UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RENATO FERNANDES HENTSCHKE

Algorithms for Wire Length Improvement of VLSI Circuits With Concern to Critical Paths

Thesis presented in partial fulfillment
of the requirements for the degree of
Doctor of Computer Science

Prof. Dr. Ricardo Reis
Advisor

Prof. Dr. Marcelo Johann
Coadvisor

Porto Alegre, June 2007

CATALOGAÇÃO NA PUBLICAÇÃO

Hentschke, Renato Fernandes

Algorithms for Wire Length Improvement of VLSI Circuits With Concern to Critical Paths / Renato Fernandes Hentschke. – Porto Alegre: PPGC da UFRGS, 2007.

175 f.: il.

Thesis (Ph.D.) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2007. Advisor: Ricardo Reis; Coadvisor: Marcelo Johann.

1. Placement. 2. Routing. 3. 3D Circuits. 4. Critical Paths. 5. Physical Design. 6. Algorithms. 7. CAD. 8. Microelectronics. I. Reis, Ricardo. II. Johann, Marcelo. III. Título.

"1D is not enough; 2D is good; 3D is even better!"

—

ACKNOWLEDGMENTS

Initially and most importantly, I'd like to thank my family which I am very proud of, starting with my parents **Suzana** and **Carlos**. They both provided me with an excellent formation at home and also at school. Suzana, as the best mom in the world, always gave me the all the care and affection I need to work and to live. Carlos gave me a lot of insight and advices on how to live my professional life and don't miss good opportunities. My wife (recently married) **Carolina**, with all her love and will to help, have been indirectly involved in the development of the thesis reading texts, watching presentation practices and all the support she could provide. All my family members participated on my formation and particularly here I would like to thank them for their trustfulness on my ability to work professionally. I'd like to include the names of my sister **Cristina**, my "afilhado" **Arthur** and other important family names: **Machado**, **Marcia**, **Gabriel**, **Elizabeth**, **Paulo**, **Paula**, **Eduardo**, **Claudia**, **Joaquim**, **Eunice**, **Carolina**, **Evany**, **Olga**, **Arlindo**.

Secondly, I'd like to thank people who have been involved on my academic formation, starting with my advisors **Ricardo Reis** and **Marcelo Johann**. Ricardo has been my advisor since 1998 (9 years now). During this time, I had the opportunity to learn not just technical contents but many important teachings for my life. Ricardo is always very helpful to offer me opportunities abroad which led me to travel a lot. Marcelo works with me since 1998 as well. Since the beginning, Marcelo is an example of technically successful researcher. I should mention that a big chunk of my PhD work started with many of his ideas and our work together.

A special acknowledge to the students **Guilherme Flach** and **Felipe Pinto** which worked very hard to code the global placer and other smaller but important tasks. Their very long work schedule was driven by their strong will to work and grow as researchers and developers. Their contribution to this work is very significant. My bet is that both have a very promising future.

Other people with important participation on the work: **Sandro Sawicki**, my personal friend and neighbor (same building) that worked with me on the I/O pins section; we had very important conversations on technical and non-technical issues, specially during barbecues at the top of our building. **Fabio Cecin**, my friend since we developed games at Amok, contributed significantly with my computer science formation; I could learn a lot from him during our shared rides to the University with the cost of some tokens. **Eduardo D'Avila**, my friend since under-graduation course and Crazy Birds partner playing guitar, vocals and sax, contributed with Simulated Annealing equations, specially at the time of my masters. Eduardo and Fabio are reference top programmers for me. **Gustavo Wilke**, my friend and grad student from the same advisor, besides being a good friend and partner for billiards and other sports, contributed with my paper writing and research

skills with important advices and reviews. **Lucas Brusamarello**, my friend and also Crazy Birds partner (Bass Guitar and Keyboards), was my first student as co-advisor for his under-grad final project; we exchanged many ideas on Simulated Annealing, latex and unix. **Gustavo Neuberger**, my friend and financial advisor, participated with me on CADAthlon 3 times; his technical skills contributed a lot to our 2nd place on 2005. **Reginaldo Tavares**, my friend and UERGS partner, that worked with me for some time on logic and physical synthesis. **Glauco Valim dos Santos**, my routing trees partner, provided valuable contributions to the routing work of this thesis.

It is also very worth mentioning other friends: **Fernando Paixao Cortes**, **Felipe Marques**, **Cristiano Lazzari**, **Alessandro Girardi**, **Antonio Carlos Beck Filho** (Caco), **Julio Mattos**, **Adriel Ziesemer Jr.**, **Fernanda Lima Kasternsmith**, **Luigi Carro**, **Lisane Brisolara**[2], **Cristina Meinhart**, **Marcelo Lubaszewski**. They all participated on my technical and academic formation being my co-workers at projects I participated.

One important experience for me was the IBM internship from 2004 to 2005. At this time I met many friends and technical references worth mentioning. **Reinaldo Bergamaschi** introduced me to my manager there, besides being very supportive in technical and speciality non-technical issues in the US. **Jaganathan Narasimham**, my mentor at IBM with all his experience, that worked with me in the AMAZE algorithm and other projects there, teaching me a lot about EDA. I was very fortunate to have the opportunity to work with him. **David Kung** that in fact offered me the internship and **Ruchir Puri** that were my managers, providing excellent technical feedback. **Hua Xiang** that worked with me on a 3D Circuits project.

Other people with contributions to the thesis: **Renan Fonseca** (Elmore delay), **Diogo Fiorentin** (Partitioning Algorithm), **Robert Patti** (information of Tezzaron technology), **Rhett Davis** (help on 3D technologies), **Charles Alpert** and **Steve Quay** (support of AHHK algorithm), **John Lillis** and **Milos Hrkic** (P-Trees support), **Sachin Sapatnekar** (advices on 3D circuits).

TABLE OF CONTENTS

LIST OF ABBREVIATIONS AND ACRONYMS

2D	Bi-dimensional
3D	Tri-dimensional
CAD	Computer-aided Design
CPU	Central Processing Unit
CRS	Compress Row Storage
DFS	Depth First Search
DRAM	Dynamic Random Access Memory
DP	Detailed Placement
E/S	Entrada e saída
GP	Global Placement
I/O	Input and Output
IC	Integrated Circuit
ILP	Integer Linear Programming
LD	Logic Distance
MIT	Massachusetts Institute of Technology
MITLL	Massachusetts Institute of Technology Lincoln Laboratory
MMC	Multiple Markov Chains
MRST	Minimum Rectilinear Steiner Tree REVIEW THIS ONE
PLF	Path Length Factor
RLM	Random Logic Macro (Random Logic Block)
SF	Sharing Factor
SOI	Sillicon on Insulator
SoC	System-on-a-chip
SRAM	Static Random Access Memory
SA	Simulated Annealing
TA	Threshold Accept

TSV	Tezzaron Super Via
VLSI	Very Large Scale Integration

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

This thesis targets the wire length improvement of VLSI circuits considering critical elements of a circuit. It considers the problem from two different perspectives: placement and routing.

On placement, it explores methods to perform placement of 3D circuits considering issues related to vertical interconnects (3D-Vias). A complete flow, starting from the I/O pins handling, global placement, detailed placement and 3D-Via placement is presented. The I/O pins algorithm spreads the I/Os evenly and aids the placer to obtain a reduced number of 3D-Vias. The global placement engine based on Quadratic algorithm considers the technology information and 3D-Via pitch to reduce wire length and balance the cells distribution on 3D. Critical connections can be handled by insertion of artificial nets that lead to 3D-Via avoidance for those nets. Finally, 3D-Vias are placed by a fast algorithm based on Tetris legalization. The whole framework enforces the potential benefits of 3D-Circuits on wire length improvement and demonstrates efficient algorithms designed for 3D placement that can be incorporated in new tools.

On routing, a new flexible Steiner tree algorithm called AMAZE is proposed, combining existing and new methods that are very effective to produce short wire length and low delay to critical elements. A biasing technique provides close to optimal wire lengths while a path length factor and a sharing factor enables a very wide delay and wire length trade-off. While AMAZE presents significant improvements on a industry standard routing algorithm (Maze Routers), it produces routing trees with comparable speed and beter delay than heuristic Steiner tree algorithms such as AHHK and P-Trees.

Keywords: Placement, Routing, 3D Circuits, Critical Paths, Physical Design, Algorithms, CAD, Microelectronics.

Algorítmos para redução do comprimento dos fios de circuitos VLSI considerando caminhos críticos

RESUMO

Esta tese objetiva propor algorítmos para a redução do tamanho dos fios em circuitos VLSI considerando elementos críticos dos circuitos. O problema é abordado em duas perspectivas diferentes: posicionamento e roteamento.

Na abordagem de posicionamento, a tese explora métodos para realizar posicionamento de um tipo particular de circuito VLSI, que são conhecidos como circuitos 3D. Diferente de trabalhos anteriores, este tese aborda o problema considerando as conexões verticais (chamadas 3D-Vias) e as limitações impostas pelas mesmas. Foi realizado um fluxo completo, iniciando no tratamento de pinos de entrada e saída (E/S), posicionamento global, posicionamento detalhado e posicionamento das 3D-Vias. A primeira etapa espalha os pinos de E/S de maneira equilibrada objetivando auxiliar o posicionamento para obter uma quantidade reduzida de 3D-Vias. O mecanismo de posicionamento global baseado no algorítmo de *Quadratic Placement* considera informações da tecnologia e requerimento de espaçamento de 3D-Vias para reduzir o comprimento das conexões e equilibrar a distrubuição das células em 3D. Conexes críticas podem ser tratadas através da insercão de redes artificiais que auxiliam a evitar que 3D-Vias sejam usadas em conexões críticas do circuito. Finalmente, 3D-Vias são posicionadas por um algorítmo rápido baseado na legalizaçãao Tetris. O *framework* completo reforça os potenciais benefícios dos circuitos 3D para a melhora do comprimento das conexões e apresenta algorítmos eficientes projetados para circutos 3D podendo estes serem incorporados em novas ferramentas.

Na abordagem de roteamento, um novo algorítmo para obtenção de árvores de Steiner chamado AMAZE é proposto, combinando métodos existentes com novos métodos que são efetivos para produzir fios curtos e de baixo atraso para elementos críticos. Um técnica de *biasing* atua na redução do tamanho dos fios, obtendo resultados próximos da solução ótima enquanto que dois fatores de *timing* chamados *path-length factor* e *sharing factor* propiciam melhora do atraso para conexões sabidas como críticas. Enquanto que AMAZE apresenta melhorias significativas em um algorítmo padrão na indústria de CAD (Maze Routers), ele produz árvores de roteamento com uso de CPU comparável com algorítmos heurísticos de árvore de Steiner e menor atraso.

Palavras-chave: posicionamento, roteamento, circuitos 3d, síntese fisica.

1 INTRODUCTION

Microelectronics design automation is a very vast field of science. For half a century, people have been researching new techniques to automate some parts of the design tasks. Such tasks were related to massive manual work, such as placement and routing (physical design) of standard cells. As the years passed, new design steps were automated and today the whole design flow has associated CAD tools. At the same time, new challenges to CAD research appeared, to cope with new design and technology problems. The technology for IC fabrication shrunk dimensions systematically offering more resources for designers. Moore's Law (MOORES LAW: MADE REAL BY INTEL INNOVATION, 2005) says that every year doubles the transistor count. This complexity growth demands better CAD algorithms for a successful automation of the design. At the same time, the dimension of the components in a microchip is being measured in nanometers, leading to many manufacturing related new challenges to be dealt by both designers and CAD engineers. Nowadays, technology drives the advances on CAD as much as the tight design requirements.

One of the most important issues imposed by recent technologies are related to circuit wires. First, consider the design size growth while the component sizes are becoming dramatically smaller. This scenario imposes larger, denser and more complex wiring networks. Secondly, consider that the delay of active elements decreased faster than the interconnect delay. Today, interconnect resistance is extremely relevant, while it was ignored in the past. For these reasons, interconnect delay is responsible for more than 50% of a circuit delay. Thirdly, consider the manufacturing and parasitics problems on recent technologies, which basically imposes strict design rules for a more regular layout. Interconnect regularity is a more complex issue due to the randomized topology of a circuit netlist. Finally, consider circuit power, which is strongly affected by the capacitance of circuit nodes. The large wires create considerably large capacitance amounts to be charged or discharged. In conclusion, fast design cycle, routability[1], timing, power and manufacturability are strongly affected by interconnect complexity of a design.

In order to cope with wire related problems, the effort of reducing wire length is a very relevant issue on CAD research. Shorter wires are faster, dissipate less power, lead to less complex wiring networks affecting routability and manufacturability. As a very relevant topic, wire length reduction has been largely addressed by industry and academia research. Among the proposed techniques, new algorithms for synthesizing circuits are one of the more effective means.

The many stages on CAD design flow can be grouped in four consecutive steps: sys-

[1]Routability is defined by the ability of routing algorithms to route all wires under electrical and topological restrictions

tem level synthesis, high level synthesis, logic synthesis and physical synthesis. System level tasks are related to modeling of the whole system, partitioning in hardware and software and finally synthesizing the code to a lower level according to the nature of the application. High level synthesis is related to the hardware synthesis of a behavioral description. Logic synthesis is responsible for the boolean optimization and mapping to an adequate gate set that can be implemented in the physical level. Physical synthesis is responsible for the translation of a netlist into a viable silicon implementation that respects all electrical issues to guarantee that the circuit will run properly considering the design specifications.

Physical synthesis step is a very complex step since it is strongly affected by the nanometer design constraints. It is composed by subtasks as follows. In a cell based methodology, the gates should be efficiently sized and the nets buffered in order to balance capacitive loads and driving strength to meet timing constraints. These gates must be properly placed in the die, which lies into the *placement* problem. As these gates should be connected by wires, placement stage determines the sizes of all the wires and for this reason it is a very critical stage in the CAD flow. After placement, the *routing* problem connects the circuit elements with metal wires. The reader can refer to (SHERWANI, 1998) for more basic information of Physical Design subtasks.

Physical design is directly responsible for handling the interconnect related issues and must work on wire length optimization. Nowadays, there is a huge amount of work on wire length driven placement and routing algorithms. However, with the constant growing design complexity together with constant technology shrinking, there is a constant demand for new solutions. Particularly, timing and power optimization are more recent problems that impact significantly placement and routing, demanding constant update of algorithms. Timing and power analysis are able to identify critical components of the circuit that could receive special concern and consequently shorter wire length than the circuit average. The timing-driven and power-driven placement and routing algorithms, largely studied in the literature, are examples of such methodologies.

Technology shirking used to be one important tool to improve delay and power of VLSI circuits. Today, wire related problems tend to increase dramatically at each new technology node. Note that the size of elements in a VLSI circuit are close to atomic dimensions. This scenario suggests the research for alternative solutions.

Recently, the 3D circuit technologies were proposed. The 3D circuits, that are starting to be a reality in the industry, arrive as a possible improvement to the interconnect paradigm. It is expectable that arranging the chip components in a 3D space can reduce the interconnect length between the components. In fact, It has been demonstrated by some recent work ((DAS, S. et al, 2004), (ABABEI; MOGAL; BAZARGAN, 2005), (DAVIS, W. et al, 2005), (BANERJEE, K. et al, 2001) and many others that 3D circuits can potentially reduce wire lengths. It is also shown that the wire length improvement is proportional to the circuit size (OBENAUS; SZYMANSKI, 1999). However, timing improvement is still an open issue. It is reasonable to consider that a 3D implementation of a design can improve timing compared to a 2D implementation. To achieve that, new algorithms and theoretical models must be developed.

There is massive interest from both industry and academia in 3D integration issues. Major companies like IBM, Intel, Samsung, Micron, Cadence and Infineon are some examples. There are also some very interesting work coming from startups, such as Ziptronix, Xanoptix and Tezzaron. From Academia, there is strong initiative from MIT, Cornell University, University of Minessotta, Stanford University, IMEC, Purdue University

and Tohuku University. There exists research on 3D integration from technology perspective (such as copper bonding, SOI stacked layers, diffusion soldering, and others), design perspective (stacked memories, commercial solutions for stacked processor and memory) and CAD perspective (new algorithms and tools). More details can be found at Tezzaron home-page (3D ICS INDUSTRY SUMMARY, 2005) (TEZZARON HOMEPAGE, 2005). They provide a wide range picture of companies, universities and initiatives related to 3D ICs.

The 3D circuits open a huge research space for CAD algorithms, specially in the physical design level. New algorithms must be provided in order to handle 3D arrangement of circuit elements and to take full benefit of technology while considering new issues caused by the 3D integration. Today, the research in this field is still on the early years. Consider the 3D placement problem, for instance. The richness and variety of techniques that led to significant improvement on 2D circuit placement over the decades will be able to deliver similar maturity on the 3D placement field.

This thesis targets the wire length improvement of VLSI circuit considering critical elements of a circuit. It considers the problem from two different perspectives: placement and routing. On placement, it explores methods to perform placement of 3D circuits considering issues related to vertical interconnects (3D-Vias - see section 2.3 while searching for wire length improvement. A very significant volume of new methods and contributions are presented in the text and fully validated in a tool called Z-Place. On routing, a new flexible algorithm called AMAZE is proposed, combining existing and new methods that are very effective to produce short wire length and low delay to critical elements. While AMAZE presents significant improvements on a industry standard routing algorithm (Maze Routers), it produces routing trees with enough flexibility that enables it to be used on several applications such as wire length estimation (Steiner tree, timing driven Steiner tree), buffering, global routing. It could potentially be applied to 3D routing as well.

The text is organized as follows. Chapter 2 summarizes the contributions of the thesis for a better and objective reading of the text. Chapter A.2 presents the 3D circuits and discusses how to take advantage of the technology to produce short wire lengths considering existing research. Chapter A.3 presents Z-Place algorithms, containing the contributions of this thesis to the 3D placement community. Chapter 5 reviews existing routing algorithms relevant to the scope of this thesis on flexible algorithms as well as presents AMAZE algorithm and properties, containing the contributions of this thesis to the routing community. Finally, chapter A.5 presents a summary of the conclusions of the PhD research that appear throughout the text on “partial conclusion" sections.

2 SUMMARY OF CONTRIBUTIONS

For a better reading of this text, this chapter presents a summary of the new contributions presented all over the text. This thesis presents algorithms for wire length improvement on VLSI circuits with concern to critical paths. The algorithms basically handle two distinct problems: cell placement and routing.

With the advent of 3D circuits and the demand for placement solutions that are able to effectively take advantage of its potential, the cell placement algorithms on this thesis were designed to handle 3D placement. The algorithms represent a complete flow for getting a block of cells and placing them fully legally into various tiers of active area, minimizing wire length. In summary, the placement flow consists of: a method to handle I/O pins, a method to place cells globally while planning 3D-Vias at the same time, detailed placement algorithms, a method to place 3D-Vias and finally a method to handle critical connections into 3D avoiding them to take expensive 3D-Vias. At each step, there are algorithms and partial conclusions presented in the text.

In the routing side, a fast and flexible algorithm is presented. It combines existing and new methods that are very effective to produce short wire length and low delay to critical elements.

2.1 3D-Placement

2.1.1 Z-Place flow

Z-Place performs the tasks of: I/Os handling (section A.3.1.2), global placement (section A.3.1.3), detailed placement, 3D-Vias placement (section A.3.1.5). The detailed placement is basically a windowed Threshold Accept (DUECK; SCHEUER, 1990) improvement within each tier. The planning of inter-tier connections (3D-Vias) is performed at the global placement as well as the handling of critical wires (section A.3.1.4).

2.1.2 I/O Pins Handling

An algorithm for I/O pins partitioning and placement targeting 3D circuits is proposed (HENTSCHKE; JOHANN; REIS, 2006). The method starts from a standard 2D placement of the pins around a flat rectangle and outputs a 3D representation of the circuit composed of a set of tiers and pins placed at the four sides of the resulting cube. The proposed algorithm targets a balanced distribution of the I/Os that is required both for accommodating the pins evenly as well as to serve as an starting point for cell placement algorithms that are initially guided by I/O's locations, such as analytical placers. Moreover, the I/O partitioning tries to set pins in such a way that it favors the cell placer to reach a reduced number of 3D-Vias. The method works in two tasks: first partition the

I/Os considering the logic distances as weights; second, fix the I/Os and perform partitioning of the cells. The experimental results show the effectiveness of the approach on balance and number of 3D-Vias compared to simplistic methods for I/O partitioning, including traditional min-cut algorithms. Since our method contains the information of the whole circuit compressed in a small graph, it could actually improve the partitioning algorithm at the expense of more CPU time.

2.1.3 Global Placement

Under a horizontally sliced cube model, the task of the global placement is to propose a placement of all cells within one of the available tiers; a second task is to perform a reasonable 3D-Via planning considering the characteristics of the target technology.

The main placement engine is analytic targeting at Quadratic wire length minimization. Our placer has similar features to the one from (VISWANATHAN; PAN; CHU, 2005) for 2D circuits. We apply 3D Cell Shifting in order to spread the cells that are initially placed in invalid and overlapped coordinates concentrated at the center of the cube. The 3D cell shifting enforces a true 3D behavior of Z-Place.

