated in the cost function to maximize profitability. For example, let us say that a typical floorplanner optimizes area, inter-module wirelength, and *ViaCount* using the cost function given by eqn (8.13).

$$Fitness = \alpha\ DS + \beta\ WL + \gamma_1\ VC_{\text{inter}} \tag{8.13}$$

where ***DS*** is the dead space, ***WL*** is inter-module wirelength and $\boldsymbol{VC_{inter}}$ is inter-module via count. The constants $\alpha, \beta,$ and γ_1 are real valued tuning parameters. For a yield-aware 3-D floorplanning, the fitness function can be designed to incorporate functional yield and chip revenue as shown in eqn (8.14).

$$Fitness = \alpha\ DS + \beta\ WL + \gamma_1\ VC_{\text{inter}} + \gamma_2 Y_f^{RC} + \gamma_3\ \text{Revenue}(RC) \tag{8.14}$$

where Y_f^{RC} is the functional yield, and *Revenue(RC)* is the chip revenue for a redundancy configuration *RC* that was presented in Section 8.6. In addition, γ_1 and γ_2 are additional tuning parameters. The area penalty due to insertion of via redundancy can be added by increasing the sizes (width and height) of modules based on the area required by redundant vias and MUXes used for via rerouting. Since the floorplanning algorithms based on stochastic search methods such as simulated annealing (SA) and evolutionary algorithms (EA) are iterative procedures, the analytical yield models can be very useful in the fast

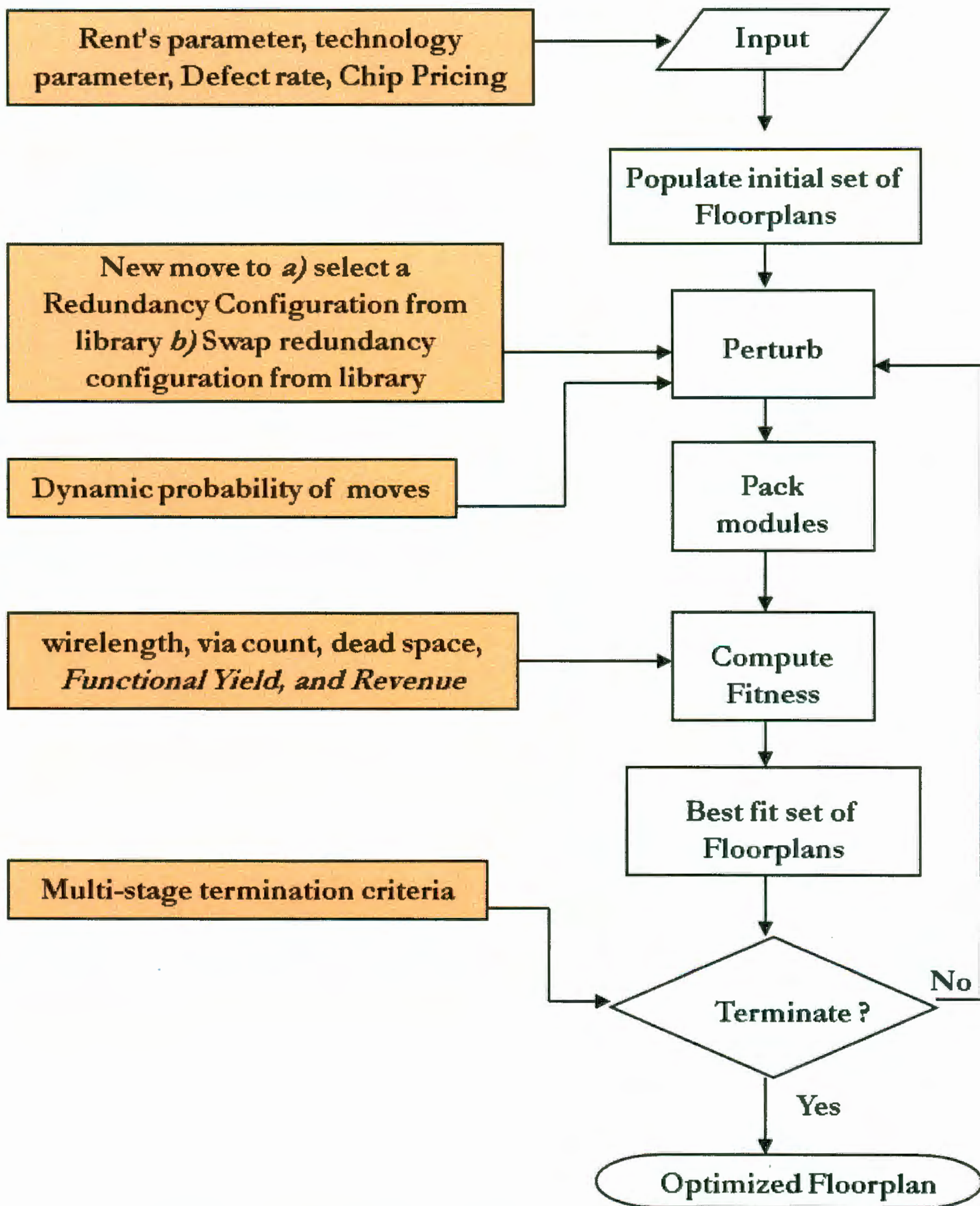


Figure 8.4: The flow chart of a Yield-Aware 3-D Floorplanning.

estimation of these yield metrics during runtime of the floorplanner. Furthermore, a library of these different redundancy configurations can be prepared, and the floorplanning algorithm can be randomly allowed to select a redundancy configuration from the library

using a set of new moves (see Figure 8.4) that can either randomly *select* a redundancy configuration for a floorplan solution or can *swap* the present redundancy configuration with another redundancy configuration from the library. The methodology to derive analytical models for other redundancy configurations is similar to the method presented in Section 8.1. However, the mathematical term introduced to minimize the error in eqn. (8.2) might be different, or may be required to introduce different values of r in the same equation. It might be possible to derive a unified analytical model for the three redundancy configurations (QWP, OWP, and OTC) if a generic method to handle the error for all of them can be established. Please note that there is an asymmetry in the analytical models because wireless vias are considered defect free whereas all TSVs (primary and redundant) have a non-zero probability of being defective.

CHAPTER 9: CONCLUSIONS AND FUTURE WORK

The following conclusions have been derived from the research work presented in this dissertation:

- Placement-aware 3-D floorplanning provides an opportunity for system level total wirelength reduction which can in turn reduce power consumed by interconnects.
- Placement-aware module splitting enables 3-D placement of logic gates which has the potential to reduce chip temperature and improve performance.
- The placement-aware 3-D floorplanning tool bridges the existing gap between 3-D floorplanning and 3-D placement.
- Feasibility conditions derived on a sequence pair representation help in eliminating the infeasible floorplan solutions that cannot satisfy vertical sub-module alignment.
- LCSLS provides a fast 3-D packing algorithm satisfying a set of vertical constraints. It eliminates the need for the creation of the time consuming 3-D constraint graphs used in the 3DCG algorithm.
- Statistical 2-D/3-D wirelength distribution models with analytical solutions help in the quick prediction of wirelength reduction due to placement-aware module splitting.

- The EA based stochastic method that uses a parallel search for an optimal solution has been shown to be faster and better than a SA based optimization for 3-D floorplanning problems.

- Vertical constraints on sequence pairs coupled with the LCSLS packing algorithm can be used for a fast 3-D floorplan with vertical module alignment for bus-driven 3-D design.

- Through Silicon Vias (TSVs) that connect circuits across different device layers suffer from thermo-mechanical stress and pose a serious yield loss problem. Wireless via redundancy along with TSVs introduced in the original design shows promising results for the yield enhancement of 3-D ICs.

- The proposed Quad Wireless Plus redundancy configuration provides a good balance between high yield and low redundancy overhead in terms of area, and delay.

- The Octal TSV Complete configuration provides high yield but it takes 100% redundant vias and uses 8:1 MUXes which has larger delay penalty than Quad Wireless Plus configuration (which uses 4:1 MUXes).

Research work is endless and there are always chances for improvement and the advancement of existing knowledge. Consequently, we propose the following future work:

- **Improvement in the cost function of 3-D Floorplanning:** The cost function is the single most important factor that significantly determines the quality and convergence of the search process in a combinatorial optimization. At present the 3-D FVC and 3-D FMA algorithms require different tuning parameters for different benchmarks. A range for each of the tuning parameters has been identified using the sensitivity analysis of the cost function. However, it would be worth investing in an improved cost function that does not require frequent changes in the tuning parameters. A meta-GA based approach may be useful in exploring improvement of the cost function.

- **Development of wirelength models with non-identical sub-modules**: At present, 3-D FVC splits modules into identical sub-modules only. However, depending on design requirements, it might be desirable to split modules into non-identical sub-modules. To estimate the wiring cost for such partitions, mathematical models that can handle non-identical rectangular sub-modules need to be developed. This type of design requirement can occur in a fixed outline 3-D floorplan design in which a module might be split into non-identical blocks if non-identical spaces are available within fixed dies of two or more device layers.

- **Investigation of novel redundancy configurations:** This dissertation presents a set of redundancy configurations for yield improvement. However, it would be worth

searching for any other efficient redundancy configurations that can improve yield while keeping the redundancy cost minimum (in terms of the number of redundant vias, delay, power, etc.).

- **Yield-aware 3-D floorplanning**: A tentative flow chart of a yield-aware 3-D floorplanning algorithm was presented in Chapter 8 which can be designed to quickly evaluate yield and revenue metrics during floorplan optimization. Since floorplanning is an iterative process, the number of TSVs may change during several iterations by an inter-layer perturbation that moves modules from one device layer to another. Thus the yield metrics will also change even for the same design at a fixed defect rate. Therefore analytical yield models will be very helpful in fast estimation of yield metrics during floorplanning.

- **Thermally Driven Placement-Aware 3-D Floorplanning**: Heat extraction and thermal management are among the major challenges faced in the design of 3-D ICs. Previous works on thermally driven 3-D floorplanning consider heat extraction using a heat sink only. However, recent advancement in heat extraction using the microchannel liquid cooling technique has been proposed in [15],[16],[17],[18],[19]. It would be worth designing a new thermally driven 3-D floorplanning that considers microchannel liquid cooling and optimizes the chip temperature along with area, wirelength, via count, etc., and comparing its effectiveness with the previous 3-D floorplanners.

- **CNT-based spiral inductor design for wireless vias:** The wireless vias in 3-D ICs use spiral inductors as transformers for interfacing between two device layers. As discussed in Chapter 7, the power consumption in wireless vias due to copper based inductors is higher compared to MUXes and TSVs used for redundancy. The inductor design should optimize the power dissipation and footprint area of the spiral inductor while considering the amount of inductance required and the tradeoffs in terms of parasitic resistance. Furthermore, the design should also consider imperfections imposed by CNT fabrication technology. For example, the CNTs needed for inductor design should be metallic. However, due to imperfections in the manufacturing process, semiconducting and metallic CNTs are usually mixed in a bundle. Therefore designers must also consider these fabrication imperfections during inductor design. This problem is a good MS thesis topic with strong chances of getting research publications.

- **Noise analysis for wireless vias within 3-D ICs:** The insertion of wireless vias will create additional electromagnetic field inside 3-D ICs [82]. The spiral inductors that are used as an interface between two device layers are generally created in the upper metal layer and they may interact with neighboring wires and TSVs causing cross-talk noise. Recent studies on TSV-to-TSV coupling have been presented in [110],[111],[112],[113]. Similarly, electromagnetic studies on wireless vias and misalignment tolerance between inductors have been presented in [95]. However, there are no studies on the interaction of wireless vias with TSVs or interconnects. Thus further study is required to determine how these inductors will interact with the

surrounding interconnects, TSVs and CMOS devices. One important thing to note is that even if there may be several redundant wireless vias inside 3-D chips, all of them need not be active at the same time. For example, let us assume that there are 1000 redundant wireless vias in a 3-D IC but there are only 100 failed TSVs. In this case, only 100 wireless vias will be activated using via re-routing. Furthermore, depending on the circuit functionality, all of them might not be switching simultaneously. Thus a careful noise analysis will be required.

CHAPTER 10: SUMMARY OF MAJOR CONTRIBUTIONS

The major contributions of this dissertation can be briefly summarized as follows:

- Extended a 3-D floorplanning tool inherited at the beginning of the research (which optimized area and inter-module wirelength, included module splitting to allow 3-D placement within split modules, and was satisfying vertical constraints). The new extended tool is called a placement-aware 3-D floorplan with vertical constraints algorithm (3-D FVC). The extension was accomplished by *a)* extending existing stochastic wirelength distribution models from square to rectangular 2-D modules and 2-D modules that are divided into two parts and placed in two consecutive device layers in 3-D ICs, *b)* designing a fast packing algorithm, *c)* deriving a set of vertical constraints on a sequence pair representation for vertical alignment of sub-modules, and *d)* making changes to many steps of the basic EA based floorplanning algorithm such as adding Rent's parameters and the technology node in the input, dynamic probabilities of moves, introducing additional cost components to the cost function, introducing two new moves (*submodule-merge and change feasibility configuration*) and multi-stage termination criteria. The extended algorithm (3-D FVC) bridges the pre-existing gap between 3-D floorplanning and 3-D placement tools. The vertical constraints derived on the sequence pair allow us to identify feasible solutions on the topological representation that can satisfy vertical constraints. 3-D FVC statistically captures the wirelength reduction due to 3-D placement inside modules and identifies certain sets of modules that can benefit from 3-D placement. As a result, it

reduces system level total wirelength by ~9.8% compared to existing state-of-the-art 3-D floorplanners that do not include placement-aware module splitting. This work appears in [33] and [34].

- Based on models available in the literature for square 2-D module wirelength distributions, a set of rectangular 2-D and 3-D wirelength distribution models (one rectangular 2-D model and three rectangular 3-D models) have been derived. These models allow us to estimate the wiring reduction within the modules due to 3-D placement of logic gates inside 3-D modules. This work has been published in [32].

- A fast module packing algorithm (LCSLS) has been designed which quickly translates the topological floorplan representation to geometrical floorplan while satisfying the vertical constraints imposed on modules/sub-modules. This work appears in [34].

- Modified our 3-D FVC algorithm to include vertical module alignment. The modified tool is called 3-D floorplan with vertical module alignment (3-D FMA). It satisfies designer-specified sets of constraints for bus-driven 3-D design and heterogeneous 3-D integration. It optimizes footprint area, inter-module wirelength and via count while satisfying the given set of constraints. An approximate runtime comparison with LTCG (another 3-D floorplanner that also performs vertical module alignment) shows that 3-D FMA is faster than LTCG.

- The 3-D IC yield problem due to failure of TSVs caused by thermo-mechanical stress has been formulated, and a set of redundant physical via/wireless via

redundancy configurations has been proposed. Based on the via redundancy technique, various redundancy configurations have been presented. Monte-Carlo simulation results show that the functional yield of 3-D ICs can be enhanced using our proposed via redundancy solutions. The initial version of this work has been published in [35] and [36]. A matured version of this work has been submitted for publication in a journal and is currently under review [37].

- The cost of redundancy overhead in terms of area, delay and power has been presented. Furthermore, a stochastic method for parametric yield estimation has been presented. This work appears in [36] and [37].

- A set of analytical models have been presented to quickly estimate functional yield, parametric yield, and chip revenue. The comparison of the analytical models' results with Monte-Carlo simulation results shows close agreement between them. This work appears in [36] and [37].

REFERENCES

[1] K. Banerjee, S. Souri, P. Kapur, and K.C. Saraswat, "3-D ICS: A Novel Chip Design for Improving Deep-Submicrometer Interconnect Performance and System-on-Chip Integration," *Proceedings of the IEEE*, Vol. 89, No. 5, pp. 602-633, May 2001.

[2] Y. Deng, and W. P. Maly, "Interconnect Characteristics of 2.5-D System Integration Scheme", *Proc. Int'l Symp. Physical Design*, pp. 171 – 175, Apr 2001.

[3] S. Das, A. Chandrakasan, and R. Reif, "Design Tools for 3-D Integrated Circuits," *Proc. ASP-DAC*, pp.53-56, Jan. 2003.

[4] S. Das, A. Chandrakasan, and R. Reif. "Three-Dimensional Integration: Performance, Design Methodology, and CAD Tools," *Proc. ISVLSI*, pp. 13-18, Feb. 2003.

[5] A. Fan, A. Rahman, and R. Reif, "Copper wafer bonding," *Electrochem. Solid State Lett.*, vol. 2, pp. 534-536, 1999.

[6] A. Fan, R. Reif, K. Chen, and S. Das, "Fabrication Technologies for Three-Dimensional Integrated Circuits," *Proc. Int'l Symp. Quality Electronic Design*, pp. 33-37, 2002.

[7] S. Reda, G. Smith, and L. Smith, "Maximizing the Functional Yield of Wafer-to-Wafer 3-D Integration," *IEEE Trans. VLSI*, vol. 17(9), pp. 1357 - 1362, 2009.

[8] L. Smith, G. Smith, S. Hosail, and S. Arkalgud, "3-D: It All Comes Down to Cost," *3-D Architectures for Semiconductor Integration and Packaging*, 2007.

[9] S. Lim, "TSV-Aware 3-D Physical Design Tool Needs for Faster Mainstream Acceptance of 3-D ICs," *DAC 2010 knowledge center article*, pp. 1-11, 2010.

[10] K. Bernstein, P. Andry, J. Cann, P. Emma, D. Greenberg, W. Haensch, M. Ignatowski, S. Koester, J. Magerlein, R. Puri, and Albert Young, "Interconnects in the Third Dimension: Design Challenges for 3-D ICs," *Proc. Design Automation Conference*, pp. 562-567, 2007.

[11] International Technology Roadmap for Semiconductors (ITRS), www.itrs.net

[12] B. Goplen, and S. Sapatnekar, "Thermal Via Placement in 3-D ICs," *Proc. Intl. Symp. on Physical Design*, pp. 167-174, 2005.

[13] J. Cong, J. wei, Y. Zhang, "A Thermal-Driven Floorplanning Algorithm for 3-D ICs," *Proc. Int. Conf. on Computer Aided Design,* pp. 306-313, *Nov 2004.*

[14] B. Goplen, and S. Sapatnekar, "Efficient Thermal Placement of Standard Cells in 3-D ICs using a Force Directed Approach," *Proc. Int. Conf. on Computer Aided Design,* pp. 86-89, 2002.

[15] D. Tuckerman and R. W. Pease, "High-Performance Heat Sinking for VLSI," *IEEE Electron Device Letters*, vol. 2(5), pp. 126–129, 1981

[16] H. Mizunuma, M. Behnia, and W. Nakayama, "Forced Convective Boiling of a Fluorocarbon Liquid in Reduced Size Channels - An Experimental Study," *Journal of Enhanced Heat Transfer,* vol. 9(2), pp. 69–76, 2003.

[17] W. Qu and I. Mudawar, "Thermal Design Methodology for High-Heat-Flux Single-Phase and Two-Phase Micro-Channel Heat Sinks," *IEEE Transactions on Components and Packaging Technologies,* vol. 26(3), pp.598–609, 2003.

[18] X. Wei and Y. Joshi, "Optimization Study of Stacked Micro-Channel Heat Sinks for Micro-Electronics Cooling," *IEEE Transactions on Components and Packaging Technologies,* vol.26(1), pp. 55–61, 2003.

[19] N. Lei, A. Ortega, and R. Vaidyanathan, "Modeling and Optimization of Multilayer Minichannel Heat Sinks in Single-Phase Flow," *In IEEE InterPACK Conference, InterPACK 2007-33329,* pp. 29-43, 2007.

[20] T. Brunschwiler, B. Michel, H. Rothuizen, U. Kloter, B. Wunderle, H.Oppermann, and H. Reichl, "Forced Convective Interlayer Cooling In Vertically Integrated Packages," *Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems,* pp. 1114-1125, 2008.

[21] D. Selar, C. King, B. Dang, T. Spencer, H. Hacker, P. Joseph, M.Bakir and J. Meindl, "A 3-D-IC Technology with Integrated Microchannel Cooling," *International Interconnect Technology Conference,* vol.1(4), pp.13–15, 2008.

[22] H. Mizunuma, C Yang, and Y. Lu, "Thermal Modeling for 3-D-ICs with Integrated Microchannel Cooling," *Proc. . Int. Conf. on Computer Aided Design,* pp. 256-263, 2009.

[23] R. Filippi, J. McGrath, T. Shaw, C. Murray, H. Rathore, P. McLaughlin, V. McGahay, L. Nicholson, P. Wang, J. Lloyd, M. Lane, R. Rosenberg, X. Liu, Y. Wang, W. Landers, T. Spooner, J. Demarest, B. Engel, J. Gill, G. Goth, E. Barth, G. Biery, C. Davis, R. Wachnik, R. Goldblan, T. Ivers, A. Swinton, C. Barile, and J. Aitken, "Thermal Cycle Reliability of Stacked Via Structures with Copper Metallization and an Organic Low-k Dielectric," *Int'l Reliability Physics Symposium,* pp. 61-67, 2004.

[24] R. S. Patti, "Three-Dimensional Integrated Circuits and the Future of System-on-Chip Designs," *Proceedings of the IEEE,* vol. 94(6), pp. 1214-1224, 2006.

[25] C. Ferri, S. Reda, and I. Bahar, "Strategies for Improving the Parametric Yield and Profits of 3-D ICs," *Proc. Intl. Conference on Computer-Aided Design,* pp. 220-226, 2007.

[26] G. Smith, L. Smith, S. Hosali, and S. Arkalgud, "Yield Considerations in the Choice of 3-D Technology," *Intl. Symp. on Semiconductor Manufacturing,* pp. 535-537, 2007.

[27] D. Bentz, J. Zhang, M. Bloomfield, Jian-Qiang Lu, R J. Gutmann, and T. S. Cale, "Modeling Thermal Stresses of Copper interconnects in 3-D IC Structures ," *Proc. COMSOL Multiphysics User's conference*, pp. 321-326, 2005.

[28] J. Zhang, M. Bloomfield, J-Q Lu, R. J. Gutmann, and T. S. Cale, "Modeling Thermal Stresses in 3-D IC Inter-wafer Interconnects," *IEEE Transactions on Semiconductor Manufacturing*, vol. 19(4), pp. 437- 447, 2006.

[29] B. Swinen, W. Ruythooren, P. Moor, L. Bogaerts, L. Carbonell, K. Munck, B. Eyckens, S. Stoukatch, D. Tezcan, Z. Tokei, J. Aelst, and E. Beyne, "3-D Integration by cu-cu Thermo-compression Bonding of Extremely Thinned Bulk-Si Die Containing 10 μm pitch Through-Si vias," *International Electron Devices Meeting*, pp. 1 - 4, 2006.

[30] C. Bower, D. Malta, D. Temple, J. Robinson, P. Coffman, M. Skokan, and T. Welch, "High Density Vertical Interconnects for 3-D Integration of Silicon Integrated Circuits," *IEEE Electronic Components and Technology Conference*, pp. 399 – 403, 2006.

[31] A. Kahng, "Classical Floorplanning Harmful," Proc. *ISPD*, pp. 207–213, 2006.

[32] Rajeev Nain, Rajarshi Ray, and Malgorzata Chrzanowska-Jeske, "Rectangular 3-D Wirelength Distribution Models," *Proc. IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, pp. 109 – 112, 2008.

[33] Rajeev Nain, and Malgorzata Chrzanowska-Jeske, "Placement-aware 3-D Floorplanning," *Proc. IEEE International Symposium on Circuits and Systems (ISCAS)*, pp.1727-1730, 2009.

[34] Rajeev Nain, and Malgorzata Chrzanowska-Jeske, "Fast Placement-Aware 3-D Floorplanning using Vertical Constraints on Sequence Pairs," In press, *IEEE Transactions on Very Large Scale Integration (TVLSI) Systems*, vol. pp. no. 99, 1-14, doi: 10.1109/TVLSI.2010.2055247,URL:

http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5545492&isnumber=4359553

[35] Rajeev Nain, and Malgorzata Chrzanowska-Jeske, "3-D Yield in the Presence of Defects in Through Signal Vias," *IEEE International workshop on Design for Manufacturability & Yield (DFM&Y)*, pp. 84-87, 2009.

[36] Rajeev Nain, Shantesh Pinge, and Malgorzata Chrzanowska-Jeske, "Yield Improvement of 3-D ICs in the presence of Defects in Through Signal Vias," *IEEE Int'l Symposium on Quality Electronic Design (ISQED)*, pp. 598-605, 2010.

[37] Rajeev Nain, Rehman Ashraf, and Malgorzata Chrzanowska-Jeske, "3-D IC Yield Enhancement in the Presence of Through-Silicon Via Failure," Under Review, *IEEE Transactions on Very Large Scale Integration (TVLSI) Systems.*

[38] P. Miranda, A. Moll, "Thermo-Mechanical Characterization of Copper Through-Wafer Interconnects", *IEEE Electronic Components and Technology Conference,* pp. 844-848, 2006.

[39] G. Blakiewicz, Malgorzata Chrzanowska-Jeske, M. C. Jeske and J. Zhang, "Substrate Noise Modeling in Early Floorplanning of MS-SOCs", ASPDAC, pp. 819 – 823, 2005.

[40] M. Healy, M. Vittes, M. Ekpanyapong, C. Ballapuram, S. Lim, H. Lee, G. Loh, "Micro architectural floorplanning under performance and thermal tradeoff", *Design & Test in Europe (DATE),* pp. 1288-1293, 2006.

[41] H. Xiang, X. Tang and Martin D. F. Wong, "Bus-Driven Floorplanning", *Int. Conference on Computer Aided Design,* pp. 66, 2003.

[42] F. Balasa, "Modeling Non-Slicing Floorplans with Binary Trees", *Int. Conference on Computer Aided Design,* pp.13-16, 2000.

[43] X. Tang and D. F. Wong, "Floorplanning with Alignment and Performance Constraints", *Proc. Design Automation Conference,* pp. 848-853, 2002.

[44] G.Wu, S.Wu, Y. W. Chang, and Y. C. Chang, "B*-trees: A New Representation for Non-slicing Floorplans," *Proc. Design Automation Conference,* pp. 458–463, Jun. 2000.

[45] Z. Li, X. Hong, Q. Zhou, Y. Cai, J. Bian, H. Yang, V. Pitchumani, and C. K. Cheng, "Hierarchical 3-D Floorplanning Algorithm for Wirelength Optimization," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 12, pp. 2637–2646, Dec. 2006.

[46] E. Young, C. Chu and M. Ho "Placement Constraints in Floorplan Design", *IEEE Transactions on VLSI systems,* vol. 12, issue 7, July 2004, pp. 735-745, 2004.

[47] H. Murata, E. Kuh, "Sequence Pair Based Module Placement for hard/soft/pre-placed Modules", *ISPD 1998,* pp. 167-172, 1998.

[48] Adya, S. N. and Markov, I. L., "Fixed-Outline Floorplanning: Enabling Hierarchical design", *Proceedings of the International Conference on Computer Design*, pp.1120 – 1135, 2003.

[49] Y. Tam, E. F. Y. Young and C. Chu, "Analog Placement with Symmetry and other Placement Constraints", *Intl. Conference on Computer Aided Design,* pp. 349-354, 2006.

[50] L. Xiao and E. F. Y. Young, "Analog Placement with Common Centroid and 1-D Symmetry Constraints", *Proceedings of ASPDAC,* pp. 353-360, 2009.

[51] B. Wang , Malgorzata Chrzanowska-Jeske, and G. Greenwood, "ELF-SP – Evolutionary Algorithm for Non-slicing Floorplans with Soft Modules", *Proc. Int. Symposium on Circuits and Systems,* vol.2, pp.681-684, 2002.

[52] Malgorzata Chrzanowska-Jeske, B. Wang and G. Greenwood, "Floorplanning with Performance-based Clustering", *Proc. Int. Symposium on Circuits and Systems,* vol. 4, pp. 724-727, 2003.

[53] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI Module Placement Based on Rectangle Packing by the Sequence Pair," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.,* vol. 15, no. 12, pp. 1518–1524, Dec. 1996.

[54] S. Nakatake, K. Fujiyoshi, H. Murata, and Y. Kajitanid, "Module Packing Based on the BSG-structure and IC Layout Applications," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.,* vol. 17, no. 6, pp.519–530, June 1998.

[55] P. N. Guo, C. Cheng, and T. Yoshimura, "An O-tree Representation of Non-Slicing Floorplan and its Applications," *in Proc. DAC,* 1999, pp.268–273, 1999.

[56] X. Hong, G. Huang, Y. Cai, J. Gu, S. Dong, C. Cheng, and J. Gu,"Corner Block List: An Effective and Efficient Topological Representation of Non-slicing Floorplan," *Proc. Int. Conf. Comput.-Aided Des.,* Nov. 2000, pp. 8–12.

[57] J. M. Lin and Y. W. Chang, "TCG: A Transitive Closure Graph Based Representation for Non-slicing Floorplans," *Proc. Design Automation Conference,* pp. 764–769, June 2001.

[58] X. Tang and D.Wong, "FAST-SP:A Fast Algorithm for Block Placement Based on Sequence Pair," *Proc. ASP-DAC,* pp. 521–526, 2001.

[59] K. Bazargan, R. Kastner, and M. Sarrafzadeh, "3-D Floorplanning: Simulated Annealing and Greedy Placement Methods for Reconfigurable Computing Systems," *Proc. Int. Workshop Rapid Syst. Prototyping,* pp. 38–43, June 2000.

[60] L. Cheng, L. Deng, and D. F. Wong, "Floorplanning for 3-D VLSI design," *Proc. ASP-DAC,* pp. 405–411, 2005.