Global placement has constraints related to 3D-Vias feasibility and area balance. The Z-Cell Shifting methodology (HENTSCHKE, R. et al, 2006) sorts the cells and obtains threshold points considering all 3D-Vias in order to assign groups to tiers. Finally, an iterative refinement stage provides a greedy heuristic that optimizes linear 3D wire length while controlling the 3D-Vias to be within the upper bound (HENTSCHKE, R. et al, 2007a).

Experimental results demonstrated that by manipulating this upper bound we can play with a trade-off between number of 3D-Vias and 3D wire length. The trade-off is triggered by the 3D-Via's area; Z-Place adapts itself to the technology, improving the wire length only if there is available space for that. We studied the trend of adding more tiers against 2D solutions provided by Fastplace tool (VISWANATHAN; PAN; CHU, 2005) and verified that our best efforts (with small pitch 3D-Vias) produces improvements of 15% for 2 tiers, 20% for 3 tiers and finally 27% for 4 tiers in average.

2.1.4 Handling Critical Paths

We exploit the problem of keeping critical paths with no 3D-Via connections. There are several issues on 3D-Vias that would make them not attractive to critical wires, such as large capacitance and resistance related both to the 3D-Via itself as well as the wires and vias needed to connect to it, crossing all metal layers.

In order to keep the cells on the same critical path together, an artificial pin connecting all cells in a critical path is inserted in the netlist. The connection has a high weight for the Z axis in contrast with a low (or zero) weight for X or Y. Experimental results on a benchmark set of 14 circuits placed into 2, 3 and 4 tiers demonstrated the effectiveness of the approach by reducing the number of critical 3D-Vias from several hundreds to zero in all benchmarks. The total wire length and number of 3D-Vias were only slightly affected (0.03% and 2% in average).

2.1.5 3D-Via Placement

3D-Vias are assigned to layers between the circuit tiers (HENTSCHKE; REIS, 2007). The problem of 3D-Vias placement consists of placing the 3D-Vias with no overlap with other 3D-Vias in the same layer. Since cell placement is fixed, positions inside the net

bounding box are preferred and wire length minimization is used as target function. We present a heuristic based on the Tetris legalization approach (KHATKHATE, A. et al, 2004) for the 3D-Via legalization. Our experimental results show that the algorithm can accommodate the 3D-Vias in such a way that wire length overhead is close to zero in easy instances and still very low for harder instances (in most of the cases it is less than 0.1% and it is less than 5% in all cases). Compared to an existing approach (YAN, H. et al, 2005), it obtains slightly better results with an advantage of orders of magnitude on run time.

2.2 Steiner Routing with AMAZE

AMAZE (HENTSCHKE, R. et al, 2007b) addresses the problem of generating good topologies of rectilinear Steiner trees using path search algorithms. On run time, clever usage of heuristic search methods and a constant time insertion data structure to store the elements to be routed leads to a very fast algorithm (415 runs on 7 pin nets consumed 0.94s).

A biasing technique that favors wire sharing with future path searches proposed for wire length improvement produces trees that are within 2% from optimal topologies in average. By introducing a sharing factor (marking some wires prohibited for sharing) and a path-length factor (reducing driver-to-sink distance) we show how to trade-off wire length for delay. Our experimental results show that AMAZE is more effective to optimize delay to critical sinks than state-of-the-art heuristics for Steiner trees, such as AHHK (from 26% to 40%) and P-Trees (from 1% to 30%) while keeping the properties of a routing algorithm. We also analyzed the ability of AMAZE to handle blockages and verified experimentally that AMAZE produces tree with better delay to the critical sinks than P-Trees from 6% (5 pin nets) to 21% (9 pin nets).

While AMAZE presents significant improvements on a industry standard routing algorithm (Maze Routers), it produces routing trees with enough flexibility that enables it to be used on several applications such as wire length estimation (Steiner tree, timing driven Steiner tree), buffering, global routing.

2.3 Dictionary of terms

Some terms used throughout the text are not accurately used in the existing literature since they might have different meanings. In order to precisely define the terms, table 2.1 presents basic definitions as they are used in this thesis.

Table 2.1: Dictionary of terms

3D circuits	A VLSI Circuit with multiple layers of active area such that they are stacked in the vertical dimension.
3D-Vias	Any connection between a pair of adjacent tiers
Bulk	Every VLSI chip is composed by a very thick silicon block on the back side; the active area is built in a thin slice of the silicon, composing the top of the chip. Metal layers are deposited on the top of the active area. The Bulk is exactly the silicon block that sustains the chip.
Critical wires	Some wires in the circuit must receive an special concern of the tool because they are part of a timing (or power) critical region of the circuit. Usually they receive an extra weigh or special algorithm such that their length will be smaller than non-critical wires.
I/O Pin	I/O Pin designate the I/O interface of a block in the circuit. The I/O pins are placed on the boundary of the block and serves the purpose of communicating the block with other blocks. The layout of an I/O pin would be simply a small metal peace.
I/O Pad	An I/O Pad is the interface of a chip with the external world. An I/O Pad can be placed anywhere in the chip, but is usually placed in the boundaries. The layout of an I/O Pad is a big metal block (big enough to connect to external wires) with some circuitry (buffers) to interface the outside world currents with inner circuit currents
Integration Strategy	The style of integration between a pair of adjacent tiers. It can be either face-to-face, face-to-back or back-to-back. A tier has an active layer and metal layers on the top. The f̈aceöf a tier is the metal layers, while the active area is the b̈ack.
SOI	SOI is a manufacturing technology that separates the silicon substrate into two blocks with a buried insulator layer. The bottom silicon that does not have any active functionality; it just serves the purpose of sustaining the chip structure (otherwise it could break). The silicon in the top contains the transistors and other active elements. It is very thin compared to the bottom silicon.
Tier	An active layer of the 3D VLSI circuit
Through Vias	An special type of 3D-Via that is used only for face-to-back or back-to-back integrations. The Through Vias dig a hole in the active area and go though it to connect to a tier (or a pad) on the back side.
Via	Via is used in the text to designate the regular via that connects a metal layer with an adjacent metal layer.

3 3D CIRCUITS AS A NOVEL DESIGN PARADIGM

A 3D circuit can be understood as a VLSI chip with stacked active layers called **tiers**. Figure A.1 provides a didactic view of a 3D Chip with intercalated active layers and metal layers. This organization is subject to the integration strategy (see section 3.2). More details on the technologies, how they are manufactured and impacts on design methodologies will be presented are provided in the following sections.

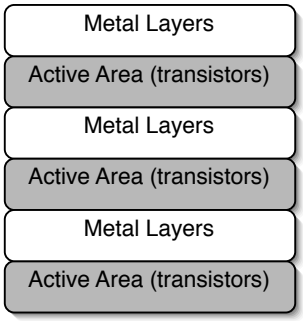


Figure 3.1: A didactic picture of 3D circuit, tiers of active area and metal layers

3.1 3D Assembly

This section is written based on recent publications from (GUPTA, S. et al, 2005) and (PATTI, 2006a). More details can be retrieved from the referred papers.

According to (PATTI, 2006a), the assembly of 3D Chips is performed in different integration granularities. This methods are summarized in table 3.1, from the coarser grain to the finest possible grain.

Chip stacking is simply the vertical stacking of fully pre-manufactured chips. The chips have regular buffered I/O connections integrated usually by wire bonding (DAVIS, W. et al, 2005). Since all inter-chip communication must pass through the I/O buffers

Table 3.1: Manufacturing methods for 3D-ICs sorted by granularity from coarser to finest.

Chip Stacking
Die-on-Wafer Stacking
Wafer-Level Stacking
Transistor Stacking

Table 3.2: Summary of 3D ICs technologies for integration and communication of several tiers. Extracted from (DAVIS, W. et al, 2005)

Integration Technology	Communication Technology	Figure Index
Wire Bonded	Wires outside the chip	figure 3.2.(a)
Microbump	Routing on 3D package	figure 3.2.(b)
	Face-to-face connection	figure 3.2.(c)
Contactless	Capacitive connection	figure 3.2.(d)
	Inductive connection	figure 3.2.(e)
Through Via	Bulk	figure 3.2.(f)
	SOI	figure 3.2.(g)

going outside of the chip, this methodology does not provide any advantage to circuit performance and power, reducing only the area occupied by the chip on the board. This technique is applied for cell-phones and other portable devices.

Die-on-wafer stacking is performed by stacking individual tested dies into a host wafer. Positions of the host wafer can be also pre-tested. The individual dies are placed using a pick-and-place equipment, that is a bottleneck for the cost, quality and size of inter-chip communication. Patti (PATTI, 2006a) reports that the placement misalignment today is about 10 μm. Depending on the type of 3D-Via used to communicate the dies (via types are summarized in section A.2.1), this method can provide higher integration density compared to chip stacking. Currently, several startup companies announced their technology for 3D integration based on die-to-wafer stacking, such as ZyCube, Ziptronix and Xan3D (PATTI, 2006a).

Wafer-level stacking bonds entire wafers into a stack. Tezzaron is one company working with this kind of integration. Compared to die-on-wafer stacking, Tezzaron's technology (GUPTA, S. et al, 2005) achieves better alignment (1 μm) and a more planar surface, leading to more integrated communication.

Both die-on-wafer and wafer-level integration strategies are used commercially in the present days. Finally, the **transistor stacking** methodology is an ideal integration of active layers fabricated in the same wafer, dismissing any equipment for wafer alignment. Today, those devices cannot be fabricated mainly due to high temperature processes during the wafer manufacturing. Basically, the technology for fabricating high-performance transistors demands temperatures that would destroy any copper or aluminum used to manufacture metal layers bellow it. There is ongoing research in order to solve this issue and this technology is promising in the future.

3.2 3D-Vias Technologies and Integration Strategies

This section studies the ways of integrating two or more tiers into a single chip and how they can communicate. According to (DAVIS, W. et al, 2005) the integration and communication technology can be classified as shown in table 3.2 and illustrated by figure 3.2 (extracted from (DAVIS, W. et al, 2005)).

The **wire bonded** technology was already mentioned in the previous section at the description of *chip stacking*. Tiers of different sizes are stacked and I/O Pads are placed in the boundary of the tiers in such a way that they are not blocked by the upper tier. The main disadvantage of this technology is that wires are out of the chip scope, so they must be buffered and the pads consume very large areas.

Microbump technology provides gold micro contacts (bumps) placed in the top metal layer (sometimes the top two metal layers may be blocked for other routing). For this technology, chips can be stacked face-to-back and the package itself can provide routing space (3D package). On the other-hand, stacking the chips in a face-to-face fashion provides simpler (and consequently better) routing requiring no wiring channels in the package. The tiers are placed in such a way that their respective bumps are physically connected. Face-to-face integration is limited to two tiers.

The **contactless** technologies can be summarized as capacitive and inductive coupling. The capacity coupling technologies require the chips to be placed face-to-face because the contacts have a very tight proximity constraint. Inductive coupling is usually integrated face-to-back.

Finally, **Through Vias** consists of digging a hole through the tier for face-to-back comunication. Sometimes, such as in MITLL 3D technology, the first two tiers are integrated face-to-face while the rest of the tiers are stacked face-to-back. Even two chips connected face-to-face will need face-to-back communication with the I/O pads, as reviewed in the next section. Due to silicon polishing issues, the traditional Bulk technologies require a much larger pitch compared to SOI processes for 3D, such as in the MITLL (more details are provided in section 3.3.2). But still in the face-to-face integration, the technology for digging the hole in the oxide and depositing metal is similar.

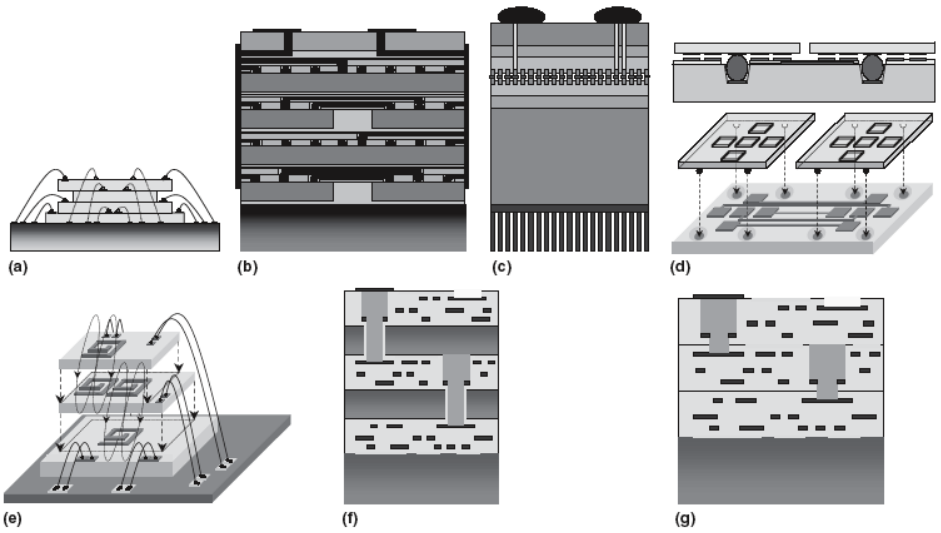


Figure 3.2: Illustration of the integration and interconnect technologies: (a) wire bonded; (b) microbump in a 3D-package; (c) microbump face-to-face; (d) contactless with buried dumps; (e) contactless with inductive technology; (f) Through Vias with Bulk; (g) Through Vias with SOI. Extracted from (DAVIS, W. et al, 2005).

3.3 Useful Technology Details of 3D-Vias

For simplicity, for now on, we call **3D-Via** every connection between a pair of adjacent tiers, applicable for any kind of technology.

A technology for 3D-Circuit fabrication is based on: a methodology for assembling the stack (chip-level, wafer-level, etc...); a polishing technique to improve integration; a

technology for aligning the chips; a technology for bonding the chips together (organic glues, metal-to-metal, etc.). Today, a variety of those technologies are found, impacting geometry, routing and electrical characteristics of 3D-Vias. For instance, 3D-Via pitch is strongly affected by the quality of the alignment of the stacked wafers. Assuming that those methodologies are currently under research, we can expect that 3D-Vias technologies will improve and provide smaller pitches in the near future.

Two technology geometry numbers have a very important impact on chip design and CAD: 3D-Via pitch and length. The pitches determine the amount of 3D-Vias that can be effectively used within a certain area. It also determines how close the vertical connections can be placed. 3D-Via length is important to compute the effective wire length. More importantly, in order to connect to a 3D-Via, a wiring network that goes through all the metal layers. The length of this network should be accounted in wire length. Note that this distance goes through a number of metal layers, depending whether the integration is face-to-face or face-to-back. In a face-to-face integration, a connection between adjacent tiers must go though twice the number of metal layers of a chip (or the sum the number of metal layers in both chips); in the face-to-back integration, only the metal layers of one chip must be traversed. The metric *3D wire length* is used to compute wire length accounting also for the *Z* axis. It can be computed simply by adding the usual Half Perimeter (SHERWANI, 1998) to the effective distance on the *Z* axis.

Routing characteristics of 3D-Vias are also very important to the design of 3D VLSI chips. Some 3D-Vias technologies demand blocking of metal layers; that can vary from only the top metal layers to the whole set. Still, even if the 3D-Via requires only the top layer, some routing must be done to connect it to a transistor.

Electrical characteristics are very important in order to compute delay and power consumed by those connections. Obviously that high capacitances and resistance could potentially invalidate the use of 3D-Vias for critical connections in the circuit; however, even in favorable scenarios, inter-tier connections still have to go through all the metal layers and regular vias, which is also costly.

3.3.1 Tezzaron Technologies

As mentioned above, Tezzaron Technologies are based on wafer level assembly. The 3D-Via technology can be either Through Vias and Microbump contacts integrated at the wafer level. The wafers are bonded in a metal-to-metal methodology. Tezzaron provides many internet links, articles and information on their website (TEZZARON HOMEPAGE, 2005).

Tezzaron connects tiers either integrated in face-to-face or face-to-back. Every tier of the 3D chip contains *Copper Pads* that are used by all inter-tier connections. For the face-to-face connection, the direct physical contact of the Copper Pads serves as the 3D-Via connecting the devices. For face-to-back connections of tiers t_1 and t_2, the Copper Pad in tier t_1 must be connected to a Through Via that goes through the bulk of t_2; the ending point of the connection on t_2 will depend on the technology being used. Tezzaron currently has two Through Via technologies: Super-Via and Super-Contact (both are trademarks of Tezzaron) (PATTI, 2006a)(GUPTA, S. et al, 2005).

The Super-Via technology builds an entire piece of metal that connects the Copper Pad of tier t_1 to the Copper Pad of tier t_2, passing though all the metal layers. For this reason it obviously blocks routing in all metal layers in that particular area. Figure 3.3 shows a picture of the actual chip provided by Tezzaron. The picture shows 3 wafers integrated in face-to-face (wafers 1 and 2) and face-to-back (wafers 2 and 3). The picture also provides

some data on the vertical distances of the chip. This data will be used later to build table A.1 in section A.2.1. Note that face-to-face communication is done by direct physical contact of *Copper Pads* similarly to the Microbump technology, while the face-to-back connections make use of Super-Vias.

Observe the following facts from figure 3.4:

- Wafer 1 is connected to wafer 2 in a face-to-face fashion. The 3D-Vias connecting them are simply direct contact of Copper Pads.
- Wafer 2 and 3 integration is face-to-back. The connection between them (see the only connection on the right side of the picture) is accomplished by a Super-Via that connects the Copper Pad of Wafer 3 to the Copper Pad of wafer 2.
- Wafer 1 is connected to I/O Pads that are placed on the back-side of its Bulk (in a back-side metal layer). This connection can also be considered face-to-back and is done with Super-Vias.
- Wafer 3 is also connection to I/O Pads with Super-Vias.

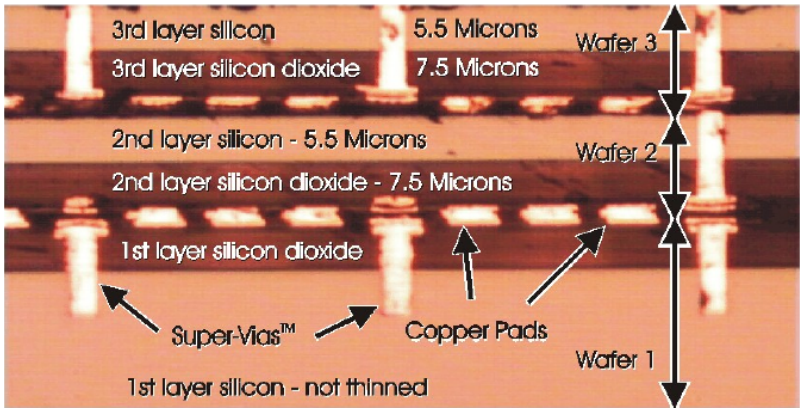


Figure 3.3: A picture of 3D Stacked circuits from Tezzaron technology using Super-Vias. Extracted from (TEZZARON HOMEPAGE, 2005).

The second generation of Tezzaron process introduced the Super-Contact technology, which does not use all metal layers above the contact immersed in the active area. Figure 3.4 shows another microscopic picture of an actual Tezzaron chip with Super-Contact. The upper right corner of the picture displays a Super-Contact. It can be clearly verified that the contact does not go through the metal layers, providing that empty space to route other nets. Note also that the face-to-face communication is identical to the one in the Super-Vias.

Observe the following facts from figure 3.3:

- There are two wafers connected face-to-face.
- There is one face-to-back connection to an I/O Pad. This connection (on the right upper side of the picture) stops at metal one, leaving routing space available on the upper metal-layers.

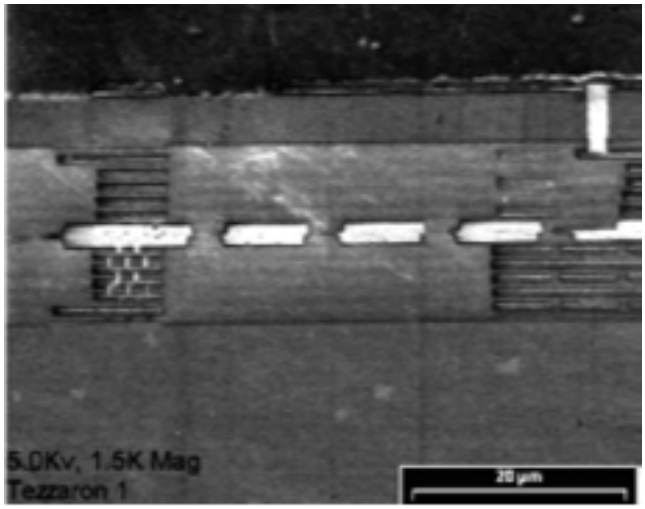


Figure 3.4: A picture of two Stacked circuits from Tezzaron technology using Super-Contacts to connect to I/O pins placed on the back-side of the wafer Bulks. Extracted from (PATTI, 2006a).