[61] H. Yamazaki, K. Sakanushi, S. Nakatake, and Y. Kajitani, "The 3-D-packing by Meta Data Structure and Packing Heuristics," *IEICE Trans. Fundamentals,* vol. E38-A, pp. 639–645, 2000.

[62] P. Yuh, C. Yang, and Y. Chang, "Temporal Floorplanning using the T-tree Formulation," *Proc. Int. Conference on Computer Aided Design,* pp. 300–305, 2004.

[63] P. Shiu, R. Ravichandran, S. Easwar, and S. Lim, "Multi-layer Floorplanning for Reliable System-on-Package," *Proc. Int. Symp. Circuits Syst.,* vol. 5, pp. V-69–V-72, May 2004.

[64] W. L. Hung, G. M. Link, Y. Xie, N. Vijaykrishnan, and M. J. Irwin, "Interconnect and thermal-aware floorplanning for 3-D microprocessors," *Proc. Int. Symposium on Quality Electronic Design,* pp. 98–104, Mar. 2006.

[65] P. Zhou, Y. Ma, Z. Li, R. Dick, L. Shang, H. Zhou, X. Hong, and Q. Zhou, "3-D STAF: Scalable temperature and leakage aware floorplanning for three-dimensional integrates circuits," *Proc. Int. Conference on Computer Aided Design,* pp. 590–597, 2007.

[66] T. Ma, and E. Y. Young, "TCG-based Multi-Bend Bus Driven Floorplanning," *Proc. ASP-DAC,* pp. 192 – 197, 2008.

[67] J. Law, E. Young, and R. Ching, "Block Alignment in 3-D Floorplan Using Layered TCG," Proc. GLSVLSI, pp. 376-380, 2006.

[68] K. Fujiyoshi, and H. Murata "Arbitrary Convex and Concave Rectilinear Block Packing Using Sequence-Pair," *IEEE Trans. CAD,* vol.19, no.2, pp.224-233, 2000.

[69] X Dong, and Y Xie, "System-Level Cost Analysis and Design Exploration for Three-Dimensional Integrated Circuits (3-D ICs)," *Proc. ASP-DAC,* pp. 234-241, 2009.

[70] J. Cong, G Luo, J. Wei and Y. Zhang, "Thermal-Aware 3-D IC Placement Via Transformation," *Proc. ASP-DAC,* pp. 780-785, 2007.

[71] J. Davis, V. De, and J. Meindl, "A stochastic Wirelength Distribution for Gigascale Integration (GSI)—Part I: Derivation and validation," *IEEE Trans. Electron Devices,* vol. 45, no. 3, pp. 580–589, Mar. 1998.

[72] R. Zhang, K. Roy, C. Koh, and D. Janes, "Stochastic Wire-length and Delay distributions of 3-dimensional circuits," *Proc. Int. Conference on Computer Aided Design*, pp. 208–214, 2000.

[73] A. Rahman, A. Fan, and R. Reif, "Wire-length Distribution of Three-Dimensional Integrated Circuits," *Proc. IEEE Int. Conference on Interconnect Technology*, pp. 671–678, 1999.

[74] B. S. Landman, and R. L. Russo, "On a pin versus block relationship for partitions of logic blocks," *IEEE Trans. Comput.*, vol. 20, no. 12,pp. 1469–1479, Dec. 1971.

[75] S. Das, A. Chandrakasan, and R. Rief, "Calibration of Rent's rule models for three-dimensional integrated circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 12, no. 4, pp. 359–366, Apr.2004.

[76] M. Lanzerotti, G. Fiorenza, and A. Rand. "Predicting Interconnect Requirements in Ultra-large-scale Integrated Control Logic Circuitry." *Proc. SLIP*, pp. 43-50, 2005.

[77] M. Lanzerotti, G. Fiorenza, and A. Rand, "Microminiature packaging and integrated circuitry: The work of E. F. Rent, with an application to on-chip interconnection requirements," IBM J. Res. Dev., vol. 49, no.4/5, pp. 777–803, Jul./Sep. 2005.

[78] Pingqiang Zhou (The author of 3-D STAF [65]), *Personal communication,* May 6, 2008.

[79] J. W. Joyner, "Opportunities and Limitations of Three-dimensional Integration for Interconnect Design," *Ph.D. Thesis*, Georgia Institute of Technology, July 2003.

[80] A. Rahman, "System Level Performance Evaluation of Three-Dimensional Integrated Circuits," *Ph.D. Thesis*, MIT, Feb 2001.

[81] W. E. Donath, "Placement and Average Interconnections Lengths of Computer Logic," *IEEE Trans. of circuits syst.* Vol. CAS-26, pp. 272-277, Apr. 1979.

[82] J. Xu, L. Luo, S. Mick, J. Wilson, and P. Franzon, "AC Coupled Interconnect for Dense 3-D ICs," *IEEE Trans. on Nuclear Science*, vol. 51, no. 5, pp. 2156-2160, Oct. 2004.

[83] S. Mick, J. Wilson, and P. Franzon, "4 Gbps High-Density AC Coupled Interconnection," Proc. Custom Integrated Circuits Conference, pp. 133-140, 2002.

[84] S. Kuhn, M. Kleiner, R. Thewes, and W. Weber, "Vertical Signal Transmission in Three-dimensional Integrated Circuits by Capacitive Coupling," *1995. IEEE International Symposium on Circuits and Systems*, pp.37-40, 1995.

[85] W. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A. Sule, M. Streer, and P. Franzen, "Demystifying 3-D ICs: The Pros and Cons of Going Vertical," *IEEE Design & Test of Computers*, vol. 22, no. 6, pp. 498-510, Nov.-Dec.2005.

[86] K. Tsubaki, H. Shioya, J. Ono, Y. Nakajima, T. Hanajiri, and H. Yamaguchi, "Large magnetic field induced by carbon nanotube current -proposal of carbon nanotube inductors," *Device Research Conference Digest*, vol. 1, pp. 119-120, June 2005.

[87] K. Tsubaki, Y. Nakajima, T. Hanajiri, and H. Yamaguchi, "Proposal of Carbon Nanotube Inductors," *Institute of Physics Publishing Journal of Physics: Conference Series* 38, pp. 49-52, 2006.

[88] A. Nieuwoudt, and Y. Massoud, "Carbon Nanotube Bundle-Based Low Loss Integrated Inductors," *IEEE Int'l Conf. on Nanotechnology*, pp. 714-718, 2007.

[89] M. Budnik, A. Raychowdhury, A. Bansal, and K. Roy, "A High Density, Carbon Nanotube Capacitor for Decoupling Applications," *Proc. Design automation Conf.* 2006, pp. 935-938.

[90] H. B. Bakoglu, "Circuits, Interconnections, and Packaging for VLSI," *Addison-Wesley*, 1990.

[91] www.ece.virginia.edu/~mrs8n/cadence/SynthesisTutorial/tsmc18.pdf

[92] N. Weste and D. Harris, "CMOS VLSI Design – A Circuits and Systems Perspective," *Addison-Wesley*, 2005.

[93] J. A. Davis, V. K. De, J. A. Meindl, "A Stochastic Wire-Length Distribution for Gigascale Integration (GSI)—Part II: Applications to Clock Frequency, Power Dissipation, and Chip Size Estimation," *IEEE Trans. on Electron Devices*, Vol. 45, No. 3, pp. 500-597, 1998.

[94] Y. Ma, Y. Liu, E. Kursun, G. Reinman, and J. Cong, "Investigating the Effects of Fine-grain Three-Dimensional Integration on Microarchitecture Design," *ACM J. Emerg. Technol. Comput.* Syst., vol. 4, no. 4, 2008, Article 17.

[95] T. Kuroda, "Wireless Proximity Communications for 3D System Integration, *IEEE Int'l Workshop on Radio Frequency Integration Technology*, pp. 21-25, Dec. 2007.

[96] P. Jacob, O. Erdogan, A. Zia, P. Belemjian, R. Kraft, and J. McDonald, "Predicting the performance of a 3D Processor-Memory Chip Stack," *IEEE Design & Test of Computers*, vol. 22, Issue 6, pp. 540-547, Nov. - Dec. 2005.

[97] C. Liu, I. Ganusov, M. Burtscher, and S. Tiwari, "Bridging the Processor-Memory Performance Gap with 3D IC Technology," *IEEE Design & Test of Computers*, vol. 22, Issue 6, pp. 556 – 564, Nov. – Dec. 2005.

[98] G. Neudeck, S. Pae, J. Denton, and. T. Su, "Multiple Layers of Silicon-on-Insulator for Nanostructure Devices," *Journal of Vacuum Science and Technology B: Microelectronics and Nanometer Structures*, vol. 17, no. 3, pp. 994-998, 1999.

[99] K. Saraswat, K. Banerjee, A. Joshi, P. Kalavade, P. Kapur, and S. Souri, "3-D ICs: Motivation, performance analysis, and technology," in Proc. 26th European Solid-State Circuits Conference. (ESSCIRC), pp. 406-414, 2000.

[100] X. Tang, and M. Wong, "On Handling Arbitrary Rectilinear Shape Constraint," *Proc. ASP-DAC*, pp. 38 – 41, 2004.

[101] Benyi Wang, and Malgorzata Chrzanowska-Jeske, "A Basic 3-D Floorplan tool," *C++ Software*, Unpublished.

[102] Y. Xia, and Malgorzata Chrzanowska-Jeske, "Considering Layout For Test Scheduling of Core-Based SoCs," in *Proc. ICECS*, pp. 1 – 4, 2005.

[103] Kenneth De Jong, "Evolutionary Computation," *MIT Press*, 2002.

[104] Garrison Greenwood, "ECE559 : Genetic Algorithms," Graduate Coursework at Portland State University, Spring 2010.

[105] M. Lanzerotti, G. Fiorenza, and R. Rand, "Assessment of On-Chip Wire-Length Distribution Models," IEEE Transactions on VLSI Systems, vol. 12, no. 10, pp. 1108-1112, Oct. 2004.

[106] G. Karypis, and V. Kumar, "hMetis: A Hypergraph Partitioning Package. [Online] Available: http://www.users.cs.umn.edu/~karypis/metis/hmetis/index.html

[107] J. Lu, "3-D Hyperintegration and Packaging Technologies for Micro-Nano Systems," Proceedings of the IEEE, vol. 97, no. 1, pp. 18 – 30, January 2009.

[108] [Online] http://en.wikipedia.org/wiki/Sensitivity_analysis

[109] J. Helton, J. Johnson, C. Sallaberry, and C. Storlie, "Survey of Sampling-Based Methods for Uncertainty and Sensitivity Analysis," Sandia National Laboratories Report, June 2006.

[110] T. Song, C. Liu, D. H. Kim, J. Cho, J. Kim, J. S. Park, S. Ahn, J. Kim, and S. K. Lim, "Analysis of TSV-to-TSV Coupling with High-Impedance Termination in 3D ICs", *to appear in 12th IEEE International Symposium on Quality Electronic Design (ISQED)*, March 2011.

[111] J. Cho, J. Shim, E. Song, J. Pak, J. Lee, H. Lee, K. Park, and J. Kim, "Active circuit to through silicon via (TSV) noise coupling," *In IEEE Electrical Performance of Electronic Packaging and Systems*, pp. 97 – 100, 2009.

[112] K. Yoon, G. Kim, W. Lee, T. Song, J. Lee, H. Lee, K. Park, and J. Kim, " Modeling and analysis of coupling between TSVs, metal, and RDL interconnects in TSV-based 3D IC with silicon interposer," *In Proc. IEEE Electronics Packaging Technology Conf.*, pp. 702 – 706, 2009.

[113] B. Curran, I. N. dip, S. Guttovski, and H. Reichl, "The impacts of dimensions and return current path geometry on coupling in single-ended Through Silicon Vias," *IEEE Electronic Components and Technology Conference*, pp. 1092 – 1097, 2009.

[114] M. Kang and W. Dai, "General Floorplanning with L-shaped, T-shaped and Soft Blocks based on Bounded Slicing Grid Structure," *in Proc. ASP-DAC*, pp. 265–270, 1997.

[115] S. Nakatake, M. Furuya, and Y. Kajitani, "Module Placement on BSG structure with Pre-placed Modules and Rectilinear Modules," *Proc. ASP-DAC*, pp. 571–576, 1998.

[116] J. Xu, P. N. Guo, and C. K. Cheng, "Rectilinear Block Placement using Sequence-Pair," *Proceedings of International Symposium on Physical Design*, pp. 173–178, 1998.

[117] Y. Pang, C.-K. Cheng, K. Lampaert, and W. Xie, "Rectilinear Block Placement using O-tree representation," *Proceedings of International Symposium on Physical Design*, pp. 156–161, 2001.

[118] Y. Ma, X. Hong, S. Dong, Y. Cai, C.-K. Cheng, and J. Gu, "Floorplanning with abutment constraints and L-shaped/T-shaped blocks based on corner block list," in Proc. Design Automation Conference, pp. 770–775, 2001.

[119] G. M. Wu, Y.C. Chang, and Y.W. Chang, "Rectilinear Block Placement using b*-trees," *ACM Transactions on Design Automation of Electronic Systems*, vol. 8(2), pp.188–202, 2003.

[120] J.-M. Lin, H.-L. Lin, and Y.-W. Chang, "Arbitrarily Shaped Rectilinear Module Placement using the Transitive Closure Graph Representation," *IEEE Transactions on VLSI Systems*, vol. 10(6), pp. 886–901, 2002.

[121] W. Davis, E. Oh, A. Sule, and P. Franzon, "Application Exploration for 3-D Integrated Circuits: TCAM, FIFO, and FFT Case Studies," IEEE Transactions on Very Large Scale Integration Systems, vol. 17, no. 4, pp. 496-506, Apr. 2009.

[122] 3-D IC & TSV Interconnects, *Yole Development*, 2010 reports. [online]: http://www.i-micronews.com/upload/Rapports/3D%20flyer.pdf

[123] A. Joseph, J. Gillis, M. Doherty, P. Lindgren, R. Kelly, R. Malladi, P. Wang, M. Erturk, H. Ding, E. Gebreselasie, M. McPartlin, and J. Dunn, "Through-silicon vias enable next-generation SiGe power amplifiers for wireless communications," IBM journal of Research and Development, vol. 52. No. 6, pp. 635-648, Nov. 2008.

[124] [online]: cseweb.ucsd.edu/classes/wi10/cse241a/slides/t_line.ppt

[125] S. Salewski, E. Barke, "An upper bound for 3D slicing floorplans," Proc. ASP-DAC, pp. 567-572, 2002.

APPENDIX A: MATHEMATICAL PROOFS OF THE FEASIBILITY CONDITION THEOREMS

We present formal proofs of the feasibility condition theorems stated in section 3.4.

Theorem1:

Feasibility Condition for two pairs of modules: Given a two-layer feasibility condition graph **G**(V, E), there exists a feasible solution to the vertical constraint problem for two module pairs represented by the graph if:

(a) ***G*** *contains a clique of size K (K* $\in$ *{3,4}), or*

(b) ***G*** *contains two cliques of size 2, such that each clique contains only nodes of the same color.*

Proof: Let us consider the case of a clique with size 4 as shown in Figure A.1 in which module A_1 has to be aligned with A_2, and B_1 has to be aligned with B_2.

Let X_{mi} denotes the x coordinate of the lower left corner of (sub) module **m** in the ith device layer.

Sequence Pair of Layer 1 : $<A_1, B_1 ; A_1, B_1>$

$$\Rightarrow A_1 \text{ is to the left of } B_1 \Rightarrow X_{A1} < X_{B1} \tag{A.1}$$

Sequence Pair of Layer 2 : $<A_2, B_2 ; A_2, B_2>$

$$\Rightarrow A_2 \text{ is to the left of } B_2 \Rightarrow X_{A2} < X_{B2} \tag{A.2}$$

Let us assume, $X_{A1} - X_{A2} = \delta$, where δ is an arbitrary constant such that $\delta \geq 0$

$$\Rightarrow X_{A1} = X_{A2} + \delta \tag{A.3}$$

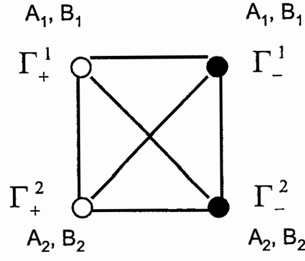


Figure A.1: Two module pairs $\{A_1, A_2\}$ and $\{B_1, B_2\}$ with vertical constraints in 2-device layers.

From Eqn (A.1) and (A.3):

$$X_{A2} + \delta < X_{B1} \tag{A.4}$$

From Eqn (A.2):

$$X_{A2} < X_{B2} \quad => \quad X_{A2} + \delta < X_{B2} + \delta \tag{A.5}$$

Combining Eqn (A.4) and (A.5):

$$X_{A2} + \delta < X_{B1}$$

$$X_{A2} + \delta < X_{B2} + \delta$$

=> A_2 can be shifted rightward by δ to align it with A_1 first. After that $\{B_1, B_2\}$ can be aligned together without disturbing the alignment of $\{A_1, A_2\}$ because B_1 and B_2's x-coordinates are to the right of both A_1 and A_2.

Therefore a clique of size 4 forms a feasible solution for vertical alignment. Similarly we can show that a clique of size 3 forms a feasible solution.

Condition (b) of theorem 1 is the same as a clique of size 4 except that lateral shifting will take place along the y-axis. Hence theorem1 is proved.

Theorem 2:

Two Layer Feasibility Condition: Let $L_1 = \{A_1, B_1, C_1, D_1, \ldots\ldots\}$ and $L_2 = \{A_2, B_2, C_2, D_2, \ldots\ldots\}$ are two sets of modules located in two different device layers L_1 and L_2 respectively. The packing on L_1 and L_2 are represented by Constrained Sequence Pairs SP_1 and SP_2 respectively. SP_1 and SP_2 are feasible if module pairs $\{(A_1, A_2), (B_1, B_2), (C_1, C_2), (D_1, D_2), \ldots\ldots\}$ can be vertically aligned simultaneously. The vertical alignment of all these module pairs is feasible if:

Every combination of two module pairs decomposed from L_1 and L_2 (without changing their relative orders) construct a feasible configuration by satisfying theorem 1 i.e.

$\{u_1, u_2\}$ and $\{v_1, v_2\}$ form **a feasible configuration** $\forall \{u_1, v_1\} \in SP_1; \forall \{u_2, v_2\} \in SP_2$.

Proof (by contradiction): Let us assume that SP_1 and SP_2 are feasible constrained sequence pairs, **but** every combination of two module pairs **does not** make a feasible configuration.

If every combination of two module pairs **does not** make feasible configuration,

=> There exists ***at least*** one combination of two module pairs which makes an infeasible configuration i.e.

(r_1, r_2) and (s_1, s_2) do not form a feasible configuration where $\{r_1, s_1\} \in SP_1;\ r_1 \neq s_1$ *and* $\{r_2, s_2\} \in SP2;\ r_2 \neq s_2$

=> SP_1 and SP_2 become infeasible because (r_1, r_2) and (s_1, s_2) cannot be vertically aligned simultaneously. *(Contradiction to our initial assumption).*

Hence it is proved *by contradiction* that for SP_1 and SP_2 to be feasible, every combination of **two module pairs** should construct a feasible configuration, thereby satisfying theorem 1.

Theorem3:

Multi Layer Feasibility Condition: Given a set of multi-layer constrained sequence pairs, there exists a feasible solution to the vertical constraint problem if:

Each combination of 2-device layers satisfies the 2-layer feasibility condition theorem (i.e. theorem 2).

Proof (using principle of superposition): Let us assume that there are L device layers ($L > 2$) in which modules are vertically constrained. The L layer feasibility condition can be decomposed in ${}^{L}c_{2}$ combinations of two-layer feasibility condition (theorem 2). If any one of the ${}^{L}c_{2}$ combinations violates theorem 2, then sub-modules constrained between those two layers can never be aligned. Thus the multi-layer feasibility condition will fail.

=> For the multi-layer feasibility condition, all ${}^{L}c_{2}$ combinations must satisfy the two layer feasibility condition (theorem 2).

APPENDIX B: STOCHASTIC RECTANGULAR 3-D WIRELENGTH DISTRIBUTION MODELS

B.1 INTRODUCTION

The wirelength distribution model provides a stochastic relation between the *total number* of interconnects and the *length* of those interconnects. The length is measured in "gate pitch" unit, defined by the average separation between adjacent gates in a chip. A wirelength distribution model is used for interconnect prediction, mostly for early interconnect planning and design for high performance chips. In addition, it provides mathematical models to establish several performance matrices such as total wirelength, average length of wire, and number of global and semi-global wires. Zhang [72], Banerjee [1], Rahman [73] and Joyner [79] have previously derived statistical wirelength distribution models applicable for square shaped 3-D chips only.

Since the module shapes are generally rectangular in the floorplan, it is necessary to develop a *rectangular* 3-D wirelength distribution model for fast wirelength estimation. Zhang, Banerjee and Rahman's models have one thing in common – all of them were derived from Davis' square shaped 2-D model [71], which is computationally fast. Joyner derived a new gate pair [79] expression that could be used to derive a rectangular model. However, no further work was done to complete the rectangular 3-D model and performance matrices such as total wirelength, average wirelength, etc. Joyner's square shaped 3-D model is also complex and computationally expensive due to lack of a closed form analytical solution.

Therefore we have chosen Zhang, Banerjee and Rahman's models for further study and extension, with an objective of *fast* wirelength prediction inside rectangular modules.

B.2 GENERAL BACKGROUND

The derivation of previous Zhang, Banerjee and Rahman's 3-D wirelength distribution models for square shaped chips are based on Davis' 2-D wirelength model. Davis' 2-D wirelength distribution model was derived using a terminal-gate relationship known as Rent's rule [74] as follows:

$$T = kN^{p} \tag{B.1}$$

where, T is the number of I/O terminals of chip, N is the total number of gates, k is Rent's coefficient and p is Rent's exponent. The Rent's exponent p denotes the complexity of wiring inside a chip and its value is a real number between 0 and 1. Zero denotes absolutely no wiring and one denotes maximum wiring such as a clique in a graph in which every node is connected to the remaining nodes using edges.

Davis assumed all the gates to be organized in a uniform array of square logic gates inside the square 2-D chip. He calculated the total number of gate pairs separated by a given manhattan distance ℓ inside the array. Based on the conservation of I/O terminals [13], he also calculated the expected number of interconnects between these gate pairs. In short form, his distribution function can be written as eqn (B.2):

$$f_{2D}(\ell) = \Gamma.M(\ell).I_{\exp}(\ell) \tag{B.2}$$

where, Γ is a normalization constant, $M(\ell)$ is the total number of gate pairs [71] in gate pitch unit and $I_{\exp}(\ell)$ is the expected number of interconnects between these gate pairs as shown in eqn (B.3) and (B.4).

$$\begin{aligned} M(\ell) &= \frac{\ell^3}{3} - 2\ell^2\sqrt{N} + 2N\ell, \qquad 1 \le \ell < \sqrt{N} \\ &= \frac{1}{3}\left(2\sqrt{N} - \ell\right)^3, \qquad \sqrt{N} \le \ell < 2\sqrt{N} \end{aligned} \tag{B.3}$$

$$\begin{aligned} I_{\exp}(\ell) &= \frac{\alpha k}{N_C}\left[\left(N_A + N_B\right)^p - \left(N_B + N_C\right)^p - \left(N_A + N_B + N_C\right)^p\right] \\ &\approx \alpha k p(1-p)\ell^{2p-4} \end{aligned} \tag{B.4}$$

where, $N_A = 1$, $N_B \approx \ell(\ell-1)$, $N_C \approx 2\ell$ and α is a function of average fanout (*f.o.*) and equals to (*f.o.*/(*f.o.*+1)); k and p are Rent's parameters.

3-D distribution models derived from Davis' model consist of a horizontal distribution model and a vertical distribution model. The horizontal component gives the distribution within each layer while the vertical component gives via distribution across different layers. Banerjee et al. [1] used the direct application of Davis'. 2-D model and calculated the horizontal distribution. Banerjee transformed the Rent's coefficient 'k' of a 2-D chip to *effective external Rent's coefficient*, k_{eff} for each layer of a 3-D chip. The transformation is based on the conservation of I/O terminals [71]. Banerjee also calculated the total number of vertical vias. However, his method does not provide via distribution across different layers. The Rent's exponent p remains unchanged under the transformation from a 2-D to 3-D

model assuming that the routing algorithm remains the same [1]. Banerjee's distribution model can be summarized mathematically as follows:

$$
\begin{aligned}
k_{\text{eff}} &= km^{p-1} \\
f_{\text{int}(B)}(m) &= mk(1-m^{p-1})(N/m)^{p} \\
f_{3D(B)}(\ell) &= mf_{2D}(\ell) + f_{\text{int}(B)}(m)
\end{aligned}
\qquad \text{(B.5)}
$$

where m is the total number of layers, (k,p) are Rent's parameters of the entire chip and k_{eff} is the Rent's coefficient of each layer. Please note that $f_{2\text{-}D}$ for each layer will use k_{eff} as input instead of k. Also f_{int} is constant and only gives the total number of inter-layer vias instead of their distribution.

Zhang et al. split the $I_{\exp}(\ell)$ into vertical and horizontal components, and derived the horizontal and vertical 3-D wirelength distribution models without transforming the original Rent's parameter of a 2-D chip [72]. Zhang's model is summarized as shown in eqn (B.6):

$$
\begin{aligned}
& f_{3D(Zhang)}(\ell) = h(\ell) + v(z) \\
& h(\ell) = \Theta\left(\frac{\ell^3}{3} - 2\ell^2\sqrt{\frac{N}{m}} + 2\ell\frac{N}{m}\right)\ell^{2p-4} \qquad 1 \le \ell < \sqrt{\frac{N}{m}} \\
& \quad = \frac{\Theta}{3}\left(2\sqrt{\frac{N}{m}} - \ell\right)^3 \ell^{2p-4} \qquad \sqrt{\frac{N}{m}} \le \ell \le 2\sqrt{\frac{N}{m}} \\
& where, \Theta \text{ is normalization constant} \\
& v(z) = \frac{\alpha kN(1 - N^{p-1} - m^{p-2} + m^{-1}N^{p-1})}{m(m-1)}(2m - 2z) \\
& where \;\; z = 1, 2, \ldots, m-1
\end{aligned}
\qquad \text{(B.6)}
$$

where m is the total number of device layers in a 3-D IC.

The third model was developed by Rahman et al. [73],[80]. Rahman initially modified eqn (B.4) by incorporating the effect of vertical wire distribution in N_B and N_C as follows:

$$N_B = \frac{1}{N_z}\left[N_z\ell(\ell-1) + 2\sum_{j=1}^{N_z-1}\left[\sum_{i=1}^{j}(\ell - it_z)(\ell - it_z - 1)u(\ell - it_z) \right]\right]$$
$$N_C = \frac{1}{N_z}\left[2N_z\ell + 4\sum_{j=1}^{N_z-1}\left[\sum_{i=1}^{j}(\ell - it_z)u(\ell - it_z) \right]\right]$$

(B.7)

where t_z is via height (in gatepitch) , u is the unit step function and $N_Z = m$ = total number of device layers. Using eqn (B.7) and eqn (B.4), the expected number of interconnects ($I_{3\text{-}D,exp}$) is obtained. Rahman's distribution model is finally derived as follows:

$$f_{3D}(\ell,t_z) = \Gamma' M_{3D}(\ell,t_z) I_{3D,\exp}(\ell,t_z)$$
$$M_{3D}(\ell,t_z) = N_z M(\ell) + \beta M(\ell)$$

(B.8)

where β is a constant which depends on N_Z.