Table 3.3: Summary of 3D ICs technologies for integration and communication of several tiers. Extracted from (GUPTA, S. et al, 2005)

	Gen I: Super-Via^{TM}	Gen II: Super-Contact^{TM}	Face-to-face (Projected)
Size	4.0 μm $\times$ 4.0 μm	1.2 μm $\times$ 1.2 μm	1.7 μm $\times$ 1.7 μm (0.75 μm $\times$ 0.75μm)
Minimum Pitch	6.08 μm	< 4 μm	2.4 μm (1.46 μm)
Feed-Through Capacitance	7 fF	2-3 fF	<<
Series Resistance	<0.25 Ω	<0.35 Ω	<

Figure A.2 provides a didactic drawing of the Tezzaron integration with Super-Contacts and Copper Pads connected face-to-face. It was made based on the Tezzaron files and on Patti's paper (PATTI, 2006a).

Table 3.3 provides a summary of the integration capability and electrical characteristics of current Tezzaron technologies for face-to-back connections (Super-Via and Super-Contact) and face-to-face ones. It also demonstrates a projection to shrink the size of the face-to-face Copper Pads. The table is extracted from Tezzaron documentation (GUPTA, S. et al, 2005).

3.3.2 MITLL 3D technology

The MITLL 3D technology (SUNTHARALINGAM, V. et al, 2005) (DAS, S. et al, 2004) provides similar integration possibilities compared to Tezzaron, building a stack of tiers where the first pair is face-to-face and all others are face-to-back. It is based on SOI wafers that provides a very good pitch for the Through Vias (DAVIS, W. et al, 2005) as summarized in the next section.

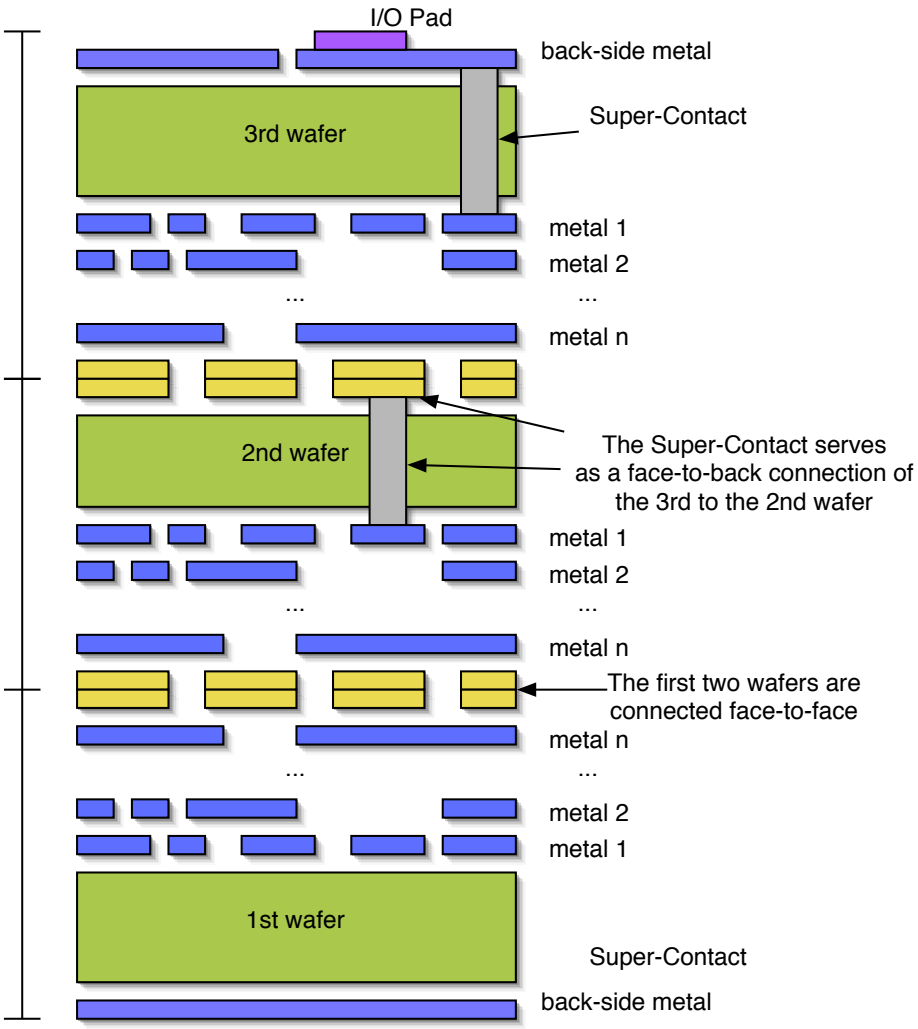


Figure 3.5: Tezzaron Technology with Super-Contacts and face-to-face connection.

Davis (DAVIS, W. et al, 2005) provides a didactic picture of the MITLL technology that was extracted to figure 3.6.

Suntharalingam et. al (SUNTHARALINGAM, V. et al, 2005) develops an image capturing chip for the MITLL 3D technology. They stacked the photodiodes assembled in one tier with the image processing modules on the second tier. The tiers are integrated face-to-face, as demonstrated in the picture extracted from their paper to figure 3.7. It seems that face-to-face connections are more expensive than Tezzaron's because they need a straight connection blocking all metal layers, while in Tezzaron face-to-face connection blocks only the top layer.

Das et. al (DAS, S. et al, 2004) describes in detail the technology used to manufacture 3D chips as well as CAD tools. Since MITLL technology is based on SOI technology, the backside silicon can be removed; this way the thickness of the active area is reduced to the thickness of active silicon (0.1 μm) plus the buried insulator (0.4 μm).

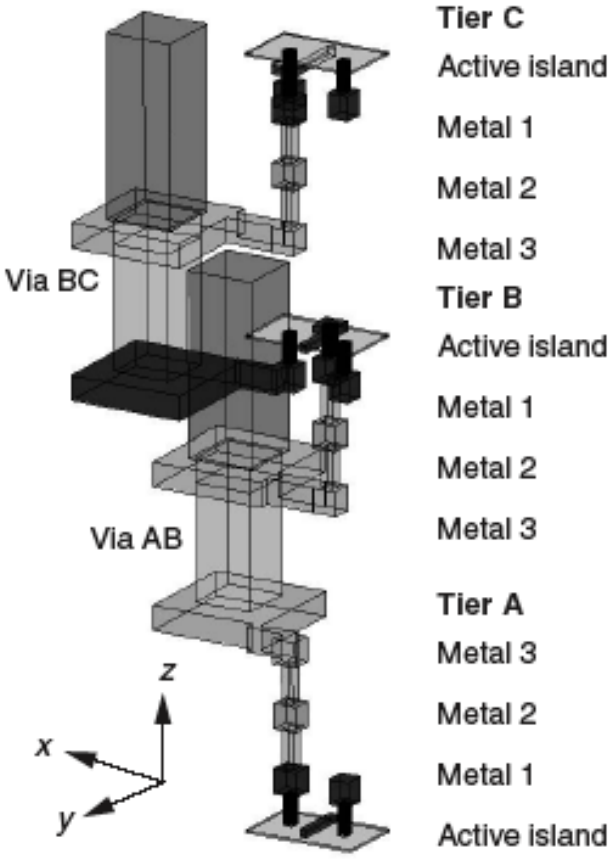


Figure 3.6: A 3D Model of the MITLL process, showing two inter-tier vias and one transistor in each tier. Extracted from (DAVIS, W. et al, 2005).

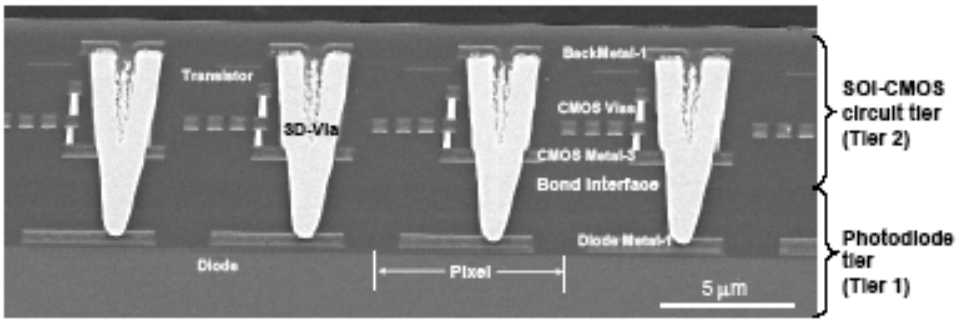


Figure 3.7: Cross-sectional picture of the 3D-Chip highlighting the 3D-Vias connecting them face-to-face. Extracted from (SUNTHARALINGAM, V. et al, 2005).

3.3.3 Summary of Important Data From the Studied Technologies

In this section, some important data for the development of the thesis is summarized. 3D-Vias are classified according to the following characteristics:

Table 3.4: Summary of collected data for 3D-Vias.

3D-Via	Integration Strategy	Tier Pitch	3D-Via Pitch	Occupy Active Area
Tezzaron (Copper Pads)	face-to-face	16-20 μm	2.4 μm	no
Tezzaron (Projected)	face-to-face	16-20 μm	1.46 μm	no
Microbump	face-to-face	16-20 μm	10-100 μm	no
Contactless (Capacitive)	face-to-face	16-20 μm	50-200 μm	no
MIT (Copper/Tantalum Pads)	face-to-face	16-20 μm	5 μm	no
TSV face-to-face	face-to-face	16-20 μm	0.5 μm	no
Tezzaron Super-Via^{TM}	face-to-back	15-20 μm	6.08 μm	yes
Tezzaron Super-Contact^{TM}	face-to-back	11-15 μm	< 4 μm	yes
Microbump 3D Package	face-to-back	11-15 μm	25-50 μm	no
Contactless Inductive	face-to-back	11-15 μm	50-150 μm	yes
MITLL Through Via (SOI)	face-to-back	9-12 μm	5 μm	yes
Through Via (regular Bulk)	face-to-back	11-15 μm	50 μm	yes
Back-to-back 3D-Via	back-to-back	6-8 μm	15 μm	yes

- The strategy used to integrate the tiers connected by the 3D-Via, that can be either face-to-face, face-to-back or back-to-back;
- The distance between adjacent tiers integrated by the 3D-Via (tier pitch);
- The pitch of the 3D-Via;
- Whether the 3D-Via occupies active area or not;

Given the thickness of the active area on each tier $thickness_{active}$ and the thickness of the metal layers $thickness_{metals}$ the distance between adjacent tiers t_i and t_{i+1} (tier pitch) will depend if they are integrated face-to-face, face-to-back or back-to-back , acording to equation 3.1. Patti (PATTI, 2006a) shows that on the Tezzaron technology, the active area on the edge tiers is thicker than on intermediate ones. This fact is ignored here for simplicity, since it will affect only the connections to the I/Os.

$$\begin{aligned} distance_{f2f}(t_i, t_{i+1}) &= 2 \times thickness_{metals} \\ distance_{f2b}(t_i, t_{i+1}) &= thickness_{metals} + thickness_{active} \\ distance_{b2b}(t_i, t_{i+1}) &= 2 \times thickness_{active} \end{aligned} \tag{3.1}$$

Thickness of a single metal layer is around 1 μm (PATTI, 2006b); the $thickness_{metals}$ can be approximated to the number of metal layers. Since it is hard to obtain thickness numbers from published works, the above equations are used to roughly obtain the tier pitch. A list of 3D-Vias and its characteristics is presented in table A.1.

We can observe that there is a variety of pitches while some 3D-Vias occupy active areas. Obviously that the best 3D-Via would be the one with the smallest pitch, with smallest tier pitch and not occupying active area. However, this combination is usually not available; let us analyze each variable individually:

- 3D-Via pitch is possibly the most important aspect to be considered, since a small pitch allows us to communicate more, have less restrictions to place 3D-Vias and possibly achieve a better design in terms of wire length and performance. From this perspective, the face-to-face connections are very attractive. For face-to-back, the Super-Contact from Tezzaron and MITLL SOI process are bellow 5 μm which is also quite acceptable to high communication demands, since it is smaller than most logic gates.

- Tiers pitch is used to compute the vertical wire length (in the Z axis).

- All microbump and face-to-face 3D-Vias do not consume active area, which is a very important aspect. Among them, face-to-face are the best choice considering their pitch requirements;

As a partial conclusion, it is quite understandable that face-to-face connection is the best choice considering the analyzed data, but it is limited to two tiers connection. For this reason, both MITLL and Tezzaron offer a face-to-face integration of the first two tiers while the rest are kept face-to-back. It would be beneficial to have a back-to-back integration because it would allow more face-to-face integrations in a stack.

It is reasonable to expect improvements on wire length (and delay/power as a consequence) of a 3D stack, specially in face-to-face where the 3D-Vias do not consume active area and have a better pitch. Obviously that 3D technology will impact design of chips; new design issues must be considered and actually the existing design methodologies must change to a new paradigm.

3.4 Potential advantages of 3D VLSI Circuits

By either analytical analysis (BANERJEE, K. et al, 2001) (DAS; CHANDRAKASAN; REIF, 2004a) (RAHMAN; REIF, 2000) (RAHMAN; REIF, 2001) and practical experimentation (DAVIS, W. et al, 2005) (DAS, S. et al, 2004) (KAYA, I. et al, 2004) (ABABEI, C. et al, 2005), it is well known that 3D circuit technology has the potential of providing many improvements to VLSI circuits, including:

- Largest wires reduction. It is very clear that wire length can reduce going to 3D. First let us study the case of the largest circuit wire. The maximum possible wire in a 3D chip equals to the half perimeter of the whole chip, which is $width + height$. On 3D, both $width$ and $height$ will go down since active area will be partitioned into two or more tiers. Davis (DAVIS, W. et al, 2005) reports that the largest wires are the ones with more shrinkage. Das also demonstrated this by experimental results (DAS, S. et al, 2004).

- Average wire length reduction. It is a conclusion of practically every paper on 3D circuit that the average wire length goes down. Das and Reif, for instance, report improvements in the order of 15% to 30% with 2 tiers from 20% to 50% with 5 tiers. This study is also subject of this thesis; experimental results are provided in section 4.10.

- Dynamic Power reduction. It is natural to consider that reducing overall wiring capacitance, power will go down. In fact, (RAHMAN; REIF, 2001) studied the resulting power dissipation in 3D circuits due to shorter connections and shorter clock

tree. It is well known that clock trees are responsible for a large chunk of the total power dissipated in a circuit. Considering both reductions, Rahman and Reif concluded that up to 22% power reduction can be achieved by migrating to 3D. Later, Das, Chandrakasan and Reif studied the same issue under performance-driven design (DAS; CHANDRAKASAN; REIF, 2004b) achieved power improvements in the order of 30% to 50%. Davis (DAVIS, W. et al, 2005) also discussed the issue of power on interconnects and clock, highlighting that on 3D-circuits we might use less repeaters (reduced short-circuit power); on his particular experiments a 23% improvement in total power was achieved.

- Timing improvement. Benefits from timing are expected because of shorter connections. Some papers already demonstrated this potential experimentally (ABABEI; MOGAL; BAZARGAN, 2005). Rahman and Reif (RAHMAN; REIF, 2000) studied this matter analytically with a critical path modeling; they reported that clock frequency can go up very significantly (by 50% to 100%) with the migration from 1 to 2 tiers. However, due to limited routing resources and 3D-Via issues, they reported that timing starts to deteriorate and possibly go back to the original value with 4 tiers and up.

- Chip area. Rahman and Reif (RAHMAN; REIF, 2000) studied the chip area as a function of wiring requirements. He demonstrated that chip area can dramatically reduce with the addition of active tiers.

All the potential advantages must be explored with proper CAD tools that must be able to define a new design paradigm solving new issues. The following section present a study of existing works on design impact of 3D circuits.

3.5 Impacts on Design

With the vertical stacking of chips, wire length can be significantly affected. For this reason, as studied in section A.2.2, there is a significant potential to be explored with 3D fabrication. On the other hand, the 3D fabrication opens a new design and CAD paradigm, specially on the physical design phase. The existing methodologies are not trivially expansible to 3D for two reasons: first, the circuit optimization rely on different tasks that includes placing logically neighbor elements in different tiers; second, the new design paradigm imposes new issues to be considered, such as thermal, 3D-Vias constraints and Yield. Existing methodologies for physical synthesis must be reviewed.

First, on section 3.5.1, we study the thermal issue, that comes from the fact that 3D provides more density and less temperature dissipation capability. Later, section 3.5.2 presents another limiting factor: yield. Finally, section 3.5.3 discusses the limitations related to the communication between tiers, that are accomplished by 3D-Vias.

All issues lead to the development of new methodologies for physical design of 3D chips. Section A.2.2.1 presents methodologies which differs from the granularity of their integration whose impact on design is studied.

3.5.1 Thermal Issues

The vertically stacked multiple active layers cause a significant increase on power density and, as consequence, high temperature spots. At the same time, 3D ICs are composed

of insulating material between the active layers, complicating significantly the heat dissipation. For instance, the thermal conductivity for the insulating material (epoxy) k_{epoxy} is 0.05W/mK, while the $k_{silicon}$ is 150W/mK and K_{copper} is 285W/mK.

Besides reduced reliability, higher temperatures lead to performance decrease and leakage current increase. Both Tezzaron and MITLL technology mentions that the limiting factor for stacking chips is the heat focus. It is easy to understand that intermediate tiers form islands of heat, since there is no good thermal conductor that can lead the concentrated heat to the outside of the chip. It is well known, though, that Through Vias can improve the heat dissipation significantly, since they are composed of metal connections that go though insulator layers. One of the most important techniques is the insertion of Thermal Vias (GOPLEN; SAPATNEKAR, 2005) (HUA, H. et al, 2006) that consists of carefully placed *dummy vias* (not connected to the circuit netlist) that serves only to the purpose of heat dissipation.

Another important heat dissipator is the existence of the back-metal layer, which is able to receive heat from the Through Vias connected to I/Os (or dummy vias) and dissipated directly in the surface of the circuit packaging. Packaging technologies can also be improved (RAHMAN; REIF, 2001)

It is widely discussed that the thermal issue is a very important drawback of the 3D circuits and research is needed to overcome this problem (ABABEI, C. et al, 2005) (DAVIS, W. et al, 2005) (HUA, H. et al, 2006) (DAS; CHANDRAKASAN; REIF, 2004b). Today, there exists good solutions that will be reviewed in the next sections. It is clear that there is an integration limit and it might not be possible to integrate a long stack of tiers (more than 5 tiers according to (HUA, H. et al, 2006)) due to heat dissipation, but for short stacks (2 or 3 tiers for instance) the existing techniques are able to provide acceptable dissipation.

3.5.2 Yield Issues

As already reviewed on section 3.1, while die-level 3D assembly stacks fully functional dies into a 3D chip, it is not possible to perform test before wafer-level assembly. Ignoring possible defects on the stacking process itself (which include to pierce the dies and add 3D-Vias), die-level assembly offers 100% yield. Under the same assumption, wafer-level stacking behaves exactly like the regular 2D chips (i.e. yield of one 2D chip with area a is the same as 3D stack of n dies with area a/n each) (PATTI, 2006a).

Tezzaron published that their 3D stacking process should not change device yields (except for the dies along the wafer's edge). They argue that yield is known as a problem for 3D circuits for a cultural issue that designers prefer to integrate larger modules on 3D and this obviously leads to a worse overall yield (PATTI, 2006a).

However, the assumption that Through Vias do not affect yield might not be sustainable. Reliability of 3D circuits might be affected with the Through Vias. Face-to-face connections (Copper Contacts, microbump, etc...) are better from the yield perspective than Through Vias since they do not need holes in the silicon.

3.5.3 3D-Via Issues

3D-Vias are vertical connections requiring large pitches with alien electrical and routing characteristics. This issue certainly introduce new CAD/design problems that must be solved in the future, such as the ones listed bellow and illustrated by figure 3.8. On placement problem, 3D-Vias introduce many constraints that must be accounted for. Some authors mention that the problem of handling 3D-Via contraints is actually too complex

(LIU, G. et al, 2005). On the other hand, 3D-Vias are important tools for improving the wire length of the circuit (DAS, S. et al, 2004) and should be used cleverly.

- 3D-Via upper-bound: given an area that will be partitioned into 3D, there is an upper bound driven by design and technology constraints to insert 3D-Via on that area; (figure 3.8.(a)).

- 3D-Via placement and legalization: 3D-Vias are also placeable objects and they should be placed legally (no overlaps with other 3D-Vias) in such a way that wire length deterioration is minimal; (figure 3.8.(b))

- Cell placement with movable obstacles: Through Vias (face-to-back) consume active area and cannot overlap with cells. Since they are movable objects, this problem can be stated as cell placement with movable obstacles; (figure 3.8.(c))

- Modeling of their characteristics: existing processes for 2D circuits are well known and well modeled. Designers already know how much resistance and capacitance a wire has, how they could be optimized, how much does wiring estimates are close to actual routing, etc. On 3D, connections must go to the top metal layer, pass though 3D-Vias whose electrical characteristics are not mature (new materials and methodologies are research topics). The modeling of those characteristics and its impact on design is a new issue to be considered; (figure 3.8.(d)).

- Routing resources: 3D-Vias consume a lot of routing resources and produce routing congestion (HUA, H. et al, 2006). This problem should be studied; (figure 3.8.(e)).

- 3D-Via trade-off: it was already shown that there is a trade-off between 3D-Vias and wire length (DAS, S. et al, 2004). Das et. al. developed an heuristic able to improve wire length with the addition of 3D-Via. Further investigation on this issue would be helpful to the development of new algorithms able to explore the trade-off; (figure 3.8.(f)).

- 3D partitioning for a stack of tiers: a set of cells connected by wires must be partitioned into a stack of tiers. Hypergraph partitioning is a similar problem (KARYPIS, G. et al, 1999), but it does not contain physical information, which in this case is uni-dimensional. Some published works such as (HENTSCHKE; JOHANN; REIS, 2006) and (ABABEI, C. et al, 2005) approached this problem by performing assignment of the partitions into tiers in such a way that the overall 3D-Vias count is minimized, as ilustrated in figure 3.8.(g). A 3D partitioner could be developed under the stack of partitions model providing a better solution for this problem.

- 3D-Via avoidance: 3D-Vias can harm circuit performance for certain connections in the netlist such as wires with a high switching activity or critical for timing. This issue should be studied further and algorithms that are able to impose those constraints without harming the solution are an important research topic to be considered; (figure 3.8.(h)).

It is subject of this thesis to be aware of the 3D-Vias during placement, so this issue will be further discussed on the next chapter.

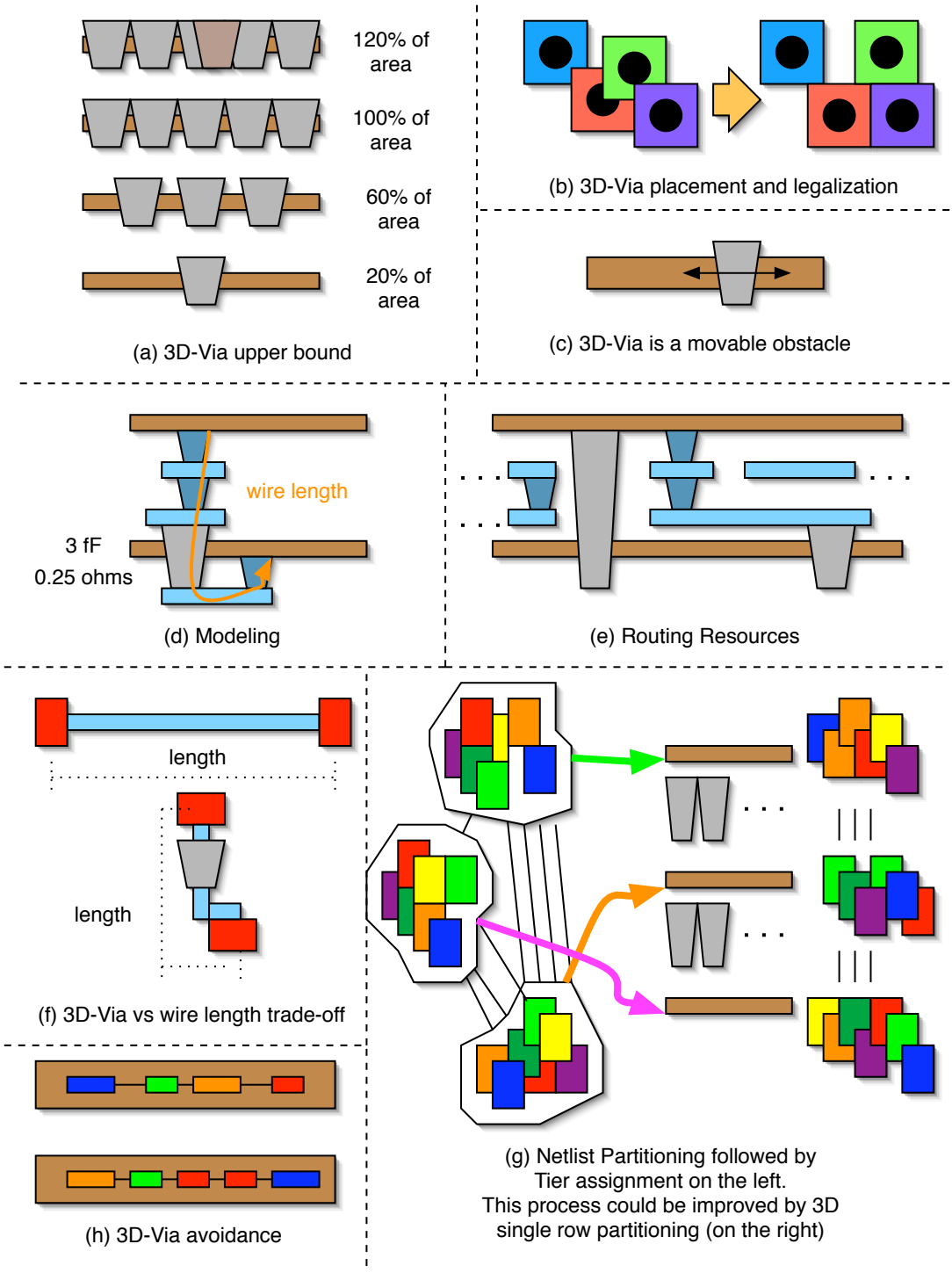


Figure 3.8: A summary of some new issues introduced in CAD and design due to 3D-Vias.

3.5.4 Design methodologies

This section explores three design methodologies that differ basically on their granularity with respect to the 3D integration. The first methodology, called *tier level integra-*

tion integrates separated tiers of different nature into a 3D stack. It is the most coarse level granularity and practically do not affect existing design methodologies, since each tier can be designed separately with a simple glue logic to integrate them. Secondly, the so called *ip core level integration* partitions big circuit blocks (ip cores) into different tiers, providing a tighter integration (more communication between tiers). Finally, *random logic level* partitioning breaks a single random logic block into 3D. This last one is a very fine grain methodology and at this level design and CAD will suffer from severe changes, specially in the physical level.

3.5.4.1 Tier Level Integration

In the SoC area, 3D fabrication actually opens many new possibilities. 3D integration makes it possible to arrange elements of the system in a chip and avoid common design problems, such as noise for analog circuitry, memory to processor bandwidth, etc. Different nature components (possibly manufactured from different processes), such as random logic, mixed signal, DRAM, SRAM, high performance and low-power logic, analog, RF, programable platforms (software, FPGAs, Flash) can be placed on different active layers, as shown in figure 3.9.

Patti, from Tezzaron, (PATTI, 2006a) present some nice designs performed with Tezzaron technology stacking different nature tiers. First, a Mixed Signal ASIC whose analog circuitry (including a heater resistor) is placed in one tier while a register file (digital) is placed on a second tier. Patti reports that there were no noise issue on the fabricated circuit; he also reports that the heater produced 8W on a $0.5mm^2$, which did not harm the circuit at all, leading to his conclusion that heat problems are not such an issue on 3D ICs. The second design is a CMOS photo sensor, that integrates a photo diodes array and additional circuitry such as amplifiers and A/D converters; Patti reports that the 3-D stacking allowed full array efficiency. Finally, Patti presents a processor and memory stack. Memory bandwidth is a known bottleneck on system's performance (DENG; MALY, 2005). Patti reports that the latency to access memory on this stack is 3 ns, which includes all latencies (for instance, wire delay). He also reports a 4Gbytes/s bandwidth on this connection, which is an order of magniture greater than his processor could reach, but demonstrates the effectiveness of the memory/processor stacking.

Obviously that this level of integration allows some important benefits without impacting design significantly, meaning that the existing tools and methodologies can be preserved. The following sections present new design methodologies that will impact significantly on design but there are more potential benefits to be explored.

3.5.4.2 IP Core Level Integration

This section discusses briefly the methodology of partitioning the design in the core granularity. The ideia, ilustrated in figure 3.10, is to apply a reasonable 3D floorplanning of the blocks and address wire length and timing improvement. Compared to the tier level integration, IP core integration demands more design effort to understand the migration to 3D and significantly affect the floorplanning phase on chip design. The downstream phases (placement and routing for instance) within the IP cores are unaffected.

Floorplanning is known to be a very hard problem and made by hand by most major companies in the industry, since automatic Floorplanning algorithms fail to be realistic and it is not easy to consider all the design issues. In the IP Core level 3D integration, the design expertise for the IP cores themselves doesn't suffer from severe impact, while the most important difference is on the 3D floorplanning step.

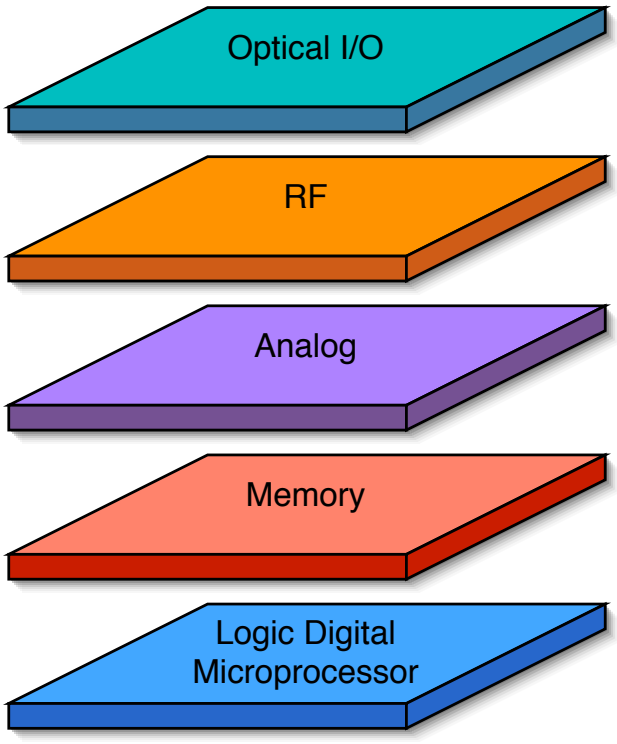


Figure 3.9: 3D circuit allows a mixed integration of different nature blocks with diminished noise effects

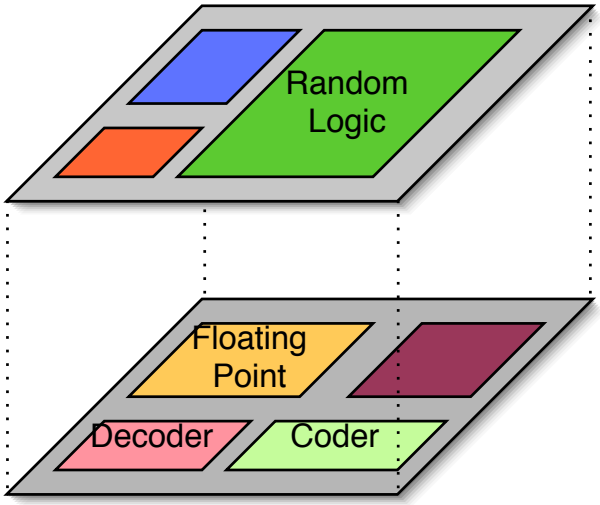


Figure 3.10: IP cores can be partitioned into 3D with improvements on circuit wiring

Some tasks and objectives of the 3D floorplanning method are listed below:

- Provide a good compaction of the blocks so that the higher density potential of 3D circuits can be explored.

- Provide a good partitioning of the blocks into circuit tiers, such that this partition aids the rest of the process to provide a good solution.
- Provide better wire length than existing 2D methods.
- Provide special attention to power and timing of some critical nets.
- Provide a reasonable solution to improve chip temperature.

Usually, the difference between floorplanners rely on the data structure used to store the layout and on the method to optimize it. While (BANERJEE, K. et al, 2001) studies the possible improvements analytically, the following works studied improvements into their practical frameworks.

Cong and Zhang (CONG; WEI; ZHANG, 2004) present a 3D floorplanner based on Simulated Annealing (KIRKPATRICK; GELATT; VECCHI, 1983). He introduced a CBA-T data structure that is applicable for 3D optimization of wire length. The temperature improvement is also one component of his optimization function, and it is based on three different thermal models that trade-off accuracy for run time. All of them do not consider the 3D-Vias as possible heat dissipator; the authors mention that this consideration could be incorporated into the Simulated Annealing engine. Their methodology partitions the blocks into tiers as the process advances, aiming at wire length and thermal optimization. Cong concluded that his methodology is able to reduce wire length by 29% and maximum on-chip temperature by 56%.

Li et al. (LI, Z. et al, 2006) studied the methodology for partitioning IP blocks into 3D in a two step approach, which they call hierarquical. The first step is to partition the blocks and after that (second step) they are placed on a respective tier. First, they highlight that the first step (block partitioning) reduces the problem size for the subsequent stage, providing faster convergence. On the other hand, that process also limits the algorithm capability to obtain good wire length solutions. They observed that min-cut partitioning (which leads to better number of 3D-Vias) is not good for wire length while increasing the cut would provide a more favorable search space to reduce wire length. This fact could lead to the conclusion that a “max-cut" approach would be better, but Li also demonstrates that this fact is not true into his framework. Actually, the best solution could not be predictable using only net-cut based methods (observation that contradicts with (YAN, T. et al, 2006)), but some additional wire length prediction scheme should be used. In their paper, Li et. al use a statistical wire length estimator that leaded to good floorplanning solution under 3D-Via upper bound and area balance constraints. They do not optimize chip temperature.

Healy et. al (HEALY, M. et al, 2007) proposed an hybrid method for 3D floorplanning optimization based on Linear Programming and Simulated Annealing. One very interesting discussion in their paper is regarding the integration strategy used, which might encourage or discourage the use of 3D-Vias. They discuss that Through Vias actually should be avoided (due to large pitch and yield problems) while face-to-face vias can be used plenty, since they will contribute to reduce wire length. Their approach provide an accurate model to measure chip temperature. The optimization algorithm optimizes multiple objectives, such as wire length and temperature reduction not inserting thermal vias.

Cong and Zhang (CONG; ZHANG, 2005) published, one year after the previously cited paper, a method to optimize heat using thermal vias. They conclude that their method could be used within a placer or floorplanner.

Wong and Lim (WONG; LIM, 2006) discussed the issue of inserting thermal vias in their 3D floorplanner. They proposed a fast thermal model based on random walks. The paper develops different combinations of the thermal optimization methods thermal aware floorplanning and thermal vias insertion. They concluded that thermal vias should not be separed from the floorplanning optimization. Using it isolated after thermal aware floorplanning, thermal vias can contribute to a heat increase under their model. They developed a method for an integrated thermal optimization that led to the best results of reducing temperature by 38% with 47% penalty on area and 22% on wire length penalty. They also highlight that thermal vias alone are able to deliver 17% heat improvement with almost no cost to wire length (4%).

In summary, thermal optimization within floorplanning is able to provide very considerable thermal optimization, according to the studied works.

3.5.4.3 Random Logic Level Integration

Figures 3.11 and 3.12 summarize the random logic level integration. This section discusses the idea of a very tight integration method, in which a circuit at the cell level is partitioned into several tiers leading to new design issues on random logic blocks. Some examples of important changes: timing analysis and synthesis should consider that some connections will go to the top metal layer and pass though a 3D-Via with different electrical characteristics; cell placement should be aware of new obstacles (through vias); routing algorithms must be aware of resources taken by 3D-Vias, etc. In fact, the whole CAD flow from logic synthesis to layout must be revisited.

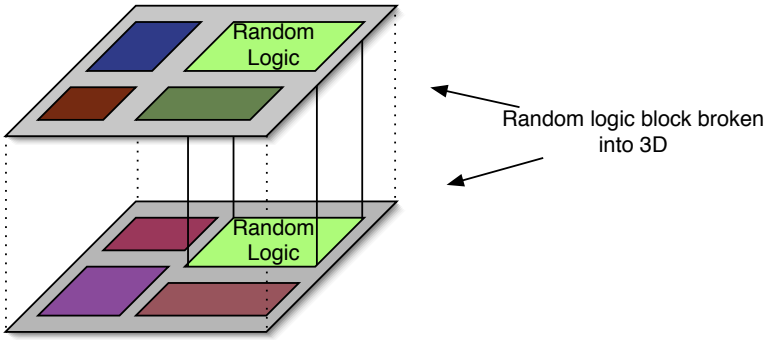


Figure 3.11: Random logic blocks could be broken into 3D.

People are pessimistic on this field because they know that very tight integration of tiers lead to more 3D-Vias and consequently those issues discussed on section 3.5.3 become more critical. The tighter the integration, more 3D-Vias can potentially be used. As already studied on previous sections, Through Vias actually lead to less reliability and possibly worse yield. On the other hand, wire length can be improved and face-to-face connections could be used as needed with no extra cost. Let us discuss this possible trade-off between more complicated design issues with important circuit improvements on wire length, timing and power.

Stating the motivation, it is common sense the connections are a bottleneck on circuit performance (which is related to wire length, timing, power and manufacturability). Shrinking used to be one good methodology to improve chip performance in the technology level, but in the recent years shrinking is actually contributing to the wire criticality

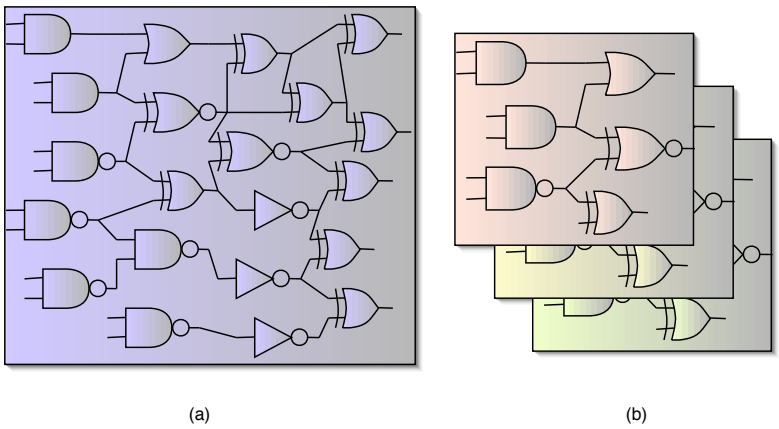


Figure 3.12: Logic into 3 tiers.

became even more important (BANERJEE, K. et al, 2001). More importantly, shrinking is approaching a feasible limit since feature sizes are in the atomic scale. Design rules are getting extremely complex and there are a few companies that will be able to design on 45nm; the design expertise for those devices will became prohibitively expensive (DAVIS, W. et al, 2005). The existing scenario on the technology perspective is very pessimistic for the future. A change of design paradigm is a realistic solution to achieve timing and power improvements with existing technologies. It might be cheaper to develop new methodologies on the 3D world rather than the complex world of design for manufacturability on 65nm and beyond.

While IP-Core level 3D integration can manage the overall chip performance, a particular random logic block is kept on a single tier. Today, circuits can have tens of millions of cells and for the future we can expect an even larger amount of cells. A block with so many cells can definitely be improved by 3D integration.

The most affected CAD stage is cell placement. On the thermal issue, some of existing works optimize heat either by spreading the cells on hot spots (ABABEI, C. et al, 2005) or by shortening nets with higher switching activity (OBERMEIER; JOHANNES, 2004). However, thermal issues could be potentially solved on the floorplanning level (see previous section) while 3D-Via issues are very critical on placement level. Placement methodologies with control of the amount of 3D-Vias into feasible margins while optimizing wire length and delay is the challenge to be discussed in the next section.

3.6 Impacts on cell placement

The integration level of random logic was selected as a subject of this thesis because of the reasons stated in the previous section. For now on, let us assume this strategy and focus on the impacts it has on design, specially on cell placement.

Cell placement is one key step on computer aided design of VLSI circuits. It is the major responsible for the overall wire length of a particular block of random logic and indirectly responsible for timing and power. Even being researched for more than three decades, cell placement is still an open problem (no optimal or common sense best solution), because problem sizes grows, new technology issues are introduced and more flexibility is required (related to blockages, congestion, etc.). Some of the placement ad-

vances on 2D are directly applicable for 3D, even for slightly different problems. A list of those features is presented bellow:

- Wire length optimization (2D)
- Low complexity (and fast) algorithms
- Blockage aware placement
- Mixed size placement
- Timing-driven placement
- Power-driven placement
- Thermal-aware placement

With the introduction of new issues stated in section 3.5, 3D placement actually demands solutions for the problems stated bellow.

- 3D-Via placement and legalization
- 3D-Via aware placement
- True 3D Wire length optimization
- Thermal aware 3D placement
- Movable blockages placement
- Timing driven 3D placement

In the next subsections each of the advances is reviewed with published works. A review of existing placement algorithms is presented. Also, the new issues of 3D circuits are presented with some published works if applicable. Note that some of this new issues are still ignored by most of the existing literature on 3D placement.

3.6.1 Review of existing placement algorithms

The placement problem is a np-hard problem. The problem named *Optimal Linear Arrangement* (GAREY; JOHNSON, 1979) that can be mapped as cell placement is proven to be np-hard.

Since placement is a np-hard problem and circuit sizes are very large and placement should be optimized for various objectives, designing proper placement algorithms is a hard task.