B.3 RECTANGULAR 2-D WIRELENGTH DISTRIBUTION MODEL

As the previously described 3-D models [1],[72],[73] used Davis' square 2-D model, we first modified Davis' model to a rectangular 2-D model and incorporated the aspect ratio $a_r (0 < a_r \leq 1)$ of chip/module in our derivation. We assume that all the logic gates inside the

rectangular module are uniformly arranged in a rectangular array as shown in Figure B.1. The summary of our derivation is as follows eqn (B.9):

$$f_{R,2D}(\ell) = \Lambda.M_R(\ell).I_{\exp}(\ell) \tag{B.9}$$

where, Λ is a normalization constant, $M_R(\ell)$ is the number of gate pairs inside the rectangular array for a given length ℓ and $I_{\exp}(\ell)$ is expected number of interconnection as defined in eqn (B.4). In Figure B.1, The gate count in a row and column defines *length* and *width* of the array respectively. We define aspect ratio a_r such that:

$$\begin{aligned} a_r &= \frac{width}{length} \; ; \; 0 < a_r \leq 1.0 \\ width &= \sqrt{a_r N} \; ; \; length = \sqrt{N/a_r} \end{aligned} \tag{B.10}$$

where, *length* and *width* are integer values in gate pitch unit. The gate pitch is calculated using the area of a rectangular module A_m and the number of gates N inside the module as follows:

$$gate\ pitch = \sqrt{A_m/N} \tag{B.11}$$

To calculate $M_R(\ell)$, we traverse in the rectangular array of logic gates row-wise from top left as shown in Figure B.1. We choose a reference gate (shown as "A" in Figure B.1) and count the total number of gates which are at a manhattan distance ℓ away from it. Then we remove the reference gate from further counting and set a new reference right next to it. We keep counting and adding the number of such gates until all the gates are traversed in

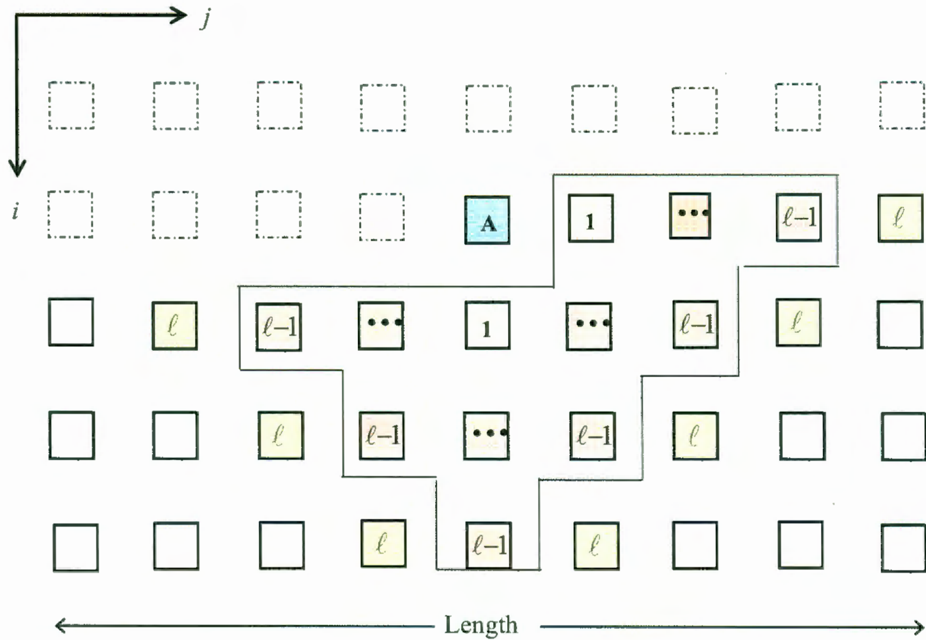


Figure B.1: Gate pair calculation in a partial manhattan circle.

the array. The final sum gives $M_R(\ell)$. In Figure B.1, the dotted gates indicate that they have already been traversed and removed to avoid double counting. The reference gate is called type "A", the gates located at a distance ℓ are called type "C" and all other gates residing between type A and type C are known as type B. These three types of gate form a partial manhattan circle [71] together. The traversal procedure of calculating $M_R(\ell)$ is mathematically defined as:

$$M_R(\ell) = \sum_{i=1}^{\sqrt{a_r N}} \sum_{j=1}^{\sqrt{\frac{N}{a_r}}} \Phi_R(i,j,\ell) \tag{B.12}$$

where $\Phi_R(i,j,\ell)$ computes the total number of gates that are a distance ℓ away from the reference gate "A" located in the ith row and j[th] column of the rectangular array. The function Φ_R is defined as follows:

$$\Phi_R(i,j,\ell) = \begin{pmatrix} (\ell+1)u_0(\ell+1) \ - \ (\ell-\sqrt{\frac{N}{a_r}}+j)u_0(\ell-\sqrt{\frac{N}{a_r}}+j) \\ -\ 2(\ell-\sqrt{a_rN}+i)u_0(\ell-\sqrt{a_rN}+i) \ - \ u_0[-\ (\ell-\sqrt{a_rN}+i)] \\ +\ [\ell-(\sqrt{a_r}+\frac{1}{\sqrt{a_r}})\sqrt{N}+j+i-1]u_0[\ell-(\sqrt{a_r}+\frac{1}{\sqrt{a_r}})\sqrt{N}+j+i-1] \\ +\ (\ell)u_0(\ell) \ - \ (\ell-j)u_0(\ell-j)+(\ell-\sqrt{a_rN}-j+i)u_0(\ell-\sqrt{a_rN}-j+i) \end{pmatrix}$$

(B.13)

where u_0 is the unit step function.

The procedure of calculating $M_R(\ell)$ using computer simulation has *O(length* x *width)* runtime inside the rectangular array of logic gates, which is computationally expensive for large number of gates. Therefore we have derived a closed form analytical solution of $M_R(\ell)$ as shown in eqn (B.14), which can be computed in constant time:

Region I: $1 \le \ell < \sqrt{a_rN}$

$$M_R(\ell) = \frac{\ell^3}{3} - \left(\sqrt{a_rN} + \sqrt{\frac{N}{a_r}}\right)\ell^2 + \frac{1}{3}\ell(6N-1)$$

Region II : $\sqrt{a_rN} \le \ell < \sqrt{N/a_r}$

$$M_R(\ell) = -\left(\sqrt{a_rN}\right)^2 \ell + N\sqrt{a_rN}\left(\frac{a_r}{3}+1\right)$$

Region III : $\sqrt{N/a_r} \le \ell < \left(\sqrt{N/a_r} + \sqrt{a_r N} - 2\right)$

$$M_R(\ell) = \begin{bmatrix} -\dfrac{\ell^3}{3} + \left(\sqrt{a_r N} + \sqrt{\dfrac{N}{a_r}}\right)\ell^2 - \dfrac{1}{3}\ell(12N-1) \\ +\dfrac{1}{3}\left(\sqrt{a_r N} + \sqrt{\dfrac{N}{a_r}}\right)\left(\sqrt{a_r N} + \sqrt{\dfrac{N}{a_r}} - 1\right)\left(\sqrt{a_r N} + \sqrt{\dfrac{N}{a_r}} + 1\right) \end{bmatrix}$$

(B.14)

Assuming $\sqrt{N/a_r} + \sqrt{a_r N} >> 1$ and $N >> 1$, eqn (B.14) is finally approximated as shown in eqn (B.15):

Region I: $1 \le \ell < \sqrt{a_r N}$

$$M_R(\ell) = \frac{\ell^3}{3} - \left(\sqrt{a_r N} + \sqrt{N/a_r}\right)\ell^2 + 2\ell N$$

Region II : $\sqrt{a_r N} \le \ell < \sqrt{N/a_r}$

$$M_R(\ell) = -\left(\sqrt{a_r N}\right)^2 \ell + N\sqrt{a_r N}\left(\frac{a_r}{3} + 1\right)$$

Region III: $\sqrt{N/a_r} \le \ell < \sqrt{N/a_r} + \sqrt{a_r N}$

$$M_R(\ell) = \frac{1}{3}\left(\sqrt{a_r N} + \sqrt{N/a_r} - \ell\right)^3$$

(B.15)

A comparison between the closed form analytical solution obtained using eqn (B.15) and computer simulated gate-pair counts for different aspect ratios is shown in Figure B.2. It indicates that our approximate gate pair count matches closely with the computer simulated data.

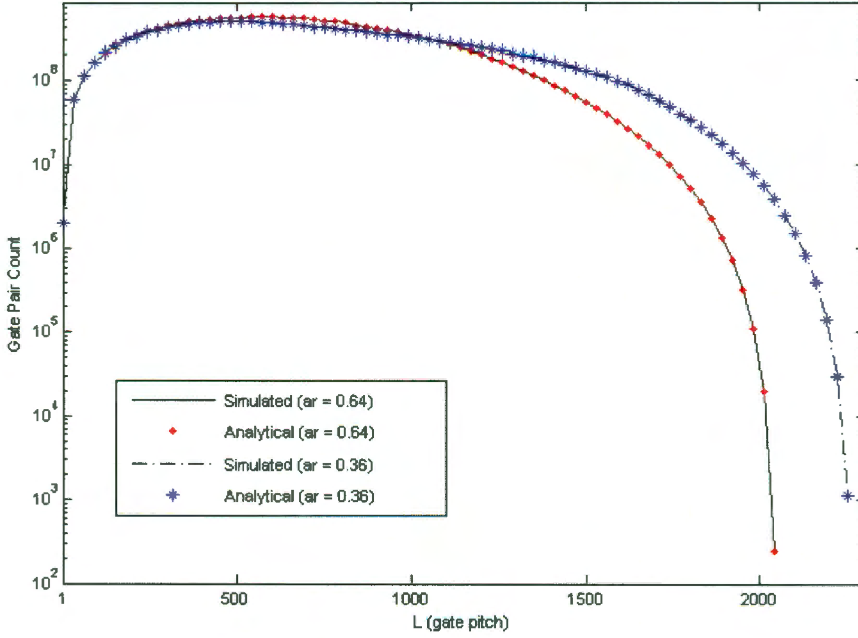


Figure B.2: Comparison of closed form analytical solution and computer simulation of $M_R(L)$, for different aspect ratios.

The total number of interconnects in a system for the known values of k, p, α and N is given by [81]:

$$I_{\text{total}} = \alpha k N \left(1 - N^{p-1}\right) \tag{B.16}$$

Now, the normalization coefficient, Λ is calculated by eqn (B.17)

$$\int_{\ell=1}^{\sqrt{a_r N}+\sqrt{N/a_r}} f_{R,2D}(\ell)d\ell = I_{\text{total}} \tag{B.17}$$

Thus,

$$\Lambda = \frac{2N(1-N^{p-1})}{\left(\frac{N^p.\Psi}{6p(p-1)(2p-1)(2p-3)} - \frac{1}{6p} + \frac{\left(\sqrt{a_r N}+\sqrt{N/a_r}\right)}{2p-1} - \frac{N}{p-1}\right)} \tag{B.18}$$

where ψ, a function of Rent's exponent p and aspect ratio a_r is given by eqn (B.19):

$$\Psi = \begin{bmatrix} (2p-3)(p-1)(2p-1)a_r^p - 6p(2p-3)(p-1)(a_r^p + a_r^{p-1}) \\ + 3p(2p-3)(2p-1)(a_r - a_r^{2-p} + 2a_r^{p-1}) \\ + 2p(p-1)(2p-1)(3a_r^{2-p} - 3a_r^{p-1} - a_r^p) \\ +3(a_r + a_r^{-1} + 2)^p - 3a_r^{-p} - 3p(2p-1)a_r^{2-p} - 6pa_r^{1-p} \end{bmatrix} \tag{B.19}$$

At p = 0.5, Λ becomes indeterminate of the form 0/0, but by using L'Hopital rule at p = 0.5, Λ converges to eqn (B.20).

$$\Lambda = \frac{4N - 4\sqrt{N}}{\left(\sqrt{a_r N} + \sqrt{\frac{N}{a_r}}\right)\left[-\ln(N) + \ln\left(\sqrt{a_r} + \frac{1}{\sqrt{a_r}}\right)^2 - 1\right] - 2\sqrt{N}\left(a_r + \frac{1}{\sqrt{a_r}}\right) + 4N - \frac{2}{3}} \tag{B.20}$$

This completes the derivation of a rectangular 2-D wirelength distribution. A performance metric, *length demand function* $D(\ell)$ that gives the total length of interconnects between a length of one gate pitch to ℓ gate pitches [21], is calculated as shown in eqn (B.21):

$$D(\ell) = \int_{1}^{\ell} \ell' . f_{R,\ 2D}(\ell') d\ell' \tag{B.21}$$

The complete evaluation of eqn (B.21) gives eqn (B.22) as shown below:

Region I: $1 \le \ell < \sqrt{a_r N}$

$$D(\ell) = \frac{\alpha k}{2} \Lambda \left(\frac{1}{3} \frac{\ell^{2p+1} - 1}{(2p+1)} - \left(\sqrt{a_r N} + \sqrt{N/a_r} \right) \frac{\ell^{2p} - 1}{2p} + 2N \frac{\ell^{2p-1} - 1}{2p-1} \right)$$

Region II : $\sqrt{a_r N} \le \ell < \sqrt{N/a_r}$

$$D(\ell) = \frac{\alpha k}{2} \Lambda \left(\begin{array}{l} \frac{1}{3} \frac{(\sqrt{a_r N})^{2p+1} - 1}{(2p+1)} - \left(\sqrt{a_r N} + \sqrt{N/a_r} \right) \frac{(\sqrt{a_r N})^{2p} - 1}{2p} + 2N \frac{(\sqrt{a_r N})^{2p-1} - 1}{2p-1} \\ - a_r N \frac{[\ell^{2p-1} - (\sqrt{a_r N})^{2p-1}]}{2p-1} + N\sqrt{a_r N} \left(\frac{a_r}{3} + 1 \right) \frac{[\ell^{2p-2} - (\sqrt{a_r N})^{2p-2}]}{2p-2} \end{array} \right)$$

Region III: $\sqrt{N/a_r} \le \ell < \sqrt{N/a_r} + \sqrt{a_r N}$

$$D(\ell) = \frac{\alpha k}{2} \Lambda \left(\begin{array}{l} \frac{1}{3} \frac{(\sqrt{a_r N})^{2p+1} - 1}{2p+1} - \left(\sqrt{a_r N} + \sqrt{N/a_r} \right) \frac{(\sqrt{a_r N})^{2p} - 1}{2p} + 2N \frac{(\sqrt{a_r N})^{2p-1} - 1}{2p-1} \\ - a_r N \frac{(\sqrt{N/a_r})^{2p-1} - (\sqrt{a_r N})^{2p-1}}{2p-1} + N\sqrt{a_r N} \left(\frac{a_r}{3} + 1 \right) \frac{(\sqrt{N/a_r})^{2p-2} - (\sqrt{a_r N})^{2p-2}}{2p-2} \\ + \frac{1}{3} \left(\sqrt{a_r N} + \sqrt{N/a_r} \right)^3 \frac{\ell^{2p-2} - (\sqrt{N/a_r})^{2p-2}}{2p-2} - \frac{1}{3} \frac{\ell^{2p+1} - (\sqrt{N/a_r})^{2p+1}}{2p+1} \\ + \left(\sqrt{a_r N} + \sqrt{N/a_r} \right) \frac{\ell^{2p} - (\sqrt{a_r N})^{2p}}{2p} - \left(\sqrt{a_r N} + \sqrt{N/a_r} \right)^2 \frac{\ell^{2p-1} - (\sqrt{a_r N})^{2p-1}}{2p-1} \end{array} \right)$$

(B.22)

B.4 EXTENSION TO RECTANGULAR 3-D WIRELENGTH DISTRIBUTION MODELS

Since we extend the previous 3-D models of square shape to rectangular shape, our models are referred with the prefix "extended" during this discussion. Using our rectangular 2-D distribution model, we have extended Banerjee's 3-D square model to a 3-D rectangular model by modifying eqn (B.5) using eqn (B.9) as follows:

$$f_{R,3D(B)}(\ell) = mf_{R,2D}(\ell) + f_{\text{int}(B)}(m) \tag{B.23}$$

The modification in Zhang's model will only be in horizontal wirelength distribution while the vertical distribution will remain unchanged even for rectangular shape. Thus we modified the horizontal distribution of eqn (B.6) as follows:

$$h_R(\ell) = \begin{cases} \Theta_R[\frac{\ell^3}{3} - (\sqrt{a_r N/m} + \sqrt{N/ma_r})\ell^2 + 2\ell N/m]\ell^{2p-4}; & 1 \le \ell < \sqrt{a_r N/m} \\ \Theta_R[-\frac{a_r N\ell}{m} + \frac{N}{m}(\frac{a_r}{3}+1)\sqrt{\frac{a_r N}{m}}]\ell^{2p-4}; & \sqrt{a_r N/m} \le \ell < \sqrt{N/ma_r} \\ \frac{\Theta_R}{3}[\sqrt{a_r N/m} + \sqrt{N/ma_r} - \ell]^3 \ell^{2p-4}; & \sqrt{N/ma_r} \le \ell < \sqrt{N/ma_r} + \sqrt{a_r N/m} \end{cases} \tag{B.24}$$