The placement algorithms can be classified in three big categories, as follows: **Simulated Annealing** approaches such as (SECHEN; VICENTELLI, 1985) (TAGHAVI; YANG; CHOI, 2005), algorithms based on **Recursive Partitioning** (ROY, J. et al, 2005) (AGNIHOTRI; ONO; MADDEN, 2005), and **force directed methods** (EISENMANN; JOHANNES, 1998). An important subset of the force directed methods are the **analytical methods**, such as the quadratic placement techniques (OBERMEIER; RANKE; JOHANNES, 2005) (HU; ZENG; MAREK-SADOWSKA, 2005) (KAHNG; REDA; WANG, 2005) (VISWANATHAN; PAN; CHU, 2005).

Table 3.5: Summary of placement algorithms and related tools

Algorithm	Simulated Annealing	Partitioning Based	Analytical
Iterative/Constructive	Iterative	Constructive	Constructive
CPU requirement	High	Average	Low
Versatility	High	Average	Average
Capo		X	
Kraftwerk			X
APlace			X
Fastplace			X
mFAR			X
Proud			X
Dragon	X	X	

The algorithms based on Simulated Annealing are mostly CPU time consumers because they perform a large search to avoid local minima. On the other hand, SA placers, such as Dragon (TAGHAVI; YANG; CHOI, 2005), are known to achieve high quality placements. Actually, in the internal structure of the Dragon placer, a partitioning process is used to aid the SA engine to save CPU time. This feature is absolutely necessary for medium size designs and up (more than 5K cells).

The placement algorithms based on recursive bisection uses a partitioning algorithm to place the cells introducing greedy decisions that serves the purpose of speeding up the algorithm. Heuristic partitioning algorithms have a lower time complexity compared to SA based methods.

The Force Directed method is an iterative method that applies forces to the gates, one by one, repeating this process many times until the placement stabilizes at some final solution. In the quadratic placement method (first proposed by Hall (HALL, 1970)), the objective function is the squared wire length. By finding the derivative of the objective function, a system of equations is derived and can be solved by analytical methods. The idea can be understood as a spring system. Cells and pads that are connected with each other have attraction forces modeled by a spring (Hooke's Law). The spring system's equilibrium state is equivalent to the optimal placement with a quadratic wire length objective function. Quadratic placement methods reach the optimal placement for a squared cost function disregarding cell overlaps. There are many techniques to remove the overlaps while spreading the cells. The method is known to be very fast and achieve good results. Many of the industrial placement tools are based on quadratic placement. However, the three techniques are used in both industry and academia and it is hard to define which one is the best.

Placement algorithms are usually classified as *iterative* and *constructive*. Iterative algorithms can be defined as algorithms that *can* optimize an already existing placement, while constructive algorithms must start from scratch. In the absence of an initial placement, an iterative algorithm will typically start from a randomly generated solution.

A summary of the reviewed algorithms is presented on table 3.5 as well as academic placers.

3.6.2 Low complexity (and fast) algorithms

There are two important issues for cell placement: wire length estimation and effective final placement. While performing logic synthesis, wire estimates are of great value to optimize timing. A cell placer is able to provide very accurate wire estimation. For this application, CPU time of the placer is critical (since it is called repeatedly) while quality requirement is relaxed. For the final placement, quality must be highly optimized while CPU time is not an important issue but still must be within feasible limits.

Over the years, problem sizes grew dramatically, so cell placement started to require algorithms with special concern to CPU time. For 3D circuits it is reasonable to expect that circuit sizes will grow as well. Among the studied algorithms, partitioning analytical methods outperformed others like Simulated Annealing specially for their reasonable time complexity. However, Simulated Annealing optimization is still very attractive under certain control over run time and problem size, since it is able to achieve good quality. Nowadays, Simulated Annealing is employed on floorplanning problems (whose problem sizes are small), constrained and windowed detailed placement (since placement is constrained to a windows, effective problem size is dramatically reduced) and other specific side optimization tasks needed on most CAD tools.

Today there are many placers with very acceptable run times. In this particular review, Fastplace was selected to be detailed since it is one of the fastest among existing placers. More details of the algorithm are presented in section 4.5.1 of this text.

Fastplace (VISWANATHAN; PAN; CHU, 2005) is based on the Quadratic Placement method. It is done in three distinct steps: coarse global placement, wire length improved global placement and detailed placement.

The main objective of the first stage is to minimize wire length and spread cells evenly on the placement area. A *cell shifting* algorithm provides repulsive forces to spread the cells. In the Cell Shifting, the placement region is first divided in several bins; the utilization of each bin is determined to shift cells from highly utilized bins to the ones with empty spaces.

The second step also perform cell shifting, but adds a new greedy algorithm that provides linear wire length (half-perimeter) and utilization improvement.

The third and last stage of Fastplace is responsible for placement legalization, associating cells to pre-determined rows and removing all overlaps. A greedy method is used to reduce wire length further.

3.6.3 Blockage Aware Placement

Blockage Aware placement is a very fundamental problem on cell placers. Suppose that cells must be placed in an area that will actually be shared with macro blocks. There are placeable areas and non-placeable areas for the cells. The non-placeable areas are called blockages or obstacles in existing placement literature. Blockage aware placement was one subject of evaluation in the 2005 ISPD Placement Contest (NAM, G. et al, 2005).

There are many techniques applicable to this particular problem. Some authors ignore blockages initially and afterwards legalize the placement. One possible technique for that are flow based legalization methods (BRENNER; VYGEN, 2004). Another possible technique is landscape smoothing, used by APlace (the winner of ISPD 2005 Placement contest) (KAHNG; REDA; WANG, 2005).

Another possibility is to consider cells as soft blocks and run a floorplanner (maintaining all hard blocks fixed). This approach is explored by (VISWANATHAN; PAN; CHU,

2005). Many other methods are available in the literature. All placers that participated on ISPD 2005 and 2006 contests provide ways to handle obstacles. More details can be found into their papers (each placer had a published paper in the regular conference proceedings).

Blockage Aware placement can definitely find applications on 3D placement. Besides regular blockages, such as macros, 3D-Vias can also be considered as blockages because they occupy active area that cannot be used by cells.

3.6.4 Mixed Size Placement

Many existing designs contain blocks of random logic mixed with movable macros. A macro is usually higher than a row; for this reason, the algorithm to place macros must be different than usual placement algorithms. The problem to place movable macros together with cells is called Mixed Size placement. Mixed size placement problem is being largely studied in the recent years (VISWANATHAN; PAN; CHU, 2006).

Capo (ROY, J. et al, 2005) was initially proposed by Caldwell in (CALDWELL; KAHNG; MARKOV, 2000). It is basically a min-cut bi-partitionning based algorithm; it was extended to handle mixed-size placement with the aid of an incorporated floorplanner named Parquet. When a placement bin contains a larger module, it switches to local floorplanning, using the bin boundries as the available area.

Usually, the most important difference from placement to floorplanning are the data structures applied to represent a solution. Parquet works with two data structures: sequence-pairs[1] and B* Tree (YAO, B. et al, 2001). Sequence pairs produces better wire length while B* Tree packs better. The floorplanner can switch from one data structure to the other if needed for a better result.

Viswanathan et. al recently presented a new version of its Fastplace algorithm to handle mixed size blocks (VISWANATHAN; PAN; CHU, 2006). They argue that force-directed methods (such as quadratic placement) "can seemlessly handle the varied sizes of placeable objects without employing additional techniques like partitioning and clustering". In the paper, an extension of the cell shifting method is provided to handle mixed-size blocks; the quadratic optimization method itself is not changed, but the cell shifting method needs to consider the block dimensions in order to compute bins utilization and add spreading forces. Two algorithms for mixed size placement legalization are introduced: initially the macros are legalized using sequence pairs and Simulated Annealing optimization of those blocks. After blocks are legalized, they are fixed as blockages and cells are legalized within the placeable segments delimited by the macro blocks.

3.6.5 Timing-Driven Placement

Timing driven placement is classified in the literature as path-based methods and net-based methods. One approach to net-based methods is to provide a maximum length budget to certain critical nets, such as the work on (OGAWA; PEDRAM; KUH, 1990). The most common approach is net weighting. Examples of net weighting methods are (KAHNG; WANG, 2004) (XIU; RUTENBAR, 2005) (RIESS; ETTELT, 1995) (KONG, 2002) (YANG; CHOI; SARRAFZADEH, 2002) . Although the method is very simple, the problem rely on obtaining the appropriate weight. Kong (KONG, 2002) use a path-counting method to obtain net weights. Other methods are based on slacks, such as

[1] Sequence pairs are a classical data structure used by floorplanning algorithms based on Simulated Annealing

(KAHNG; WANG, 2004), (LUO; NEWMARK; PAN, 2006), (XIU; RUTENBAR, 2005). Net weights can be dynamically updated during cell placement since wire related delay can modify slacks. An example of such method is (RIESS; ETTELT, 1995). The path-cased methods are known to be more precise than net-based, but more complex demanding higher CPU times. One example is the method from Swartz and Sechen (SWARTZ; SECHEN, 1995) that is similar to the Z-Place approach on detailed placement phase. Their method computes dynamically a certain number of most critical paths during the Simulated Annealing cell placement. Instead of weighting the nets, Swartz separate the wire length computation in regular wire length and critical wire length while the weighting relies only on the critical portion.

Some path-based methods use more tricky techniques that indirectly address timing improvements. The work on (HWANG; PEDRAM, 2006) avoids zig-zags on placement of critical paths by assigning a path to a signal direction and maximizes the monotonic behavior of those nets. Chou et. al (CHOU; LIN, 2002) inserts artificial nets that connects the starting and end point of the critical paths, reducing the overall wire length of the path.

On 3D circuit, there is no novel technique published yet. Ababei et. al (ABABEI, C. et al, 2005) mention that net weighting was used within their FPGA 3D placement tool, leading to 20% delay improvement.

3.6.6 Power-Driven Cell Placement

On power driven cell placement, besides doing the regular job of reducing wire length (that leads to power improvement), two kinds of procedures can be performed:

- Power distribution
- Identification of critical wires for power

Power distribution usually serves the means of thermal improvement, that will be revised in the next section. The second mentioned technique (OBERMEIER; JOHANNES, 2004) searches for wires with higher switching activity in the logic level and perform extra optimization of those wires by overweighting them (similarly to the weighting technique for timing critical nets).

On 3D circuits, power is one potential improvement that comes directly from the reduced wire length and clock tree size; additional improvement can be provided with identification of critical wires.

3.6.7 Thermal-Driven Cell Placement

Thermal-driven placement of 2D circuits is an issue studied by some papers on the past. Tsai et. al (TSA; KANG, 2000) presents a placement algorithm that targets an even distribution of the power consumption throughout the circuit. This optimization is part of a multi-objective function that includes thermal improvement and wire length. Wires are ignored on Tsai's work; cells are weighted according to their switching activity.

Thermal-Driven placement gained considerable strength on 3D placement. Just like floorplanning algorithms (reviewed in section 3.5.4.2), placement algorithms need a fast thermal model in order to identify hot spots. Besides estimating, each different placer must incorporate the thermal analysis into their algorithm core.

Goplen and Sapatnekar presented in (ABABEI, C. et al, 2005) and (GOPLEN; SAPATNEKAR, 2003) a force directed cell placer with thermal forces that improves chip

temperature. The thermal forces move the cells away from high temperature spots. Additionally, there are attractive forces to the other cells and I/O pins of the same net and repulsive forces to unmake cell overlaps. Repulsive forces related to cell overlaps and heat improvement are refreshed at each iteration. The authors reports average improvements in the order of 17% to chip temperature. The work does not consider 3d-Via costs.

Balakrishnan et. al (BALAKRISHNAN, K. et al, 2005) propose a two step 3D cell placer: first a partitioning based algorithm provides an initial placement solution focused on wire congestion and dynamic power consumption reduction. Note that power consumption impacts the thermal profile of the circuit. After that, a Simulated Annealing based refinement stage improves chip temperature, wire length, congestion and number of 3D-Vias with a multi-objective cost function. The authors observed that wire congestion is correlated to temperature hot spots based on experimental results. Their method provides an interesting trade-off between the variables in the objective function. The best configuration for chip temperature resulted in 20% to 30% increase in the other objectives while the worst temperature improvement resulted in 8% to 10% increase compared to the baseline (no temperature improvement).

3.6.8 True 3D engines

A cell placer (2D circuits) is considered a 2D placer because it is able to move cells in two dimensions and keep track of wire length measured in 2D as well. For a 3D placer it is expected that both capabilities of movement and wire length measure to be expanded to 3D in order to effectively cover the solution space. This change, though, besides being a considerable step in terms of updating tools and algorithms, modifies considerably the search space of placement algorithms. For both reasons, many existing placers still fail to provide this capability.

The works from (ABABEI, C. et al, 2005), (DAVIS, W. et al, 2005), (DENG; MALY, 2001) for instance, apply min-cut partitioning (usually with hMetis tool (KARYPIS, G. et al, 1999)) to assign cells into tiers, minimizing the 3D-Vias count. A subsequent step performs 2D placement on each tier separately; the already placed tiers can serve as a guide to subsequent tiers in order to minimize wire length. However, (LI, Z. et al, 2006) (KAYA, I. et al, 2004) (LIU, G. et al, 2005) (DAS; CHANDRAKASAN; REIF, 2003) already identified that this approach leads to worse results in terms of wire length.

Liu et. al (LIU, G. et al, 2005) builded a two step 3D placement flow similar to the one mentioned above using hMetis for partitioning the cells into tiers. They argue that building a true 3D flow is very hard and for this reason they concentrate on improving the partitioning step. They observed that the insertion of 3D-Vias could potentially improve wire length. For this reason, their cell partitioner does not perform min-cut partitioning, but tries to maximize the 3D-Vias under an upper bound constraint. In fact, since face-to-face integration allows 3D-Vias with no cost to yield or area, they could be inserted freely in order to improve wire length. Some preliminary evaluation could be performed to analyse a reasonable upper bound for those 3D-Vias. Liu's algorithm cannot achieve the exact via count provided, but tries to get a close approximation using an iterative algorithm. After the tier assignment, the algorithm uses the Capo tool (CALDWELL; KAHNG; MARKOV, 2000) to place the cells in each tier.

Das et. al (DAS; CHANDRAKASAN; REIF, 2003) (DAS, S. et al, 2004) builded a true 3D partitioning based placement engine. It recursively cuts the placement cube performing min-cut partitioning. A wire length and 3D-Via trade-off can be obtained by controlling the instant at which the cut is performed into the Z axis (e.g. the iteration at

which the design is partitioned into tiers). The optimal solution for wire length is obtained when the aspect ratio drives the cut direction (same observation provided by (YILDIZ; MADDEN, 2001)). The solution with fewer 3D-Vias can be obtained in the case where the first cut is made on the Z axis (method that would be equivalent to the ones based on hMetis assignment mentioned above).

Goplen and Sapatnekar (GOPLEN; SAPATNEKAR, 2003) formulate the 3D placement problem as a true 3D placement. They provide an analytical force directed algorithm that minimizes the squared 3D wire length. Their method is iterative; at each iteration repulsive forces related to thermal issues or cell overlaps are inserted in the system. This process makes cells spread into the placeable volume. The authors do not detail how they handle I/Os into the tiers; however, on quadratic placement methods the cells will not move in the Z axis unless the I/Os are placed in different tiers. If the I/Os are fixed in one tier, it can be understood that the repulsive forces are the only responsible for moving cells into other tiers. After placement is completed, the cells are sorted in the Z axis and finally assigned to a circuit tier. This method may fall into a false wire length optimization since actually cells cannot be placed into continuous z coordinates; the rounding of their coordinates could potentially increase circuit wire length.

Obenaus et. al, in (OBENAUS; SZYMANSKI, 1999), present an iterative force directed method for 3D placement. Different from Goplen's placer, it is not an analytical method but it moves each cell to an optimal position by fixing all other cells. They define the 3D placement problem to minimize wire length only, which handles the problem as true 3D method. 3D-Via costs and other constraints are not considered. No repulsive forces are added to the system; a bucket re-scaling method similar to cell shifting (VISWANATHAN; PAN; CHU, 2005) spreads out the cells.

In general, the drawbacks of the above mentioned works on true 3D placement is the lack of consideration to the integration strategy (since face-to-face integration encourage 3D-Vias while face-to-back discourage), the lack of a proper and legal place for the 3D-Vias (except for Kaya's placer), the balanced area distribution for the cells along the tiers with no consideration of active area occupied by Through Vias (except for Kaya as well) and finally no control over critical nets for timing. For the special case of Obenaus et. al there is an aditional drawback on the lack of realism on their method since they consider all cells as perfect squares.

3.6.9 3D-Vias placement

Between each pair of adjacent tiers there exists a set of 3D-Vias connecting them. Those 3D-Vias must be assigned to a (x, y) position; considering their required pitch, the assigned position cannot overlap with any other 3D-Via between the same tiers. This problem can be solved in the routing phase (LIM, 2005) but this work handles 3D-Vias as placeable objects in the placement level. A 3D-Via placement and legalization problem can be defined for every via layer (this concept is further discussed in section 4.9). Any 3D-Via is connected to a net and it can be placed anywhere within the net boundary with no harm to the circuit wire length. However, it might not be possible to provide this place; in such cases, there is an overhead on wire length that must be considered on the final wire length.

Yan et. al (YAN, H. et al, 2005) presents a very simple ILP formulation for this problem. First, a grid of valid places is defined based on the pitch requirement of the 3D-Vias. They compute a cost matrix that establishes a cost function for every via in every possible place in the layout. This cost could be wire length, power, thermal effect

or any other. Solving the ILP problem leads to a legal solution for 3D-Vias placement with minimum cost. However, this method has a strong weakness with respect to the problem size. Since it generates one variable for every pair of 3D-Via and valid grid placement, the number of variables generated is huge and cannot be supported by most of existing ILP solvers. For example, suppose a grid of 300x300 via placements and 1000 3D-Vias connecting a pair of tiers. In this case, the ILP formulation would have 900 million variables. The authors presented methods to reduce the problem size that slightly affect the solution quality. More analysis on Yan's method is provided in section 4.9.

3.6.10 Movable Obstacles in Placement

It was already mentioned that the Through Vias occupy active space and for this reason should serve as a obstacle to cell placement. Since 3D-Vias are also placeable objects (as reviewed in previous section 3.6.9), they are actually movable obstacles.

Kaya et. al in (KAYA, I. et al, 2004) present a force directed 3D placer that have some nice properties to handle and place 3D-Vias. Basically, any Through Via occupies active area and is placed within the circuit layout just like regular cells. This assumption apparently mean that 3D-Vias are constrained to have the same height (or smaller height) of cells, which is unrealistic; however, there is no evidence of that constrain in the papers. Mixed-size placement methods, for example, could be used (as the ones reviewed in section 3.6.4).

We can assume that this method is also a 3D-Via placement method, but it is applicable only to face-to-back (or back-to-back) Through Vias. Face-to-face Vias still need to be placed by some other method.

3.6.11 Summary of reviewed works on 3D placement

As reviewed above, a number of important features are incorporated in some existing 3D placers. Some of those features are: (A) True 3D engine (ability to optimize and measure wire length on 3D), (B) 3D-Via trade-off (ability to trade 3D-Vias for wire length), (C) 3D-Via upper bound, (D) sensibility to the type of 3D-Via (face-to-face or face-to-back), (E) active area balance considering area occupied by 3D-Vias, (F) 3D-Via Placement, (G) 3D-Vias occupying legal active area, (H) consideration to critical paths. Table 3.6 summarizes the reviewed algorithms and their features.

Table 3.6: Feature list of existing 3D placers: (A) true 3D, (B) 3D-Via trade-off, (C) 3D-Via Upper Bound, (D) Different kinds of 3D-Via considered, (E) Area balance considering 3D-Vias, (F) 3D-Via Placement, (G) 3D-Vias occupying cells space, (H) Concern to critical paths

Features:	(A)	(B)	(C)	(D)	(E)	(F)	(G)	(H)
(DAS, S. et al, 2004)	√	√	√					
(GOPLEN; SAPATNEKAR, 2003)	√							
(DENG; MALY, 2001)			√					
(OBENAUS; SZYMANSKI, 1999)	√							
(LIU, G. et al, 2005)		√	√					
(KAYA, I. et al, 2004)	√		√		√	√	√	
Z-Place	√	√	√	√	√	√		√

3.7 Z-Place Overview

Considering the potential of cell level integration discussed in section 3.5.4.3, 3D placement is an important research topic. There are limitations to be overcome by cell placers, as highlighted in previous sections. We propose a tool called **Z-Place** that integrates a full 3D placement flow, providing algorithms to the 3D placement community that could potentially be used in any tools. The tool targets the following features:

- Quadratic Placement engine: Quadratic Placement is one of the most successful algorithms for cell placement, as reviewed in section 3.6.1. It has good scalability and run time so it is usable for larger designs. More importantly, most of the existing placers in the industry and academia are using and researching improvements on this algorithm,. Recently proposed methods, such as (SPINDLER; JOHANNES, 2006), could be used.

- Handling of I/O pins: Quadratic Placement requires fixed pins in the boundary in order to compute a solution. Z-Place provides a good method to partition and place the I/Os into all tiers in such a way that this method actually helps the rest of the placement process to reduce 3D-Vias.

- True 3D engine: Z-Place incorporates a true 3D Quadratic Placement engine since it is able to move cells in all directions at the same time and measure 3D wire length.

- Area balancing sensible to the integration strategy: Z-Place allows optimization of mixed integration designs (i.e. face-to-face, face-to-back and back-to-back); it balances the area dynamically according to the amount of 3D-Vias assigned.

- 3D-Vias upper bound: Z-Place is aware that random-logic level integration might lead to an unacceptable 3D-Vias count. To solve this, Z-Place trades-off 3D wire length and 3D-Vias under an upper bound, that is also sensible to the integration strategy and 3D-Via pitch. For face-to-face integration, the constraint is relaxed to be close to the available area, improving 3D wire length.

- Multi-technology support. As reviewed in section A.2.1, the overall scenario of 3D fabrication technologies offer a wide range of 3D-Vias pitches and lengths, which impacts 3D placement very importantly. In Z-Place, technology data is a parameter of the tool, that will try to take the best benefit used of the technology. If its 3D-Vias are costly, Z-Place will avoid them while if they are cheap, Z-Place will make good use of them to improve wire length.

- 3D-Via placement: Z-Place is able to efficiently place the 3D-Vias with no overlap with each other such that wire length is minimally affected.

- Critical path consideration: Z-Place is able to identify critical paths from timing (or power) analysis and avoid the use of 3D-Vias for those paths.

A detailed description of Z-Place features and algorithms supported by experimental results is provided on the next chapter.

4 Z-PLACE: ALGORITHMS FOR 3D PLACEMENT

4.1 Introduction

Z-Place is a 3D cell placement tool. As primary objective, Z-Place tries to obtain the best possible wire length for a given netlist under certain user configurations and 3D-Vias related constraints. The constraints are defined to obtain room to place 3D-Vias. In the following sections the flow as well as all the algorithms used on Z-Place are presented in detail.

4.2 Placement benchmarks

Experimental results provided in this text are based on the ISPD 2004 benchmark set (ISPD04 - IBM STANDARD CELL BENCKMARKS WITH PADS., 2004) summarized in table 4.1. In the IBM suite, some cells that were originally multi-row cells considered as standard-cells (having their heights cropped to one row). Another set with timing information is generated with a tool to map a circuit from VHDL to a placement benchmark in the bookshelf format. Table 4.2 provides information on the VHDL designs and their netlist after high level and logic synthesis.

4.3 Proposed 3D Placement Flow

Z-Place picks a circuit netlist with I/O pads and places the entire circuit into 3D. The following are inputs of the tool:

- List of cells (with individual information of width and height - note that all cells must have a same height);
- List of I/O connections;
- List of nets (connections between cells and/or I/O pins);
- Description of physical area planned for a 2D circuit (width, height of the block with I/O pins placed in the boundary) - this information is not needed if an appropriate area definition in 3D with I/Os in the boundary is already provided by the user;
- Circuit information: number of tiers and integration strategy;
- Technology information: pitch and length of 3D-Vias

Table 4.1: Benchmark Information of IBM Suite.

Bench	# Cells	# Terminals	# Nets	Cell Area	Circuit Area
ibm01	12,506	246	14,111	2,141,920	2,380,800
ibm02	19,342	259	19,584	2,757,540	3,064,208
ibm03	22,853	283	27,401	3,375,870	3,751,968
ibm04	27,220	287	31,970	4,303,300	4,782,848
ibm05	28,146	1,201	28,446	4,471,520	4,968,416
ibm06	32,332	166	34,826	3,695,860	4,106,592
ibm07	45,639	287	48,117	6,422,050	7,136,672
ibm08	51,023	286	50,513	6,663,260	7,403,840
ibm09	53,110	285	60,902	7,755,070	8,617,104
ibm10	68,685	744	75,196	12,664,400	14,073,696
ibm11	70,152	406	81,454	10,010,800	11,125,504
ibm12	70,439	637	77,240	13,603,400	15,116,288
ibm13	83,709	490	99,666	11,592,600	12,880,896
ibm14	147,088	517	152,772	21,534,200	23,931,520
ibm15	161,187	383	186,608	21,174,900	23,532,192
ibm16	182,980	504	190,048	27,836,100	30,930,192
ibm17	184,752	743	189,581	33,185,700	36,875,184
ibm18	210,341	272	201,920	30,189,400	33,547,008

As outputs, Z-Place produces the following:

- Width and Height of every tier (if original netlist was not mapped to 3D);
- I/O pins position (if original netlist was not mapped to 3D);
- (X, Y, Z) coordinate of every cell placed;

Z-Place starts with an appropriate tier area planning and I/O pins in the boundary, as detailed in section 4.4. Since Quadratic Placement algorithm is used, this is necessary in order to compute a placement solution in 3D. The second step is the global placement, detailed in section 4.5, whose objective is to obtain approximate coordinates for all cells. Overlaps are allowed at this point, but this step searches for a reasonable spread of the clusters in the 3D space. The nets that will need 3D-Vias are decided at this point in a dynamic process that checks for availability of the resource as it trades-off 3D-Vias for wire length. A timing driven capability is proposed to avoid critical nets to use 3D-Vias.

The third step is called detailed placement (detailed in section 4.8); it legalizes the placement and improves the solution by local changes. For this step an algorithm based on the Threshold Accept heuristic (DUECK; SCHEUER, 1990) is used.

Finally, the last step is to place the 3D-Vias into legal places, that is detailed in section 4.9. The whole placement flow is illustrated in figure A.3.

4.4 I/O Pins Handling

It is assumed that the boundary of a random logic block, in 2D, is delimited by I/O pins and that the I/Os can be moved to any tier. I/O pins play two important roles in the placement of a block: first, I/Os limit the area boundary of the block; second, the

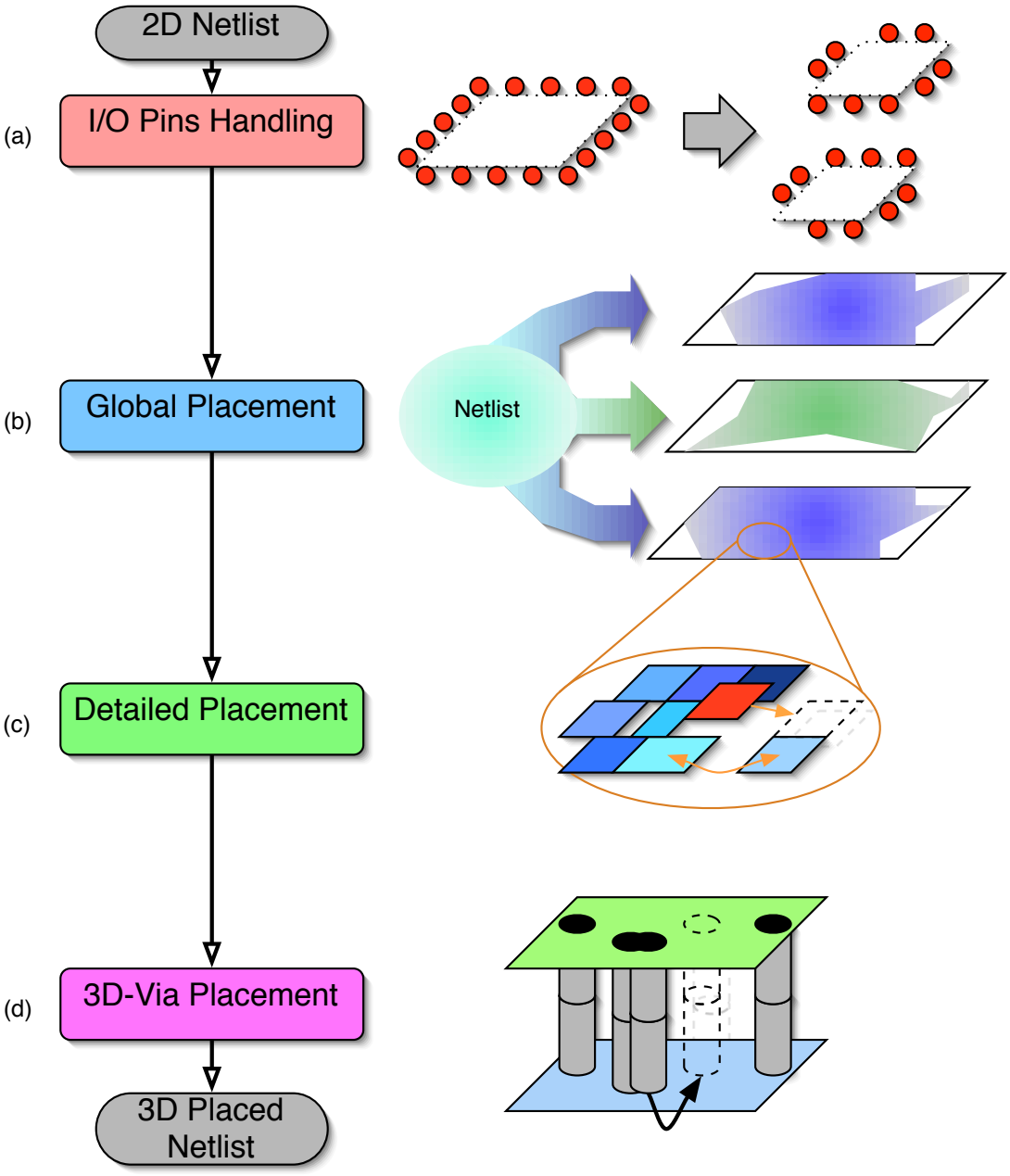


Figure 4.1: Proposed Placement flow for 3D Circuits

Table 4.2: Benchmark information obtained from real circuits VHDLs.

Bench	# Cells	# Terminals	# Nets	CircuitArea
b03	133	10	161	1,476
b07	396	11	428	3,477
b08	161	15	187	1,638
b09	163	4	185	1,677
b10	184	19	209	1,560
b11	604	15	641	5,256
b12	925	13	981	8,280
b13	275	22	325	2,703
b14	3,528	88	3,750	24,804
b15	7,639	108	8,058	62,001
b17	22,958	136	24,275	191,400
b18	59,552	61	62,277	476,100
b20	7,701	56	8,089	56,880
b21	7,721	56	8,112	56,643
b22	11,717	56	12,330	86,136
b14-1	3,641	88	3,829	25,758
b15-1	7,620	108	8,039	61,746
b17-1	22,921	136	24,238	190,965
b18-1	59,972	61	62,572	469,908
b20-1	7,929	56	8,320	58,800
b21-1	7,824	56	8,218	58,080
b22-1	12,164	56	12,775	90,000

pins are used as tips for many placement algorithms to reduce wire lengths. Consider the Quadratic Placement algorithm (ALPERT, C. et al, 1997), that is used by the leading industry and most of the existing academic cell placers. It requires I/Os at the boundary in order to compute a solution.

This section studies the I/O pins partitioning and placement problem. As already shown in figure A.3.(a), Z-Place distributes the I/O pins through all the boundary of the block. Summarizing the motivation, the goal is to find a good partitioning method for the I/Os that is able to maintain a good I/O pins balance leading to area balance between the tiers. At the same time, we indirectly address the reduction of 3D-Vias.

To the author's knowledge, the algorithm presented in the section is the first proposed approach to partition and place the I/O pins of a block into 3D. Previous works based on 3D Quadratic Placement probably needed a similar approach, but we did not find out any published method in the literature. It is assumed that simplistic solutions are being adopted.

This chapter evaluates the impact of different approaches for the I/O partitioning and propose an algorithm that is based on the logic distance of the I/Os as partitioning criterion. Summarizing the motivation, we want to find a good partitioning method for the I/Os that is able to maintain a good I/O pins balancing leading to area balance between the tiers. At the same time, we indirectly address the minimization of 3D-Vias and better wirelength.

4.4.1 Problem Definition

Given a 2D placement netlist with pre-placed I/O pins at the boundary of the region available for cell placement, the migration to a 3D netlist (ready for 3D placement) has the following goals:

- Area allocation: the width and height of the tiers must be calculated according to the number of tiers.
- I/O partitioning: the I/Os must be partitioned into different tiers.
- I/O placement: the I/Os must be placed at the boundary of the block, delimiting the area for cell placement.

We understand that the I/O partitioning problem should not perform the cells partitioning and that this is a task of the cell placement. Figure A.4 illustrates the I/O pins migration. As formulated in the next section, the netlist migration preserves some properties of the 2D solution, such as whitespace, aspect ratio, I/O pins orientation and ordering. Our objective is to provide a migration algorithm that facilitates the 3D-Via minimization. From the perspective of the I/O pin partitioning our idea is to provide a good starting point for the cell partitioning. The algorithm should provide good I/O pins balance and respect the mentioned properties.

In order to study the effect of our partitioning to the 3D-Via count, we follow the methodology presented in (ABABEI; MOGAL; BAZARGAN, 2005) that performs min-cut partitioning for the cells and tier assignment with Simulated Annealing after the I/Os are fixed in 3D. In our case, though, the min-cut have initially pre-placed fixed pins (I/Os). In this thesis, we propose to study the impact of the 3D-Vias in the tier area.

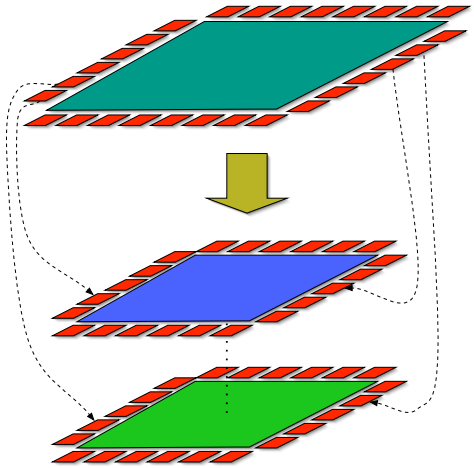


Figure 4.2: Migration (from 2D to 3D) of a netlist with pre-placed I/O Pins

4.4.1.1 Formal Definition

Before placement, a 2D circuit netlist Nl is composed by a set of gates $G = \{g_1, g_2, g_3, , g_n\}$, a set of I/O pins $P = \{p_1, p_2, p_3, , p_m\}$ and a set of nets connecting them $N = \{n_1, n_2, n_3, , n_o\}$. A hypergraph *Hg* represents the netlist, where $G \bigcup P$ is

the set of nodes and *N* is the set of hyperedges. The fixed position of each I/O pin p_i is given by $X[i]$ and $Y[i]$ ($i \leq m$) and its orientation by $Or(p_i)\epsilon\{north, south, east, west\}$. The area *A* (height *H* and width *W* having its bottom left corner at coordinate (x_{ini} ,y_{ini}) position) inside the I/O pins is assigned for cell placement. The whitespace ratio *S* on the placement area is achieved by subtracting the total gate area (*Ga*) from the area available inside the I/Os and dividing the result by *Ga*. The aspect ratio *Ar* is computed by *W* divided by *H*.

Let *Z* be the set of tier numbers $\{1, 2, ..., z\}$. The problem to be solved is defined as follows: given a 2D placement netlist *Nl* with fixed I/O pins, find a set of tiers $T = \{t_1, t_2, , t_z\}$ (*z* is the number of tiers) and their correspondent A_i, Ar_i, Ga_i, W_i, H_i, P_i, S_i, Or_i, X_i and Y_i ($i \leq z$) such that equations 4.1-4.8 hold.

$$P_1 \cup P_2 \cup ... \cup P_z = P \tag{4.1}$$

$$(\forall a, b\epsilon Z)(a \neq b \rightarrow P_a \bigcap P_b = \emptyset) \tag{4.2}$$

$$(\forall i\epsilon Z)(Wh_i \approx Wh) \tag{4.3}$$

$$(\forall i\epsilon Z)(Ar_i \approx Ar) \tag{4.4}$$

$$(\forall i\epsilon Z)(\forall j\epsilon Z)(W_i = W_j \wedge H_i = H_j) \tag{4.5}$$

$$(\forall i\epsilon Z)(\forall a\epsilon P_i)(Or_i(a) = Or(a)) \tag{4.6}$$

$$(\forall i\epsilon Z)(\forall a\epsilon P_i)(\forall b\epsilon P_i)(Or(a) = Or(b) \wedge X_i[a] < X_i[b] \rightarrow X[a] < X[b]) \tag{4.7}$$

$$(\forall i\epsilon Z)(\forall a\epsilon P_i)(\forall b\epsilon P_i)(Or(a) = Or(b) \wedge Y_i[a] < Y_i[b] \rightarrow Y[a] < Y[b]) \tag{4.8}$$

In other words, each tier will have its own set of I/O pins and no tier will share an I/O; the whitespace and aspect ratio must be evenly allocated; the orientation and ordering of the pins must be preserved.

4.4.2 Proposed algorithm

Let $Ld(p_i, p_j)$ be the length of the shortest path in *Hg* from p_i to p_j (e.g. the logic distance between p_i and p_j). The algorithm for I/O partitioning is described as follows.

Algorithm 1 I/O Pins Partitioning and Placement algorithm

1: Compute $Ld(i, j)\forall i, j\epsilon P$
2: Create a complete graph Pg such that P is the set of nodes and $Ld(i, j)(i, j\epsilon P)$ is the cost of the edge connecting nodes i and j.
3: Perform the partitioning of Pg into $P_1, P_2, , P_z$ configured to perform min-cut optimization at a 1% maximum unbalance ratio.
4: Compute Ga_i ($i\epsilon Z$) by Ga divided by z
5: Compute A_i by adding Wh_i to Ga_i
6: Compute the dimensions of the tiers based on equation 4.9.
7: Place the I/O pins around the boundary of the block by simple stretching according to equation 4.10.
8: Legalize I/O Positions

$$W_i = \sqrt{A_i} \times Ar_i \tag{4.9}$$

$$H_i = \frac{\sqrt{A_i}}{Ar_i}$$

$$(\forall i \epsilon z)(\forall p \epsilon P_i) X_i[p] = \frac{(X[p] - x_{ini}) \times W_i}{W} \tag{4.10}$$
$$(\forall i \epsilon z)(\forall p \epsilon P_i) Y_i[p] = \frac{(Y[p] - y_{ini}) \times H_i}{H}$$

The first step of the algorithm is illustrated in A.5.(a). Considering that in a real circuit net fanouts are limited, node degrees can be considered bounded or constant for the sake of complexity analysis. Thus, a single BFS search has an $O(n)$ complexity. The algorithm can be performed by m^2 BFS searches in Hg resulting in a $O(m^2 n)$ time complexity. Since the number of I/O pins do not exceed a few thousand, it is feasible to use BFS. By using a single search to compute the distance from a pin p_i to every $p \epsilon P$, the complexity can go down to $O(mn)$.

On step2, the values of Ld are used to create a Pg graph connecting all pairs of I/O pins, as shown in figure A.5.(b).

For the third step, we used the hMetis tool (KARYPIS, G. et al, 1999). The tool accepts edge weights in the input which modifies they way they are accounted in the cut computing. Basically a edge with 2 as weight would be considered twice as much as a net with 1 weight. We assigned the inverse of the edge costs as their weights in order to penalize more the long movements and try to keep shorter paths toghether. A very tight I/O balance is imposed in order to keep a similar amount of I/Os in each tier. In section 4.4.4.3 the effects of unbalancing the I/O pins are discussed.

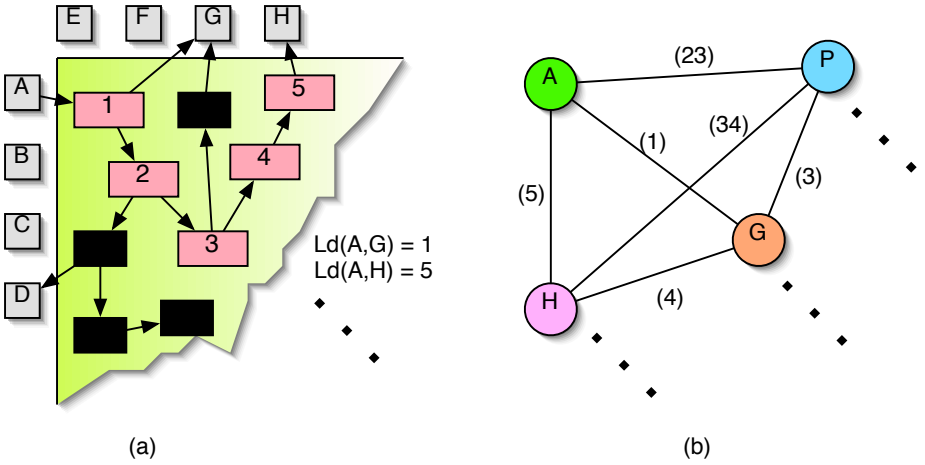


Figure 4.3: An ilustration of the logic distance between I/O pins (a) and a part of the correspondent complete graph (b)

The forth step can be accomplished by a simple division of the total gate area by the number of tiers. So far, it is not possible to know whether such perfect cells partitioning will be achievable, but it is a reasonable assumption. Nevertheless, S_i could be changed to compensate the Ga_i inaccuracy.