Thus the extended Zhang's model appears as:

$$f_{R,3D(Zhang)}(\ell) = h_R(\ell) + v(z) \tag{B.25}$$

where $v(z)$ is same as defined in eqn (B.6). The normalization constant, Θ_R is given by (B.26)

$$\Theta_R = \frac{\alpha k m^p (N/m)[1-(N/m)^{p-1}]}{\frac{(N/m)^p.\Psi}{6p(p-1)(2p-1)(2p-3)} - \frac{1}{6p} + \frac{(\sqrt{a_r N/m}+\sqrt{N/ma_r})}{2p-1} - \frac{(N/m)}{p-1}} \quad (B.26)$$

At $p = 0.5$, Θ_R becomes indeterminate of the form 0/0. Using L'Hopital rule at $p = 0.5$, Θ_R becomes as follows:

$$\Theta_R = \frac{2\alpha k.m^p\left(N/m-\sqrt{N/m}\right)}{(\sqrt{a_r N/m}+\sqrt{N/ma_r})[-\ln\frac{N}{m} + \ln[\sqrt{a_r}+\frac{1}{\sqrt{a_r}}]^2-1] - 2\sqrt{N/m}[\sqrt{a_r}+\frac{1}{\sqrt{a_r}}] + 4\frac{N}{m}-\frac{2}{3}}$$

(B.27)

Finally, Rahman's model is extended to a 3-D rectangular model by modifying eqn (B.8) with the use of eqn (B.9) and the modified model is represented by eqn (B.28).

$$\begin{aligned} f_{R,3D}(\ell,t_z) &= \Gamma' M_{R,3D}(\ell,t_z) I_{3D,\exp}(\ell,t_z) \\ M_{R,3D}(\ell,t_z) &= N_z M_R(\ell) + \beta M_R(\ell) \end{aligned} \quad (B.28)$$

B.5 EXPERIMENTAL RESULTS

We implemented our newly developed 3-D rectangular models in C++/STL to generate data for experimental results. These results are also valid for the original 3-D models of square chips because they are special cases of our models ($a_r = 1$). Our models are referred with the prefix "extended" during the discussion.

B.5.1 Effect of Aspect Ratio on Wirelength Distribution and Total Wirelength

We started by plotting a graph to observe the impact of aspect ratio on our newly developed 2-D wirelength distribution model which was later extended to the 3-D domain. Figure B.3 indicates that a reduction in aspect ratio will increase the global wire count but decrease the semi-global wire count. We can see these dual trends as the two curves cross each other in Figure B.3. When we plot the total wirelength against various aspect ratios, we observe that the total wirelength decreases with decreasing aspect ratio for a_r < 0.6 as shown in Figure B.4. However, it saturates for $0.6 < a_r \leq 1$. This trend is seen in 3-D circuits

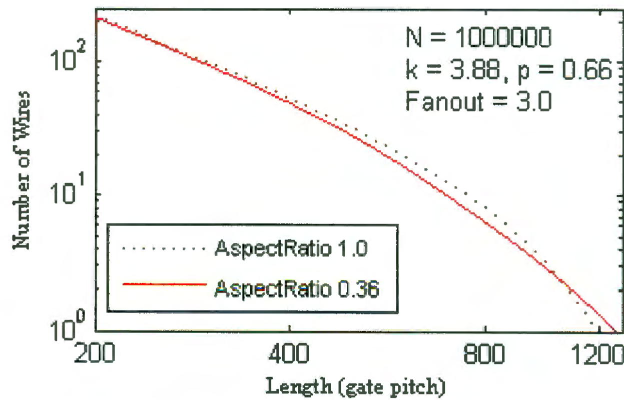


Figure B.3: Effect of aspect ratio on 2-D rectangular wirelength distribution.

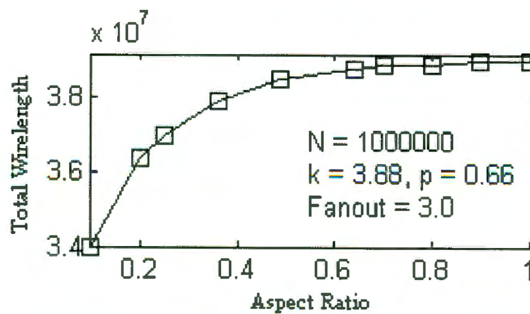


Figure B.4: Effect of varying aspect ratio on total wirelength (in gate pitch).

as well. Therefore a smaller aspect ratio (less than 0.6) is preferred for the total wirelength reduction.

B.5.2 Comparison Among Rectangular 3-D Wirelength Distribution Models

The comparison between extended Banerjee's and Zhang's models shows that their horizontal components converge in spite of different derivation approaches. Figure B.5 indicates this convergence. Extended Banerjee's model however does not provide the distribution of vertical vias among different layers. Extended Zhang's model gives the distribution of vias, but it does not tell which via is associated with which net/wire. Extended Rahman's model takes care of this issue. It considers via as a part of a particular net and calculates the length of the net. A separate comparison between extended Zhang's model and extended Rahman's model is presented in Figure B.6. It can be seen that the extended Rahman's model predicts lesser global and semi-global wire compared to the extended Zhang's model. However the number of local wires is larger than Zhang's model due to the assumption of *direct vertical integration* [73],[80] in Rahman's derivation. In the integration, any gate is allowed to be connected in different layers. Therefore many global and semi-global wires are replaced by local wires and vias. Rahman's approach is computationally expensive due to the lack of a closed form analytical solution because of changes imposed by eqn (B.7) and eqn (B.8). However, it gives a closer prediction of the wirelength distribution compared to Banerjee and Zhang's models. The summary of our comparison is presented in Table B.1.

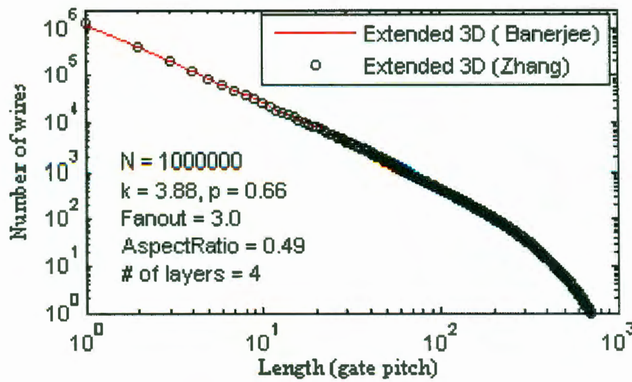


Figure B.5: Extended Zhang's and Banerjee's horizontal 3-D models.

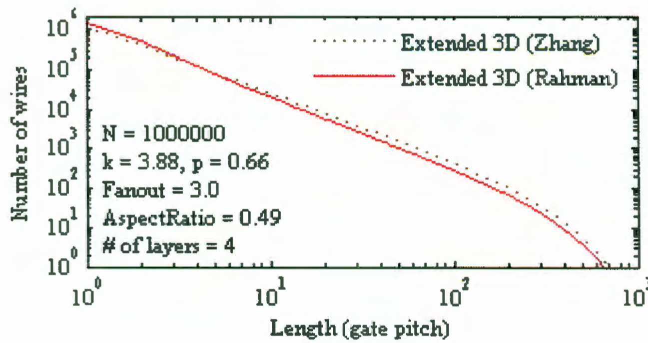


Figure B.6: Comparison of extended Zhang's and Rahman's 3-D models.

TABLE B.1: COMPARISON OF VARIOUS RECTANGULAR 3-D WIRELENGTH DISTRIBUTION MODELS

Rectangular 3-D models derived from :	Analytical solution available?	Via distribution available?	Relative computation time	Relative accuracy
Banerjee's	Yes	No	Fastest	Low
Zhang's	Yes	Yes	Faster	Medium
Rahman's	No	Yes	Slower	High

APPENDIX C: RENT'S PARAMETER EXTRACTION

C.1 EXTRACTION OF RENT'S PARAMETERS FROM THE NETLIST OF A CIRCUIT BLOCK

Let us assume that we are given the netlist of a circuit block for which Rent's parameters are needed to be extracted. The first step is to perform a k-way partition [105]of the netlist with multilevel recursive bisection using a circuit partitioner such as hMetis [106]. For these partitions, k is given by the list {2, 4, 8, 16, 32, ...} with the highest value in the sequence chosen such that no partition contains zero gates. As a result of the multilevel recursive circuit partitioning, a list of I/O terminals (T) and number of gates (N) for each partition is obtained. Next a log-log plot of T vs. N for each partition is plotted and a least square linear fit is obtained from the plot. The linear fit can be expressed as:

$$\mathrm{Log}(T) = p\,\mathrm{Log}(N) + \mathrm{Log}(k) \tag{C.1}$$

$$\Rightarrow \quad T = k\,N^{p} \tag{C.2}$$

Thus, from the least square linear fit plot, the values of k and p are obtained.

C.2 RENT PARAMETER FOR MODULES OF FLOORPLAN BENCHMARKS

For the placement-aware 3D floorplanning experiments, we randomly assigned Rents parameter to modules assuming that the module represented different types of circuits such as microprocessor, SRAM, ASIC like control logic circuit etc. The Rent's parameters for

different types of circuits were directly taken from [77]. Tables C.1 to C.5 show the Rent's parameter assigned to different modules in each floorplanning benchmark.

TABLE C.1: RENT PARAMETERS FOR EACH MODULE OF AMI33 BENCHMARK

ami33			ami33			ami33		
Module	*k*	*p*	Module	*k*	*p*	Module	*k*	*p*
0	6	0.12	11	23.3	0.3	22	0.82	0.45
1	1.4	0.63	12	7.3	0.46	23	2.09	0.36
2	1.4	0.63	13	2.2	0.66	24	1.4	0.63
3	0.82	0.45	14	1.4	0.63	25	1.4	0.63
4	2.09	0.36	15	0.82	0.45	26	1.4	0.63
5	3.8	0.75	16	2.14	0.8	27	2.2	0.66
6	2.14	0.8	17	2.09	0.36	28	7.3	0.46
7	0.8	0.69	18	3.8	0.75	29	4.4	0.61
8	2.2	0.66	19	6	0.12	30	0.8	0.69
9	4.4	0.61	20	0.82	0.45	31	0.8	0.69
10	20.5	0.3	21	2.09	0.36	32	7.3	0.46

TABLE C.2: RENT PARAMETERS FOR EACH MODULE OF AMI49 BENCHMARK

ami49			ami49			ami49		
Module	*k*	*p*	Module	*k*	*p*	Module	*k*	*p*
0	3.8	0.75	17	1.4	0.63	34	0.82	0.45
1	1.4	0.63	18	2.09	0.36	35	2.09	0.36
2	2.2	0.66	19	3.8	0.75	36	2.09	0.36
3	2.14	0.8	20	1.4	0.63	37	3.8	0.75
4	2.09	0.36	21	1.4	0.63	38	1.4	0.63
5	3.8	0.75	22	0.82	0.45	39	1.4	0.63
6	2.14	0.8	23	3.8	0.75	40	1.4	0.63
7	0.8	0.69	24	4.4	0.61	41	0.82	0.45
8	2.2	0.66	25	0.8	0.69	42	2.2	0.66
9	4.4	0.61	26	3.8	0.75	43	2.14	0.8
10	20.5	0.3	27	2.09	0.36	44	2.14	0.8
11	23.3	0.3	28	1.4	0.63	45	4.4	0.61
12	7.3	0.46	29	3.8	0.75	46	7.3	0.46
13	2.2	0.66	30	2.14	0.8	47	0.82	0.45
14	0.8	0.69	31	2.2	0.66	48	1.4	0.63
15	1.9	0.5	32	2.09	0.36	-----		
16	6	0.12	33	1.4	0.63	-----		

TABLE C.3: RENT PARAMETERS FOR EACH MODULE OF N100 BENCHMARK

n100			n100			n100		
Module	*k*	*p*	Module	*k*	*p*	Module	*k*	*p*
0	6	0.12	34	0.82	0.45	68	3.8	0.75
1	1.4	0.63	35	1.4	0.63	69	1.4	0.63
2	1.4	0.63	36	0.8	0.69	70	3.8	0.75
3	0.82	0.45	37	4.4	0.61	71	1.4	0.63
4	0.82	0.45	38	6	0.12	72	20.5	0.3
5	2.09	0.36	39	0.8	0.69	73	0.8	0.69
6	0.8	0.69	40	1.4	0.63	74	6	0.12
7	0.8	0.69	41	0.82	0.45	75	2.09	0.36
8	0.82	0.45	42	0.82	0.45	76	0.82	0.45
9	2.14	0.8	43	1.4	0.63	77	23.3	0.3
10	2.2	0.66	44	1.4	0.63	78	2.09	0.36
11	4.4	0.61	45	0.82	0.45	79	6	0.12
12	7.3	0.46	46	2.2	0.66	80	0.8	0.69
13	2.2	0.66	47	2.2	0.66	81	23.3	0.3
14	0.8	0.69	48	1.9	0.5	82	0.82	0.45
15	6	0.12	49	4.4	0.61	83	1.4	0.63
16	1.4	0.63	50	0.8	0.69	84	3.8	0.75
17	3.8	0.75	51	2.2	0.66	85	3.8	0.75
18	3.8	0.75	52	2.2	0.66	86	1.4	0.63
19	1.4	0.63	53	3.8	0.75	87	1.9	0.5
20	1.4	0.63	54	1.4	0.63	88	1.9	0.5
21	3.8	0.75	55	1.4	0.63	89	2.2	0.66
22	4.4	0.61	56	2.2	0.66	90	0.8	0.69
23	0.8	0.69	57	3.8	0.75	91	0.8	0.69
24	2.09	0.36	58	2.2	0.66	92	3.8	0.75
25	1.4	0.63	59	0.8	0.69	93	0.8	0.69
26	3.8	0.75	60	3.8	0.75	94	2.2	0.66
27	2.2	0.66	61	0.8	0.69	95	3.8	0.75
28	1.4	0.63	62	0.8	0.69	96	2.2	0.66
29	0.82	0.45	63	2.2	0.66	97	7.3	0.46
30	2.09	0.36	64	1.9	0.5	98	2.09	0.36
31	3.8	0.75	65	1.9	0.5	99	0.82	0.45
32	3.8	0.75	66	3.8	0.75	-----		
33	1.4	0.63	67	1.4	0.63	-----		