The steps 5 and 6 compute the area of the tiers such that aspect ratio and whitespace are preserved from the original 2D circuit. At this point, new aspect ratio or whitespace could be used.

Finally, the steps 7 and 8 compute the x and y coordinates of the I/Os to their target tiers. The original orientation and ordering is preserved, since the I/O placement is a mapping from their original position into a smaller area. A legalization (step 9) is performed at the end to assure that the I/Os do not overlap.

4.4.3 Experimental Setup

The goal is to study the impact of the I/O pin partitioning in the area, number of vias and I/O pin balance. For that, we defined a simplistic 3D placement flow as follows:

1. Initially the I/O partitioning algorithm under study is performed.
2. A min-cut partitioning of Hg into z partitions is performed. The I/O pins, that have already an assigned partition, are used as fixed nodes. The hMetis tool is applied for this step. The tool is configured to keep the area as balanced as possible (maximum 1% unbalance).
3. A tier assignment (similar to the one from (ABABEI; MOGAL; BAZARGAN, 2005)) problem maps the sets $P_1, P_2, ..., P_z$ into tiers $t_1, t_2, ..., t_n$. A Simulated Annealing engine is used (see figure 4.4).
4. Cells could be placed separately in each tier. We skip this step since our goal at this point is to evaluate the number of 3D-Vias.

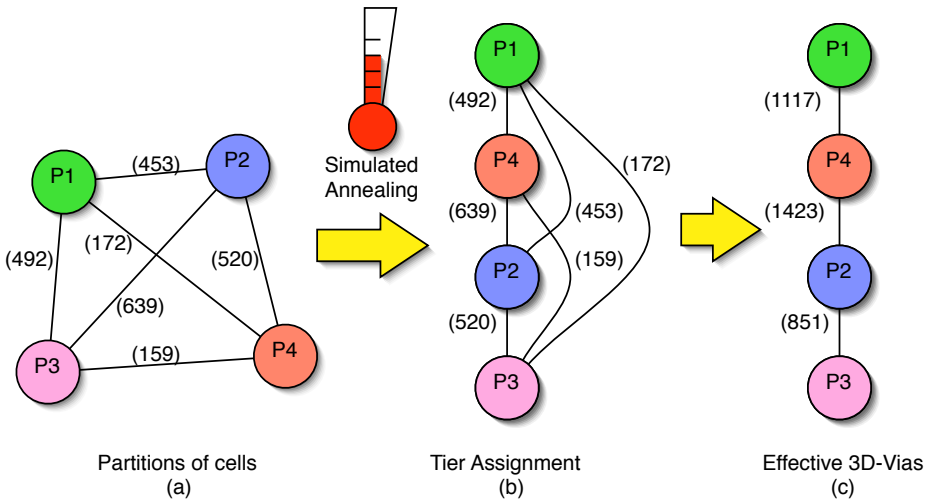


Figure 4.4: A group of partitions (a) are assigned to tiers (b) using Simulated Annealing; the effective number of 3D-Vias is shown in (c)

As there is no published previous work on I/O pins handling, the proposed I/O partitioning algorithm is compared with two other simplistic algorithms that follow the same formulation described in section 4.4.1.1. The first algorithm is called *AlternatePins*, on figure 4.5.(a). This method is a pseudo-random partitioning that goes thought the boundary line of the chip picking nodes for each partition alternatively. The *AlternatePins* replaces steps 1,2 and 3 of the flow keeping steps 4,5,6,7 and 8 untouched in order to maintain the same I/O placement policy.

The idea behind the *AlternatePins* method is to provide an optimal solution in terms of balancing the I/Os. Balancing is important for the subsequent placement stage because the I/Os play a very important role in the quadratic placement engine (ALPERT, C. et al, 1997). This algorithm computes an optimal solution for the cell placement based on attraction forces between connected cells. I/O pins, placed at the boundary, are responsible for the spreading of the cells, since otherwise they would be placed at the center point.

The second method is called *UnlockedPins*, illustrated in figure 4.5.(c). In this method, we allow hMetis to partition the I/Os as free nodes, replacing the steps 1,2 and 3 of our algorithm. The following steps of our algorithm are done for the *UnlockedPins* as well.

The idea behind the *UnlockedPins* method is to provide a favorable solution in terms of 3D-Via minimization. Since hMetis is a leading edge hyper-graph partitioner, it will generate a netlist partitioning with close to optimal number of 3D-Vias. On the other hand, I/O pins will not be spread evenly.

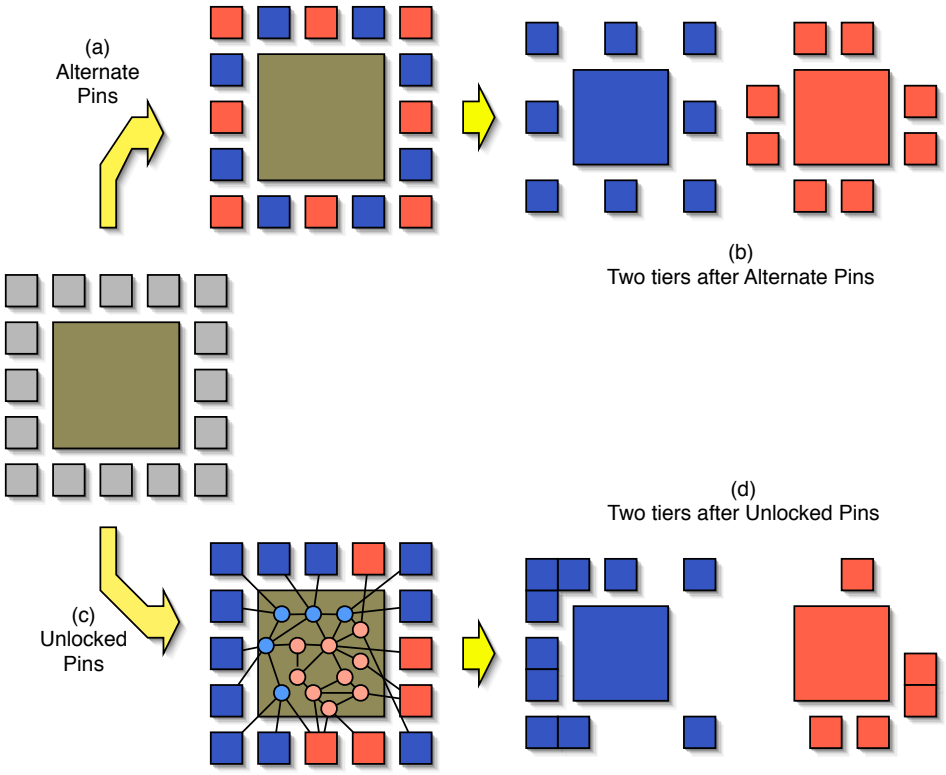


Figure 4.5: An illustration of the Alternate Pins algorithm (a) resulting in a two tier circuit (b) with perfectly balance I/O pins; the Unlocked Pins algorithm (b) uses hMetis to partition the whole Netlist, which could result in unbalanced pins (d).

The method proposed here aims at a good solution in terms of both 3D-Vias and balancing. Section (4.4.4) presents experimental results comparing the algorithm under these metrics.

4.4.4 Experimental Results

4.4.4.1 Effect on 3D-Vias

Experiments measuring the amount of 3D-Vias and the balancing of the algorithm are presented in this section. Tables 4.3, 4.4 and 4.5 report our experimental results. ISPD 2004 benchmarks (ISPD04 - IBM STANDARD CELL BENCKMARKS WITH PADS., 2004) are used targeting circuits with two, three, four and five tiers.

First, table 4.3 reports the I/O balancing measured by the standard deviation of the number of I/O pins averaged from the whole IBM benchmark suite. The average number of I/O pins from the IBM benchmarks is 264. The method *AlternatePins* delivers the optimal solution while *UnlockedPins* is very unbalanced. In some situations, the strong unbalance practically invalidates the method. The proposed algorithm has close to optimal pin balancing.

Table 4.3: Comparison of the I/O pins distribution in the tiers considering the three studied algorithms averaged from ibm01 to 1bm18.

# tiers	Algorithm	σ # I/Os
2	Our Algorithm	7
	UnlockedPins	233
	AlternatePins	0.4
3	Our Algorithm	6
	UnlockedPins	252
	AlternatePins	0.4
4	Our Algorithm	5
	UnlockedPins	177
	AlternatePins	0.4
5	Our Algorithm	6
	UnlockedPins	189
	AlternatePins	0.4

Tables 4.4 and 4.5 presents our experimental results for the total number of 3D-Vias for the whole IBM benchmark suite. The *AlternatePins* method has the worst results under this metric, which is expected since it is a pseudo-random partitioning. This fact enforces the conclusion that a simplistic I/O partitioning leads to a worse cut size. On the other hand, the method *UnlockedPins*, which was expected to have the best cut among the three methods was outperformed by our algorithm. This fact can be explained by our pre-processing stage that computes the logic distance between I/Os. It seems that the logic distance is a way to summarize the information of the whole graph into a single edge that connects I/O pins (step 2 of the algorithm). Since the graph into this step is very small compared to the whole netlist hyper-graph, the partitioning algorithm (hMetis in this case) could achieve a good partitioning for the pins and for the netlist as well. This computation requires intensive CPU usage. To overcome this problem, the distances are pre-computed and stored in a file so that the I/O partitioning runtimes are not harmed.

Tables 4.6 and 4.7 present experimental results for the maximum number of 3D-Vias between pairs of tiers.

Table 4.4: Total number of 3D vias for the proposed algorithm.

	Our Algorithm # 3D-Vias			
# tiers	2	3	4	5
ibm01	374	525	837	1162
ibm02	396	747	1156	1533
ibm03	1064	2174	2610	3974
ibm04	735	1511	2371	2852
ibm05	2258	4311	6489	9193
ibm06	1059	1642	2934	3477
ibm07	992	2050	3219	4400
ibm08	1298	2697	4018	5346
ibm09	699	1872	2495	3343
ibm10	1490	2661	4004	5216
ibm11	1190	2240	3685	4620
ibm12	2293	4094	6581	8191
ibm13	1042	1893	3099	3742
ibm14	2121	3886	5342	6667
ibm15	3002	4827	7022	9283
ibm16	2102	4316	5774	7172
ibm17	2769	5611	8526	10114
ibm18	1676	3591	4985	6581
Average	1476	2814	4175	5381

4.4.4.2 Studying the area effect of 3D-Vias

Table 4.8 presents an area impact study of the 3D-Vias considering the three algorithms (the numbers are averaged for all benchmarks). The column “Max # 3D-Vias" reports the maximum number of 3D-Vias connecting pairs of adjacent tiers; this data is extracted from tables 4.6 and 4.7. This number will impact the area requirements for 3D-Vias. The area study supposes 3D-Vias measuring $5\mu m$ and $50\mu m$, which represent a good 3D-Via pitch and a huge 3D-Via pitch respectively.

The following facts can be observed on table 4.8:

- The *big* 3D-Vias, that could be bulk based face-to-back vias, suffer from a very high penalty for the 3D-Vias. With 2 tiers, there is a penalty of around 53% of the tier area (note that our algorithm results in less 3D-Vias and also less tier area than the others). For the cases with 4 and 5 tiers, the 3D-Via area is larger than the tier area. The important conclusion here is that when targeting a big via technology it is mandatory to minimize the number of 3D-Vias in order to obtain a feasible solution. As seen in previous tables (4.6 and 4.7) the proposed algorithm can save up to 34% which translates to area savings in the order of an entire tier.

- Technologies with small vias suffers from around 2% of area penalty for the 3D-Vias, leaving room for more 3D-Vias if they are helpful.

4.4.4.3 Unbalancing the I/O pins

In the previous section we could observe that there is a trade-off between the I/O pins balance and the resulting number of 3D-Vias. The proposed algorithm for pin partitioning

Table 4.5: Comparison of the total number of 3D vias for the three studied algorithms for I/O pin partitioning over the others.

	UnlockedPins # 3D-Vias				AlternatePins # 3D-Vias			
# tiers	2	3	4	5	2	3	4	5
ibm01	441	857	838	1439	428	881	977	1372
ibm02	547	882	1214	1600	503	829	1340	1691
ibm03	1146	2282	2693	4020	1099	2530	3602	4366
ibm04	628	1583	2516	3202	750	1619	2461	4275
ibm05	2417	5372	6653	9651	2576	5428	7037	12400
ibm06	1057	1827	3128	3566	1075	1729	3429	3507
ibm07	880	3242	3302	4605	1049	3423	3482	6523
ibm08	1324	2814	4184	5698	1307	3431	4183	6327
ibm09	806	2828	2763	3518	780	2186	3757	3556
ibm10	1771	3565	4675	7116	1821	4062	4358	8492
ibm11	1490	3477	3958	5697	1494	3629	4923	7437
ibm12	2594	5350	7259	9158	2556	5569	8996	12515
ibm13	1193	3037	3264	4557	1170	2912	4618	4874
ibm14	2171	4561	6584	8085	2310	5090	7564	10113
ibm15	2890	7863	9082	11707	3126	7970	11144	13857
ibm16	2237	5816	6235	9300	2280	6216	9525	10903
ibm17	2539	7695	8733	10845	2847	8402	11420	14080
ibm18	1835	4686	5229	9072	1704	3899	5268	8193
Average	1554	3763	4573	6269	1604	3879	5449	7466
Our Improv.	5.29%	33.74%	9.53%	16.49%	8.72%	37.84%	30.52%	38.73%

aims at good balance. However, it is well known that a tight balance requirement over-constraints the partitioning process (KARYPIS, G. et al, 1999). In the proposed algorithm, the I/O balance can be controlled in step 3 that is performed by hMetis.

HMetis allows the user to configure the balance constraint for each bisection based on equation 4.11 where u is the unbalance parameter and n is the number of vertices on the hyper-graph.

$$\left[\frac{(50-u)\times n}{100}; \frac{(50+u)\times n}{100}\right] \tag{4.11}$$

For example, let $u = 10$, then the bisection balance will range from 40%-60% to 60%-40%. Now suppose that we have four partitions, then an unbalancing factor 10 will result in partitions that can contain between $0.402 \times n = 0.15 \times n$ and $0.602 \times n = 0.35 \times n$ vertices.

Our experimental results (averaged from all benchmark circuits) are reported on table 4.9 and figure 4.6. Table 4.9 presents the I/O pin unbalance measured by Standard Deviation. Figure 4.6 presents the benefits of unbalancing the I/Os to the 3D-Via count.

Table 4.6: Maximum number of 3D-Vias for proposed algorithm.

# tiers	2	3	4	5
ibm01	374	330	370	400
ibm02	396	413	403	594
ibm03	1064	1112	1088	1260
ibm04	735	887	992	887
ibm05	2258	2203	2469	2729
ibm06	1059	849	1135	948
ibm07	992	1332	1433	1524
ibm08	1298	1448	1397	1610
ibm09	699	1057	1008	1075
ibm10	1490	1450	1590	1750
ibm11	1190	1485	1605	1719
ibm12	2293	2278	2422	3173
ibm13	1042	1269	1548	1781
ibm14	2121	2272	2248	2459
ibm15	3002	2857	3199	3395
ibm16	2102	2164	2212	2625
ibm17	2769	3150	3601	3105
ibm18	1676	1871	1754	1782
Average	1476	1579	1693	1823

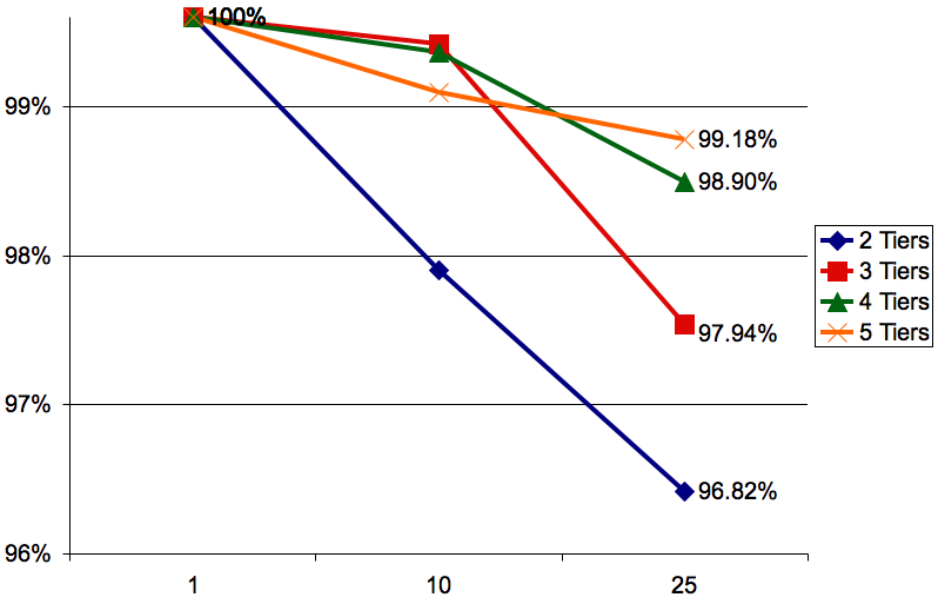


Figure 4.6: The percentage improvement on 3D-Via count of unbalancing the I/O pins.

4.4.5 Partial Conclusions

A method for the partitioning and placement of the I/O pins of a 2D block to a 3D circuit was proposed. An interesting analysis is that our method lies in the fact that it actually improved the hypergraph partitioning algorithm cut by performing only shortest

Table 4.7: Comparison of the maximum number of 3D vias for the three studied algorithms for I/O pin partitioning over the others.

	UnlockedPins Max # 3D-Vias				AlternatePins Max # 3D-Vias			
# tiers	2	3	4	5	2	3	4	5
ibm01	441	467	377	573	428	483	406	480
ibm02	547	496	485	552	503	469	498	553
ibm03	1146	1143	1021	1334	1099	1320	1485	1210
ibm04	628	862	1067	1039	750	913	1033	1454
ibm05	2417	2765	2478	2712	2576	2814	2526	3974
ibm06	1057	924	1134	935	1075	915	1193	937
ibm07	880	1980	1510	1525	1049	2050	1590	2402
ibm08	1324	1436	1445	1788	1307	1919	1448	1833
ibm09	806	1598	1092	1249	780	1356	1684	1137
ibm10	1771	1883	1741	1986	1821	2247	1724	2898
ibm11	1490	1909	1810	2230	1494	1856	1802	2610
ibm12	2594	2820	2747	2962	2556	3160	3113	4205
ibm13	1193	1606	1500	1755	1170	1611	1905	1954
ibm14	2171	2375	2307	2881	2310	2619	3274	3283
ibm15	2890	4188	3377	4099	3126	4207	4385	4163
ibm16	2237	3185	2266	3355	2280	3704	3794	3443
ibm17	2539	4165	3526	2990	2847	4539	5245	5053
ibm18	1835	2652	1852	2810	1704	2127	1856	2552
Average	1554	2025	1763	2043	1604	2128	2165	2452
Our Improv.	5.29%	28.24%	4.14%	12.06%	8.72%	34.76%	27.85%	34.51%

Table 4.8: Comparison of the 3D-Vias Area Impact Considering the Three Algorithms.

# tiers	Algorithm	Area Tier	Max # 3D-Vias	Area 3D-Vias (big - 50μm)		Area 3D-Vias (small - 5μm)	
2	OurAlgorithm	6,934,347	1,476	3,690,000	53%	36,900	1%
3		4,660,116	1,579	3,947,500	85%	39,475	1%
4		3,490,471	1693	4,232,500	121%	42,325	1%
5		2,821,087	1823	4,557,500	162%	45,575	2%
2	UnlockedPins	6,936,553	1,554	3,885,000	56%	38,850	1%
3		4,658,909	2,025	5,062,500	109%	50,625	1%
4		3,481,276	1,763	4,407,500	127%	44,075	1%
5		2,817,413	2,043	5,107,500	181%	51,075	2%
2	AlternatePins	6,926,117	1,604	4,010,000	58%	40,100	1%
3		4,640,572	2,128	5,320,000	115%	53,200	1%
4		3,489,458	2,165	5,412,500	155%	54,125	2%
5		2,816,187	2,452	6,130,00	218%	61,300	2%

path analysis. Note that the method works in two phases: first the I/O partitioning considering the logic distances as weights; second, fix the I/Os and perform partitioning of the cells. In the first phase, the I/Os are arranged in a small graph (containing only the I/Os) weighted by the logic distance on the original graph. The edge weights actually contain

Table 4.9: The Unbalance of the I/O pins measured by the Standard Deviation

	2 tiers	3 tiers	4 tiers	5 tiers
u=1	7	6	5	6
u=10	64	54	41	48
u=25	158	141	100	103

information of the whole netlist, compressed in the small I/O graph. In the second phase, the whole netlist is partitioned, however some nodes (the I/Os) are fixed, reducing the problem complexity and more importantly providing tips to the partitioning algorithm. We conclude that the reduced problem sizes with compressed information of the whole netlist actually improved the partitioning algorithm at the expense of more CPU time.