TABLE C.4: RENT PARAMETERS FOR EACH MODULE OF N200 BENCHMARK

n200			n200			n200		
Module	k	p	Module	k	p	Module	k	p
0	6	0.12	37	4.4	0.61	74	6	0.12
1	1.4	0.63	38	6	0.12	75	2.09	0.36
2	4.4	0.61	39	4.4	0.61	76	0.82	0.45
3	0.82	0.45	40	1.4	0.63	77	23.3	0.3
4	0.82	0.45	41	0.82	0.45	78	2.09	0.36
5	2.09	0.36	42	0.82	0.45	79	6	0.12
6	0.8	0.69	43	1.4	0.63	80	0.8	0.69
7	0.8	0.69	44	1.4	0.63	81	23.3	0.3
8	0.82	0.45	45	0.82	0.45	82	0.82	0.45
9	2.14	0.8	46	2.2	0.66	83	1.4	0.63
10	2.2	0.66	47	2.2	0.66	84	3.8	0.75
11	4.4	0.61	48	1.9	0.5	85	3.8	0.75
12	7.3	0.46	49	4.4	0.61	86	1.4	0.63
13	2.2	0.66	50	0.8	0.69	87	1.9	0.5
14	0.8	0.69	51	2.2	0.66	88	1.9	0.5
15	3.8	0.75	52	2.2	0.66	89	2.2	0.66
16	1.4	0.63	53	3.8	0.75	90	0.8	0.69
17	3.8	0.75	54	1.4	0.63	91	0.8	0.69
18	3.8	0.75	55	1.4	0.63	92	3.8	0.75
19	1.4	0.63	56	2.2	0.66	93	0.8	0.69
20	1.4	0.63	57	3.8	0.75	94	2.2	0.66
21	3.8	0.75	58	2.2	0.66	95	3.8	0.75
22	4.4	0.61	59	0.8	0.69	96	2.2	0.66
23	0.8	0.69	60	3.8	0.75	97	7.3	0.46
24	2.09	0.36	61	0.8	0.69	98	2.09	0.36
25	1.4	0.63	62	0.8	0.69	99	0.82	0.45
26	3.8	0.75	63	2.2	0.66	100	6	0.12
27	2.2	0.66	64	1.9	0.5	101	1.4	0.63
28	1.4	0.63	65	1.9	0.5	102	1.4	0.63
29	0.82	0.45	66	3.8	0.75	103	0.82	0.45
30	2.09	0.36	67	1.4	0.63	104	0.82	0.45
31	3.8	0.75	68	3.8	0.75	105	2.09	0.36
32	3.8	0.75	69	1.4	0.63	106	0.8	0.69
33	1.4	0.63	70	3.8	0.75	107	0.8	0.69
34	0.82	0.45	71	1.4	0.63	108	0.82	0.45
35	1.4	0.63	72	20.5	0.3	109	2.14	0.8
36	0.8	0.69	73	0.8	0.69	110	2.2	0.66

n200 contd.			n200 contd.			n200 contd.		
Module	*k*	*p*	Module	*k*	*p*	Module	*k*	*p*
111	4.4	0.61	142	0.82	0.45	173	0.8	0.69
112	7.3	0.46	143	1.4	0.63	174	6	0.12
113	2.2	0.66	144	1.4	0.63	175	2.09	0.36
114	0.8	0.69	145	0.82	0.45	176	0.82	0.45
115	6	0.12	146	2.2	0.66	177	23.3	0.3
116	1.4	0.63	147	2.2	0.66	178	2.09	0.36
117	3.8	0.75	148	1.9	0.5	179	6	0.12
118	3.8	0.75	149	4.4	0.61	180	0.8	0.69
119	1.4	0.63	150	0.8	0.69	181	23.3	0.3
120	1.4	0.63	151	2.2	0.66	182	0.82	0.45
121	3.8	0.75	152	2.2	0.66	183	1.4	0.63
122	4.4	0.61	153	3.8	0.75	184	3.8	0.75
123	0.8	0.69	154	1.4	0.63	185	3.8	0.75
124	2.09	0.36	155	1.4	0.63	186	1.4	0.63
125	1.4	0.63	156	2.2	0.66	187	1.9	0.5
126	3.8	0.75	157	3.8	0.75	188	1.9	0.5
127	2.2	0.66	158	2.2	0.66	189	2.2	0.66
128	1.4	0.63	159	0.8	0.69	190	0.8	0.69
129	0.82	0.45	160	3.8	0.75	191	0.8	0.69
130	2.09	0.36	161	0.8	0.69	192	3.8	0.75
131	3.8	0.75	162	0.8	0.69	193	0.8	0.69
132	3.8	0.75	163	2.2	0.66	194	2.2	0.66
133	1.4	0.63	164	1.9	0.5	195	3.8	0.75
134	0.82	0.45	165	1.9	0.5	196	2.2	0.66
135	1.4	0.63	166	3.8	0.75	197	7.3	0.46
136	0.8	0.69	167	1.4	0.63	198	2.09	0.36
137	4.4	0.61	168	3.8	0.75	199	0.82	0.45
138	6	0.12	169	1.4	0.63	------		
139	0.8	0.69	170	3.8	0.75	------		
140	1.4	0.63	171	1.4	0.63	------		
141	0.82	0.45	172	20.5	0.3	------		

TABLE C.5: RENT PARAMETERS FOR EACH MODULE OF N300 BENCHMARK

n300			n300			n300		
Module	*k*	*p*	Module	*k*	*p*	Module	*k*	*p*
0	6	0.12	38	6	0.12	76	0.82	0.45
1	1.4	0.63	39	0.8	0.69	77	23.3	0.3
2	1.4	0.63	40	1.4	0.63	78	2.09	0.36
3	0.82	0.45	41	0.82	0.45	79	6	0.12
4	0.82	0.45	42	0.82	0.45	80	0.8	0.69
5	2.09	0.36	43	1.4	0.63	81	23.3	0.3
6	0.8	0.69	44	1.4	0.63	82	0.82	0.45
7	0.8	0.69	45	0.82	0.45	83	1.4	0.63
8	0.82	0.45	46	2.2	0.66	84	3.8	0.75
9	2.14	0.8	47	2.2	0.66	85	3.8	0.75
10	2.2	0.66	48	1.9	0.5	86	1.4	0.63
11	4.4	0.61	49	4.4	0.61	87	1.9	0.5
12	7.3	0.46	50	0.8	0.69	88	1.9	0.5
13	2.2	0.66	51	2.2	0.66	89	2.2	0.66
14	0.8	0.69	52	2.2	0.66	90	0.8	0.69
15	6	0.12	53	3.8	0.75	91	0.8	0.69
16	1.4	0.63	54	1.4	0.63	92	3.8	0.75
17	3.8	0.75	55	1.4	0.63	93	0.8	0.69
18	3.8	0.75	56	2.2	0.66	94	2.2	0.66
19	1.4	0.63	57	3.8	0.75	95	3.8	0.75
20	1.4	0.63	58	2.2	0.66	96	2.2	0.66
21	3.8	0.75	59	0.8	0.69	97	7.3	0.46
22	4.4	0.61	60	3.8	0.75	98	2.09	0.36
23	0.8	0.69	61	0.8	0.69	99	0.82	0.45
24	2.09	0.36	62	0.8	0.69	100	2.2	0.66
25	1.4	0.63	63	2.2	0.66	101	1.4	0.63
26	3.8	0.75	64	1.9	0.5	102	0.8	0.69
27	2.2	0.66	65	1.9	0.5	103	2.2	0.66
28	1.4	0.63	66	3.8	0.75	104	2.2	0.66
29	0.82	0.45	67	1.4	0.63	105	6	0.12
30	2.09	0.36	68	3.8	0.75	106	1.4	0.63
31	3.8	0.75	69	1.4	0.63	107	1.4	0.63
32	3.8	0.75	70	3.8	0.75	108	3.8	0.75
33	1.4	0.63	71	1.4	0.63	109	2.2	0.66
34	0.82	0.45	72	20.5	0.3	110	2.2	0.66
35	1.4	0.63	73	0.8	0.69	111	0.8	0.69
36	0.8	0.69	74	6	0.12	112	4.4	0.61
37	4.4	0.61	75	2.09	0.36	113	1.9	0.5

n300 contd.			n300 contd.			n300 contd.		
Module	*k*	*p*	Module	*k*	*p*	Module	*k*	*p*
114	2.2	0.66	155	2.09	0.36	196	2.2	0.66
115	2.2	0.66	156	0.8	0.69	197	2.2	0.66
116	0.82	0.45	157	0.8	0.69	198	1.9	0.5
117	1.4	0.63	158	0.82	0.45	199	4.4	0.61
118	6	0.12	159	2.14	0.8	200	0.8	0.69
119	1.4	0.63	160	2.2	0.66	201	2.2	0.66
120	1.4	0.63	161	4.4	0.61	202	2.2	0.66
121	0.82	0.45	162	7.3	0.46	203	3.8	0.75
122	0.82	0.45	163	2.2	0.66	204	1.4	0.63
123	2.09	0.36	164	0.8	0.69	205	1.4	0.63
124	0.8	0.69	165	6	0.12	206	2.2	0.66
125	0.8	0.69	166	1.4	0.63	207	3.8	0.75
126	2.14	0.8	167	3.8	0.75	208	2.2	0.66
127	2.2	0.66	168	3.8	0.75	209	0.8	0.69
128	4.4	0.61	169	1.4	0.63	210	3.8	0.75
129	7.3	0.46	170	1.4	0.63	211	0.8	0.69
130	2.2	0.66	171	3.8	0.75	212	0.8	0.69
131	0.8	0.69	172	4.4	0.61	213	2.2	0.66
132	6	0.12	173	0.8	0.69	214	1.9	0.5
133	1.4	0.63	174	2.09	0.36	215	1.9	0.5
134	3.8	0.75	175	1.4	0.63	216	3.8	0.75
135	3.8	0.75	176	3.8	0.75	217	1.4	0.63
136	1.4	0.63	177	2.2	0.66	218	3.8	0.75
137	1.4	0.63	178	1.4	0.63	219	1.4	0.63
138	3.8	0.75	179	0.82	0.45	220	3.8	0.75
139	4.4	0.61	180	2.09	0.36	221	1.4	0.63
140	0.8	0.69	181	3.8	0.75	222	20.5	0.3
141	2.09	0.36	182	3.8	0.75	223	0.8	0.69
142	1.4	0.63	183	1.4	0.63	224	6	0.12
143	3.8	0.75	184	0.82	0.45	225	2.09	0.36
144	2.2	0.66	185	1.4	0.63	226	0.82	0.45
145	1.4	0.63	186	0.8	0.69	227	23.3	0.3
146	0.82	0.45	187	4.4	0.61	228	2.09	0.36
147	2.09	0.36	188	6	0.12	229	6	0.12
148	3.8	0.75	189	0.8	0.69	230	0.8	0.69
149	3.8	0.75	190	1.4	0.63	231	23.3	0.3
150	6	0.12	191	0.82	0.45	232	0.82	0.45
151	1.4	0.63	192	0.82	0.45	233	1.4	0.63
152	1.4	0.63	193	1.4	0.63	234	3.8	0.75
153	0.82	0.45	194	1.4	0.63	235	3.8	0.75
154	0.82	0.45	195	0.82	0.45	236	1.4	0.63

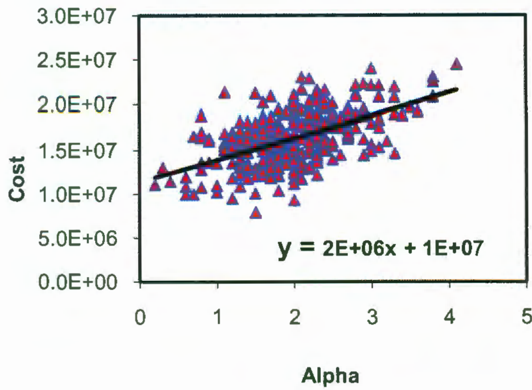
n300 contd.			n300 contd.			n300 contd.		
Module	*k*	*p*	Module	*k*	*p*	Module	*k*	*p*
237	1.9	0.5	258	3.8	0.75	279	7.3	0.46
238	1.9	0.5	259	2.2	0.66	280	2.2	0.66
239	2.2	0.66	260	2.2	0.66	281	0.8	0.69
240	0.8	0.69	261	0.8	0.69	282	6	0.12
241	0.8	0.69	262	4.4	0.61	283	1.4	0.63
242	3.8	0.75	263	1.9	0.5	284	3.8	0.75
243	0.8	0.69	264	2.2	0.66	285	3.8	0.75
244	2.2	0.66	265	2.2	0.66	286	1.4	0.63
245	3.8	0.75	266	0.82	0.45	287	1.4	0.63
246	2.2	0.66	267	1.4	0.63	288	3.8	0.75
247	7.3	0.46	268	6	0.12	289	4.4	0.61
248	2.09	0.36	269	1.4	0.63	290	0.8	0.69
249	0.82	0.45	270	1.4	0.63	291	2.09	0.36
250	2.2	0.66	271	0.82	0.45	292	1.4	0.63
251	1.4	0.63	272	0.82	0.45	293	3.8	0.75
252	0.8	0.69	273	2.09	0.36	294	2.2	0.66
253	2.2	0.66	274	0.8	0.69	295	1.4	0.63
254	2.2	0.66	275	0.8	0.69	296	0.82	0.45
255	6	0.12	276	2.14	0.8	297	2.09	0.36
256	1.4	0.63	277	2.2	0.66	298	3.8	0.75
257	1.4	0.63	278	4.4	0.61	299	3.8	0.75

APPENDIX D: SENSITIVITY ANALYSIS OF THE COST FUNCTION OF FLOORPLANNING ALGORITHMS

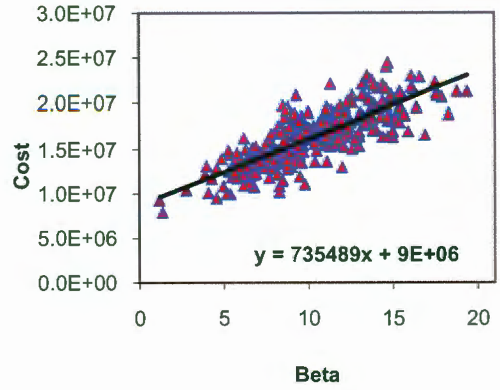
D.1 SENSITIVY ANALYSIS OF THE COST FUNCTION OF 3-D FVC ALGORITHM WITHOUT MODULE SPLITTING

In this section, we perform the sensitivity analysis of the cost function when module splitting is *disabled*. In this case, the cost is a function of dead space, inter-module wirelength, and inter-module via count as shown by eqn. (4.2). The tuning parameters α, β, and γ_1 are independent of each other and we would like to find how sensitive the cost function is with respect to these parameters.

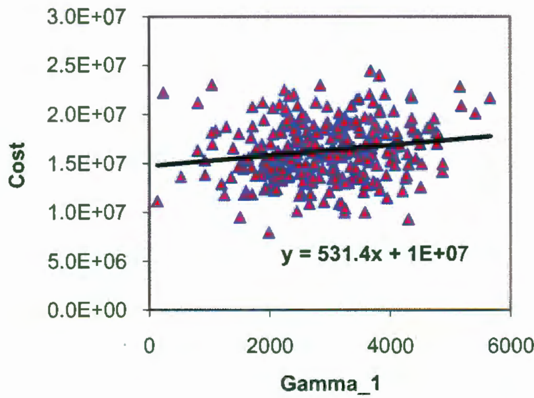
We use the sampling based method for the sensitivity analysis of the cost function. Assuming a Gaussian distribution, we randomly generate 300 samples for each of the three tuning parameters. The σ and μ values for each tuning parameter are denoted by using the parameter's name as a sub-script for σ and μ, and their values are chosen as follows *a)* $\sigma_\alpha = 2.0$, $\mu_\alpha = 0.32$, *b)* $\sigma_\beta = 10.0$, $\mu_\beta = 3.2$, and *c)* $\sigma_{\gamma 1} = 3000$, $\mu_{\gamma 1} = 990$. Please note that for the sampling based sensitivity analysis approach [108],[109], the ranges of the parameters are assumed to be known and in our experiment, we have chosen their ranges based on our experience with floorplan experiments performed on various benchmarks. Using these sampled values of tuning parameters, we run 3D-FVC without module splitting on a floorplan benchmark (*n100*) that contains 100 modules. Next we plotted scatter plots of cost vs. each individual tuning parameter, and thus obtained three scatter plots corresponding to α, β, and γ_1. Finally we used linear regression to obtain a best fit linear



(a)



(b)



(c)

Figure D.1: Scatter plots for sensitivity analysis of the tuning parameters of the cost function for 3-FVC without module splitting.

relation for cost vs. an individual tuning parameter. The scatter plots are shown in Figure D.1. Please note that the linear relation represented by a solid line, and the equation representing the line are on each of the three scatter plots.

The equations of the straight lines are given as follows:

$$Cost = 2\times10^{6}\alpha + 10^{7} \quad (D.1)$$

$$Cost = 7.35\times10^{5}\beta + 9\times10^{6} \quad (D.2)$$

$$Cost = 531.4\,\gamma_1 + 10^{7} \quad (D.3)$$

Performing the partial differentiation of cost with respect to each individual tuning parameter gives:

$$\frac{\partial(Cost)}{\partial\alpha} = 2\times10^{6} \quad (D.4)$$

$$\frac{\partial(Cost)}{\partial\beta} = 7.35\times10^{5} \quad (D.5)$$

$$\frac{\partial(Cost)}{\partial\gamma_1} = 531.4 \quad (D.6)$$

From the slope obtained after the partial differentiation, it can be observed that cost is most sensitive to α, then β. Finally, cost is the least sensitive to γ_1.

Since, the 3-D floorplanner minimizes the cost, it would be desirable to keep the values of α and β low. From Figure D.1 it is obvious that cost can be kept minimal if we keep α within the range of 0.5 to 2.0. Similarly, β should be kept in the range of 1 to 10 for minimal cost. Since the cost is least sensitive to γ_1, its value can be chosen within 1000 to 5000.

D.2 SENSITIVY ANALYSIS OF THE COST FUNCTION OF 3-D FVC ALGORITHM WITH MODULE SPLITTING

Once the module splitting in 3-D FVC is enabled, the cost function changes to eqn (4.3) and the fourth tuning parameter γ_2 is introduced. The distribution of each tuning parameter using Gaussian distribution is described as *a)* $\sigma_\alpha = 2.0$, $\mu_\alpha = 0.32$, *b)* $\sigma_\beta = 10.0$, $\mu_\beta = 3.2$, *c)* $\sigma_{\gamma 1} = 3000$, $\mu_{\gamma 1} = 990$, and *d)* $\sigma_{\gamma 2} = 7000$, $\mu_{\gamma 2} = 2200$. Similar to the previous sub-section, we sample 300 points and plot scatter plots for cost vs. each individual tuning parameter. The scatter plots are shown in Figure D.2 and the corresponding linear fit is given as follows:

$$Cost = 7\times10^6\alpha - 4\times10^7 \qquad \text{(D.7)}$$

$$Cost = -5\times10^6\beta + 3\times10^7 \qquad \text{(D.8)}$$

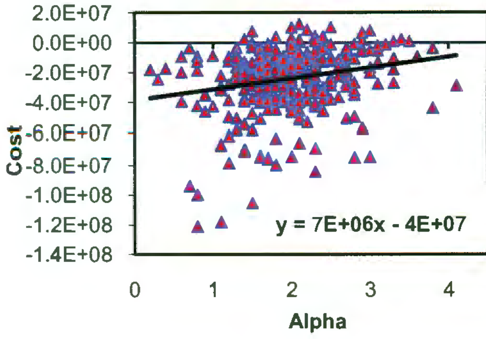
$$Cost = 2\times10^3\,\gamma_1 - 3\times10^7 \qquad \text{(D.9)}$$

$$Cost = 1.4\times10^3\,\gamma_2 - 3\times10^7 \qquad \text{(D.10)}$$

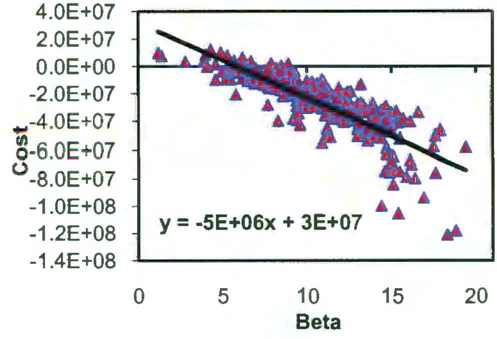
Performing the partial differentiation of cost with respect to each individual tuning parameter gives:

$$\frac{\partial(Cost)}{\partial\alpha} = 7\times10^6 \qquad \text{(D.11)}$$

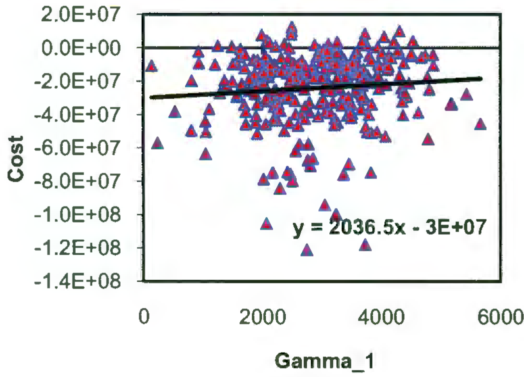
$$\frac{\partial(Cost)}{\partial\beta} = -5\times10^6 \qquad \text{(D.11)}$$



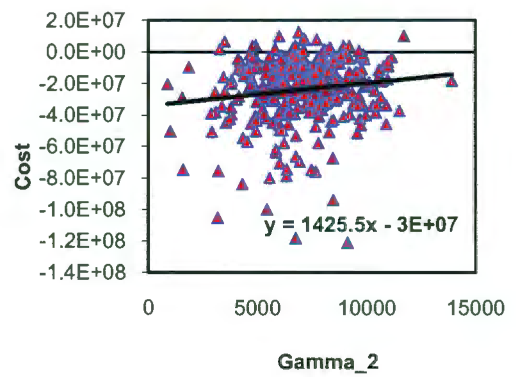
(a)



(b)



(c)



(d)

Figure D.2: Scatter plots for sensitivity analysis of the tuning parameters of the cost function for 3-FVC with module splitting.

$$\frac{\partial(Cost)}{\partial \gamma_1} = 2\times10^3 \tag{D.12}$$

$$\frac{\partial(Cost)}{\partial \gamma_2} = 1.4\times10^3 \tag{D.13}$$

From the partial differentiation, we can observe that cost is most sensitive to α and β. Furthermore, the cost increases with increasing value of α whereas the cost decreases with

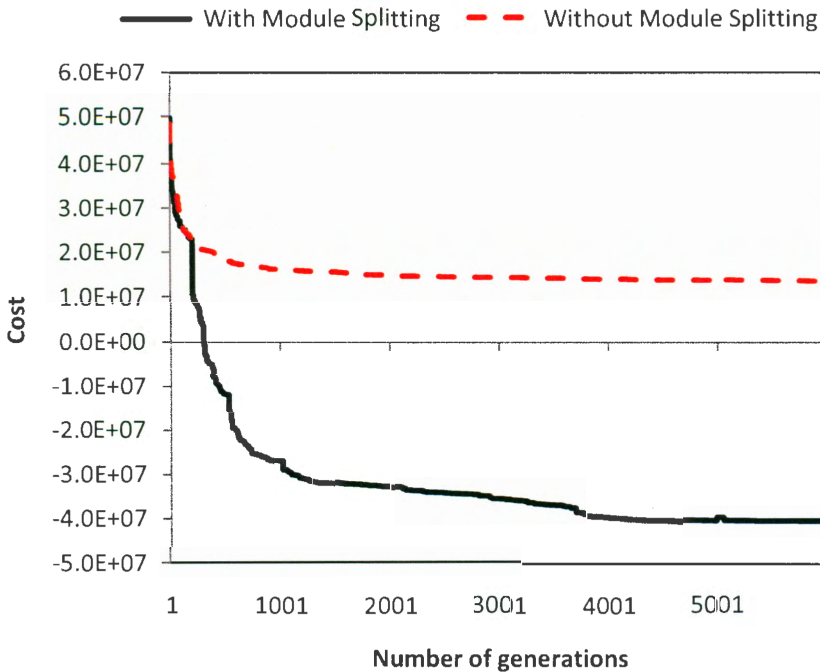


Figure D.3: Fitness Comparison of 3-D FVC with, and without module splitting.

increasing value of β. From the scatter plot it can be observed that the value of α is most suitable in the range of 0.5 to 1.5 for minimum cost. Similarly, the values of β can be chosen between 10 to 20. Furthermore, the cost is not very sensitive to γ_1 and γ_2 compared to α and β which can be observed from the near horizontal lines of the best fit linear regression in the scatter plots. Thus γ_1 can be chosen in the range of 1000 to 5000, and γ_2 can be chosen in the range of 1000 to 12000.

Figure D.3 shows the cost vs. number of generation plot for 3-D FVC *with* module splitting and 3-D FVC *without* module splitting. It can be observed that in both cases, the fitness saturates very fast within 1000 generations.

D.3 SENSITIVY ANALYSIS OF THE COST FUNCTION OF 3-D FMA

Similar to the previous section, we will perform the sensitivity analysis of the cost function of 3-D FMA. Please note that when the placement constraints are disabled, the cost function is the same as section D.1. Therefore in this section we will perform the sensitivity analysis when the placement constraints are activated in 3-D FMA and the cost is defined by eqn (5.4).

Similar to the previous two sections, we randomly generate 300 samples for each of the four tuning parameters. The σ and μ values for each tuning parameter are denoted by using the parameter's name as a sub-script for σ and μ, and their values are chosen as follows *a)* $\sigma_\alpha = 2.0$, $\mu_\alpha = 0.32$, *b)* $\sigma_\beta = 10.0$, $\mu_\beta = 3.2$, *c)* $\sigma_\gamma = 3000$, $\mu_\gamma = 990$, and d) *c)* $\sigma_\chi = 3.0$, $\mu_\chi = 0.65$. The scatter plots are shown in Figure D.4. From the linear regression of the scatter plot, we obtain the following relations:

$$Cost = 5\times10^6\alpha + 2\times10^7 \quad \text{(D.14)}$$

$$Cost = 8.1\times10^5\beta + 2\times10^7 \quad \text{(D.15)}$$

$$Cost = 4.7\times10^2\gamma + 3\times10^7 \quad \text{(D.16)}$$

$$Cost = 4\times10^6\chi + 2\times10^7 \quad \text{(D.17)}$$

Performing the differentiation of eqn. (D.14) to (D.17) gives the slope or the sensitivity of the cost function:

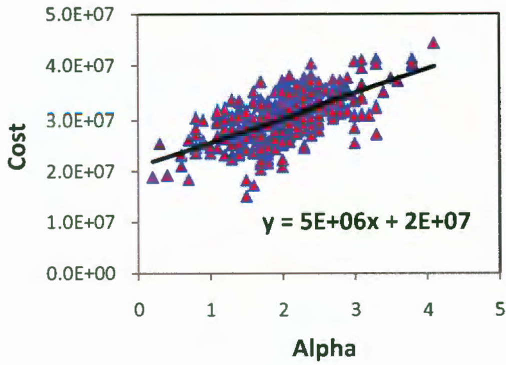
$$\frac{\partial(Cost)}{\partial\alpha} = 5\times10^{6} \quad (D.18)$$

$$\frac{\partial(Cost)}{\partial\beta} = 8.1\times10^{5} \quad (D.19)$$

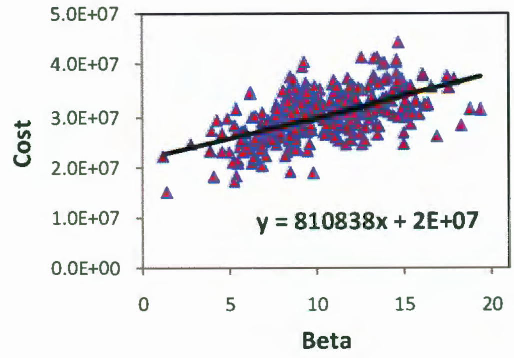
$$\frac{\partial(Cost)}{\partial\gamma} = 4.74\times10^{2} \quad (D.20)$$

$$\frac{\partial(Cost)}{\partial\chi} = 4\times10^{6} \quad (D.21)$$

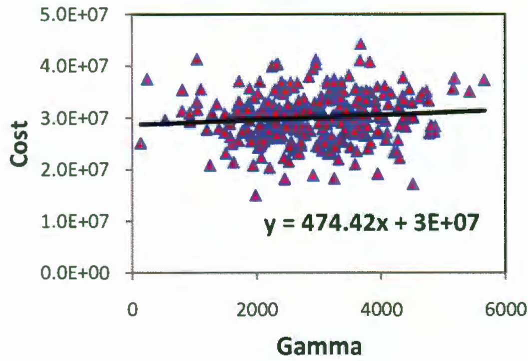
From the above slopes, it can be observed that cost is most sensitive to α and χ and least sensitive to γ. From the scatter plots of Figure D.4, α can be chosen between 0.5 – 2.0, β between 1 to 10, and χ can be chosen between 1.0 to 2.0 to achieve the optimal cost. Since the cost is least sensitive to γ (as it can be seen from the horizontal line in Figure D.4(c)), it can be chosen between 1000 – 5000.



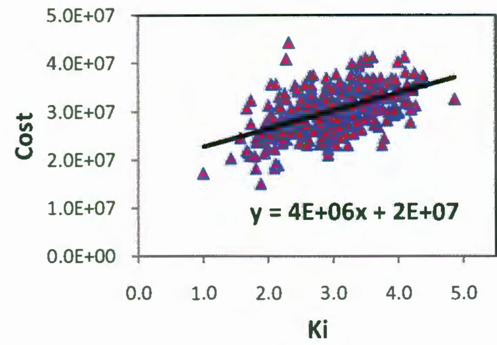
(a)



(b)



(c)



(d)

Figure D.4: Scatter plots for the sensitivity analysis of tuning parameters of the cost function for 3-D FMA algorithm.