Empirically, we showed that doing the partitioning of I/O together with the cells (UnlockedPins method) leads to strongly unbalanced number of pins, which invalidates the method. We also demonstrated the pseudo-random I/O partitioning approaches (such as AlternatePins) leads to a higher number of 3D-Vias. The proposed method demonstrated good effectiveness both in terms of I/O balance and resultant number of 3D-Vias (5% to 33% improvement on 3D-Via count compared to hMetis), outperforming both algorithms in both metrics.

After that, the area impact was studied under our simplified placement flow that minimizes the number of 3D-Vias. It was verified that the area overhead caused by 3D-Vias is prohibitively high for big (50μm pitch) 3D-Vias (in the order of 50% of the active area and up), requiring more research on via minimization methods. On the other hand, for small (5μm pitch) 3D-Vias, the impact was small (around 2% of the active area), leaving room for additional 3D-Vias if it can improve circuit performance. Any intermediary case would be able to trade 3D-Vias for performance limited by the area occupied by the 3D-Vias.

Finally, we investigated ways to further minimize the cut by working with the I/O pin balancing. We relaxed the I/O pin balance constraint keeping the area evenly distributed since the second partitioning process is still highly constrained. Adding up the advantage reported in previous works with the improvements achieved on this thesis, we can outperform hMetis partitioning from 5.5% to 34% in average.

4.5 Global Placement

The global placement takes place after the I/Os are distributed along the tiers and properly placed at the boundary of the cells area. The task of the global placement is to propose a placement of all cells within one of the available tiers. We understand as *valid* a z coordinate that fits exactly one of the tiers. It is also a task of the global placement to deliver a valid solution with respect to constraints for addressing 3D-Via related issues.

Fastplace, proposed in (VISWANATHAN; PAN; CHU, 2005), is a fast analytical placer based upon quadratic placement approach. Z-Place incorporates some algorithms of Fastplace and extend them to 3D. For this reason, the Fastplace method is reviewed in section 4.5.1.

The remaining of the global placement section is organized as follows: Sections 4.5.5 and 4.5.6 present the basic formulations of the 3D placement problem in Z-Place. Section 4.5.7 presents the flow of the proposed global placement method. The new features of our 3D placer as well as the algorithms are presented in sections 4.6.1 and 4.6.2. Those

features enable the wire length optimization on 3D and provide means to spread the cells while migrating them to a circuit tier at the same time. A dynamic area allocation for the 3D-Vias is performed while keeping 3D-Via count bellow an upper bound that is sensible to the integration strategy and 3D-Via area.

4.5.1 Fastplace Review

Fastplace (VISWANATHAN; PAN; CHU, 2005) solves a linear system of equations, which models the cells connectivity. After the system is solved, the Cell Shifting technique is applied followed by an Add Spreading Force step. These three steps are iterated until a roughly even placement is achieved. Finally, Fastplace applies an Iterative Refinement step that improves linear wire length and cell spreading at the same time.

4.5.2 Cell Shifting

The solution of the system concentrates the cells in the middle of circuit area and therefore has a high amount of overlap. To spread cells out and remove overlap, Fastplace utilizes the Cell Shifting technique, which acts as follows. Initially the circuit area is divided in regular (equal sized) bins. The size of bin is such that it accommodates in average 4 cells. The utilization of a bin is computed by accumulating the overlap area between the cells and their bin. Based on regular bin structure, an irregular bin structure is constructed (figure 4.7) in such a way that bins with high utilization will have an increased size while lower utilized ones will have decreased size.

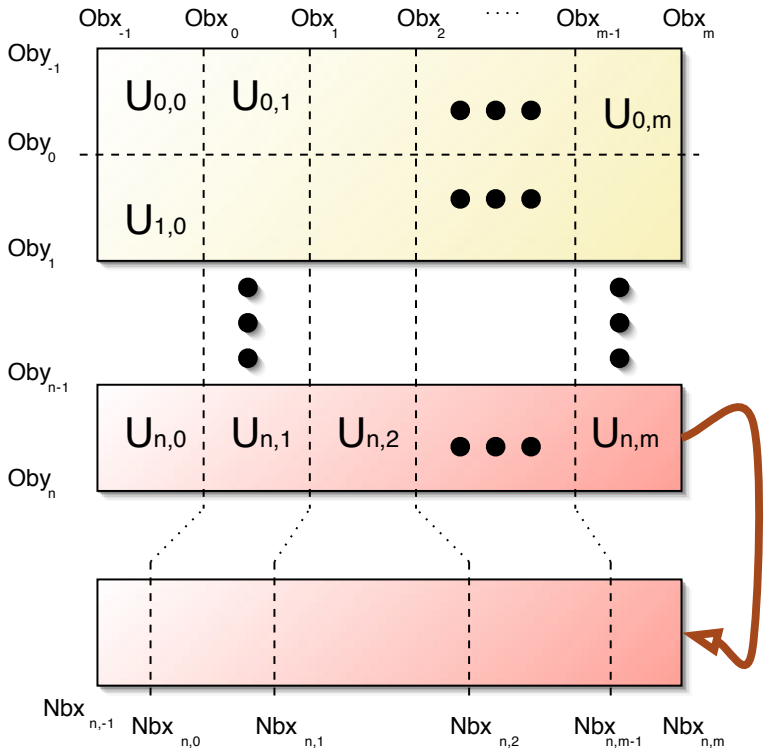


Figure 4.7: The regular bin structure used in the cell shifting method.

The new boundaries, Nbx, of a bin i is calculated by equation 4.12 where:

- Obx is the boundary coordinate corresponding to the regular bin structure;
- U is the bin utilization;
- i is the index of the current bin;
- $i+1$ is the index of the next (right) bin and $i-1$ is the index of the previous (left) bin;
- the δ is used to avoid cross-over between bin boundaries where utilization is zero.
- In y dimensions, analogous computation is performed.

$$Nbx_i = \frac{Obx_{i-1}(U_{i+1}+\delta)+Obx_{i+1(U_i+zdelta)}}{U_i+U_{i+1}+2\delta} \tag{4.12}$$

After the irregular bin structure was built, the cells are linearly mapped from equal sized bins to the correspondent irregular bin. The equation 4.13 maps a cell from its original position, x_{old}, to its target position, x_{new}, after shifting.

$$x_{new} = \frac{Nbx_{r,i}\times(x_{old}-Obxi-1)+Nbx_{r,i-1}\times(Obx_i-x_{old})}{Obx_i-Obx_(i-1)} \tag{4.13}$$

Instead of moving the cell directly to its assigned target position, (VISWANATHAN; PAN; CHU, 2005) suggests a movement to an intermediate position. Since the computation of the target position is based only on utilization, a long movement could be harmful to wire length. For this reason, a movement control parameters α_x and α_y are used for each axis defined as a real number between 0 and 1. For instance, if α is set to one, then the whole movement is performed and if α is 0.5, then the cell is moved half way. The equation 4.14 and 4.15 presents how α_x and α_y are computed respectively.

$$\alpha_x = 0.02 + \frac{0.5}{maxU} \times \frac{averageCellWidth}{cellHeight} \tag{4.14}$$

$$\alpha_y = 0.02 + \frac{0.5}{maxU} \tag{4.15}$$

The value of α is inversely proportional to the current maximum utilization of the circuit ($maxU$). This way, cells are shifted over very small distances during the initial placement iteration when the maximum utilization is higher. During the later iterations, α reaches high values.

4.5.3 Add Spreading Forces

After the cells have been shifted, new forces (one for each cell) are added to the system in order to avoid cells collapse back to their old positions in the next system resolution. This is achieved by connecting by a spring each cell to a pseudo-pin placed at the boundary of the placement region, as illustrated by figure 4.8.

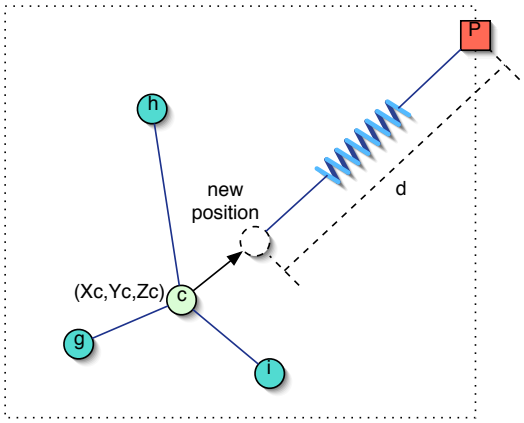


Figure 4.8: A cell movement and the force created represented by a spring.

The force vector $F = (F_x, F_y)$ applied under cell c is added in the springs system with the addition of an artificial pin P_c and an artificial net N_c that connects c to P_c. The pin P_c is placed at the boundary of the placement space (as suggested by (VISWANATHAN; PAN; CHU, 2005)). The force F is obtained based on the distance of the cell c's new position $(newx_c, newy_c)$ to all cells and I/O pins connected to it, except the distance d to the just added artificial pin. The equation 4.16 describes the computation of the x component of vector F, where $Connected_c$ is a set that contains the elements i (cells or I/O pins) connected to c in the netlist and $w_{c,i}$ is the weight of the connection from c to i that can be extracted from matrix Q. The other component for F (F_y) is calculated analogously.

$$F_x = \sum_{i \in Connected_c} (w_{c,i} \times (newx_c - X_i)) + (newx_c - X_c) \tag{4.16}$$

Finally, the weight of the connection from c to P_c, $weightN_c$, is obtained according to equation 4.17.

$$weightN_c = \frac{\|F\|}{d} = \frac{\sqrt{F_x^2 + F_y^2 + F_z^2}}{d} \tag{4.17}$$

4.5.4 Iterative Local Refinement

Since quadratic wirelength is just an indirect measure of half-perimeter, Fastplace uses a greedy technique to tries to minimize the half-perimeter while trying to reduce the maximum bin utilization. The Iterative Local Refinement step divides the circuit in bins in the same way as Cell Shifting but the bin size is 5x larger.

For each bin, all cells are tested to move to the four neighbor bins. Each movement is scored with respect to the wirelength and bin utilization. The best movement is taken unless none of the movements is able to improve the score. This process is repeated until there is no significant improvement in wirelength. At each iteration, the bin size is decreased until the cell shifting size is reached.

4.5.5 3D Integration Strategies Under Z-Place

A 3D Circuit is composed by the stacking of 2D VLSI circuits. The circuits, called tiers, are composed by an active area, metal layers and insulator layers. Depending on how the circuits are arranged, the integration between a pair of adjacent tiers can be classified as face-to-face, face-to-back or back-to-back. Davis et. al. (DAVIS, W. et al, 2005) describe the Microbump face-to-face technology in contrast with the Through Vias that are used in the face-to-back strategy. They also describe the technology from MIT labs (MITLL) that mixes the strategies for 3 tiers circuits (first tier is face-to-face to the second one, that is face-to-back to the third tier). Figure A.7 illustrates the integration strategies supported by Z-Place. It could support back-to-back as well with few adjustments. Note that the face-to-back Through Vias must dig a hole in the Bulk and for that they require active space that cannot be used by the circuit logic. On the other hand, a face-to-face 3D-Via has to go through all the metal layers of both tiers consuming more routing resources than face-to-back.

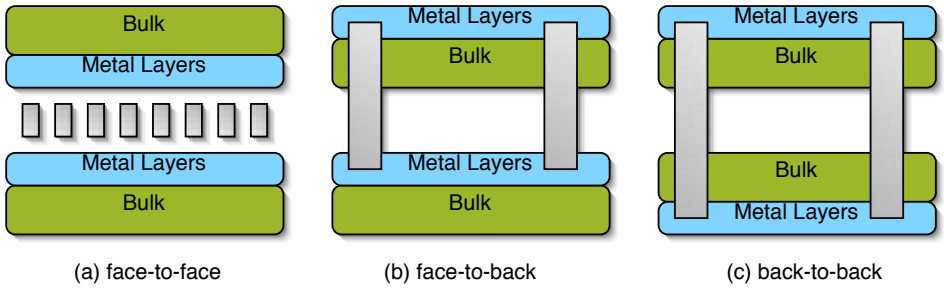


Figure 4.9: 3D Integration Strategies: (a) face-to-face, (b) face-to-back and (c) back-to-back.

From the cell placement problem perspective, we impose two constrains that are defined according to the strategy used:

- **Active area balance**. It is a task of the 3D placer to provide a balance of the cells area for each tier. Due to the space required by face-to-back 3D-Vias, our placer is constrained to leave more room in the tiers where 3D-Vias occupy active area.
- **3D-Vias Count**. It is reasonable to define an upper bound for 3D-Vias count. The upper bound is adapted to the 3D-Via technology. A certain 3D-Via area is allocated for each type of integration. The upper bound is obtained by dividing the allowed area to the area of a single 3D-Via under that type of integration.

4.5.6 Problem Formulation

Given a set of cells $C = \{c_1, c_2, c_3, ..., c_n\}$ and an Area function $CellArea(c) \in C \times \mathbb{N}$, the placement problem must compute a triple (x_c, y_c, z_c), for every cell c, that represents a *valid* placement within the placement cube minimizing 3D wire length that is informally defined as an extension of traditional 2D half perimeter adding the offset on the Z coordinate as well.

Let m be the number of tiers in a circuit. We define the placement space as a sliced cube, as illustrated in figure A.6 with m slices. Under this model, the cells can be temporarily placed in any z coordinate, but in the end of the process the cells must be in a

valid coordinate that is defined to exactly match a tier. In other words, *invalid* coordinates are defined as any intermediate coordinate between adjacent tiers.

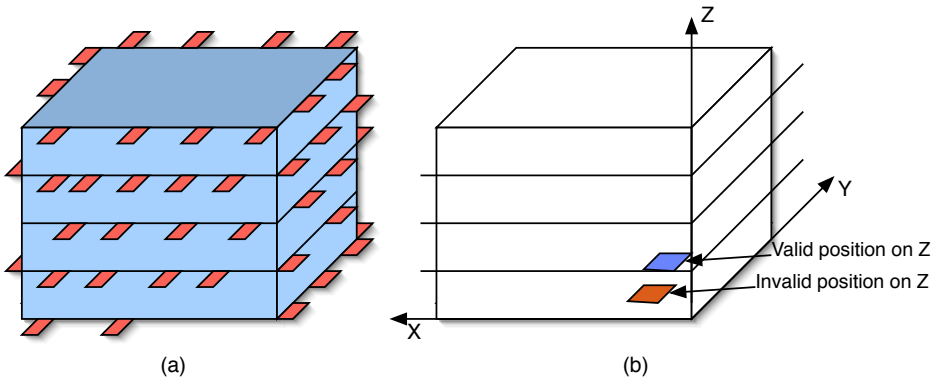


Figure 4.10: Sliced cube model with invalid coordinates and I/O in all tiers.

Every two adjacent tiers in the cube have an integration strategy that can be either face-to-face ($f2f$), face-to-back ($f2b$) or back-to-back ($b2b$). Let $I_{strategy}$ be the set $\{f2f, f2b, b2b\}$ of possible strategies. Let I be an array of $w = m - 1$ positions indexed by i ($0 \leq i < w$) that describes the integration between tier t_i and t_{i+1} as $f2f$, $f2b$ or $b2b$. Any two adjacent positions in I such as i and o ($i - o \in \{-1, 1\}$) cannot have the combination $I[i] = f2f$ and $I[o] = f2f$. In order to have two $f2f$ positions, a $b2b$ one must be between than. The following formal grammar (FORMAL GRAMMAR DEFINITION, 2007) formalizes all possible compositions of integration strategies, where the set of non-terminal symbols $N = \{\square, \Diamond\}$ the set of terminal symbols $\Sigma = \{< f2f >, < f2b >, < b2b >\}$, ϵ is the empty word and $\square$ is the start symbol.

1. $\square \rightarrow < f2f > \Diamond \mid < f2b > \square \mid \square < b2b > \square \mid \epsilon$
2. $\Diamond \rightarrow < f2b > \Diamond \mid \epsilon$

Under the sliced cube model, the space between two tiers is occupied by a 3D-Via layer. Let V be the array of 3D-Via layers with w positions. In a 3D-Via layer, the 3D-Vias must be placed legally (YAN, H. et al, 2005) and for that enough space must be provided by the cell placer. A 3D-Via have pitch requirements depending on the technology and integration strategy. We define a function $3D - Via_{pitch}(strategy) \mid stragegy \in I_{strategy}$ that returns the pitch requirement of a given integration strategy. A similar function $3D - Via_{height}(strategy)$ defines the distance (in the Z axis) between a pair of adjacent tiers integrated according to the variable $strategy$. In the Z axis, the size of the cube ($depth$) is given as by $depth = \sum_{0 \leq v < w}(3D - Via_{height}(I(v)))$. The position of a slice s ($0 \leq s \leq w$) on the cube can be located by $\sum_{0 \leq v < s}(3D - Via_{height}(I(v)))$.

Let J be the array of 3D-Vias counters indexed by j ($0 \leq j < w$). We constraint our algorithm to limit the number of 3D-Vias in a given pair of adjacent tiers according to a max 3D-Via count function $Ubound \in I_{strategy} \times \mathbb{N}$. Given the tier area ($tier_{area}$), a certain percentage of the area for each strategy ($P_{area}(strategy)$), the $UBound(strategy)$ is defined by $P_{area}(strategy) \times tier_{area} \times 3D - Via_{pitch}(strategy)$. The constraint added to the algorithm is given by equation 4.18.

$$\forall_{0<j<w}(J[i] < Ubound(I[j])) \tag{4.18}$$

Additionally we impose a constraint with respect to the area distribution. In (ABABEI, C. et al, 2005) the authors mention that leaving more whitespace in the upper tier would improve heat dissipation depending on the packaging technology. Under our model we understand that a good distribution allocates similar whitespace through the tiers. However, in the presence of face-to-back integration, some tiers must allocate extra space for the 3D-Vias. In such cases, we want that cell's area be redistributed maintaining the whitespace distribution even.

For the set C of cells, let the total cell's area in the circuit be Ca. Ca can be obtained by summing the area of each individual cell ($\sum_{c \in C} CellArea(c)$). Let:

- Ca_t be the cells area allocated for tier t;
- Va be the total 3D-Vias area in the circuit;
- Va_t be the 3D-Via area needed in tier t due to the integration of tier t with tier $t-1$ if $I(t-1) = f2b$ and $1 \leq t < z$.

We define an ideal area for the cells in a tier t as shown in equation 4.19. In our global placement we try to maintain the proposed area distribution by penalizing movements that produces a tier utilization that is larger the the ideal one.

$$IdealCa_t = \frac{Va+Ca}{z} - Va_t \tag{4.19}$$

4.5.7 Z-Place Global Placement Flow

In Z-Place, the 3D placement process that optimize purelly 3D wire length at the first iteration while it slowly starts to consider the constraints of the problem. By iterating a system of equation solving that optimizes wire length with a process that introduces weak forces that favors cells to move to a balanced position we do not affect much the wire length optimization by satisfing the constrains. The global placement flow is summarized in figure 4.11. Our algorithm is similar to the one in (VISWANATHAN; PAN; CHU, 2005) that was proposed for 2D circuits. For this reason, the method is reviewed in section 4.5.1. The main placement engine is analytic targeting at Quadratic wire length minimization. We apply 3D Cell Shifting (section 4.6.1) in order to spread the cells that are initially placed in invalid and overlapped coordinates concentrated in the center of the cube. It is a task of this stage to migrate the cells to a tier in order to obtain a *valid* z coordinate and also respect the global placement constraints. Finally, the iterative refinement stage, detailed in section 4.6.2, provides a greedy heuristic that optimizes linear 3D wire length while controlling the 3D-Vias to be within the upper bound.

4.6 3D Quadratic Placement

In Z-Place, we start applying a hybrid net modeling and weighting the connections as suggested by (VISWANATHAN; PAN; CHU, 2005). Nets of size 2 or 3 are modeled as a complete graph, while every net larger than three pins is mapped into a star. After transforming the netlist, we apply the quadratic wirelength function as described by equation

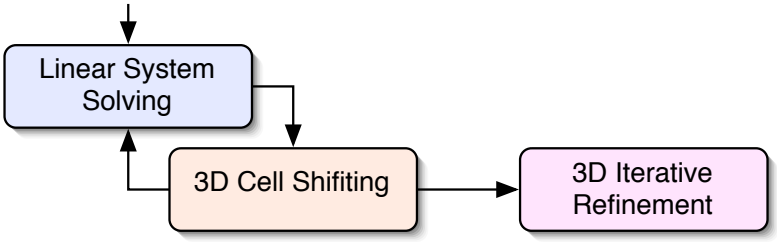


Figure 4.11: The global placement flow.

4.20. It can be observed that the three coordinates x, y and z can be optimized independently since $\phi(x, y, z) = \phi(x) + \phi(y) + \phi(z)$ as demonstrated in equation 4.20. Hereafter we describe just the computation performed for $\phi(x)$; the computation in y and z dimensions is analogous. By setting derivative of $\phi(x)$ to zero, we find the spring system's equilibrium state which minimize the total quadratic wire lengh. The minimum point can be found by solving a system of linear equations by rewriting the equation $\nabla\phi(x) = 0$ as $Q \times x = dx$. Q is a symmetric matrix which describes the weight of the connection between all pairs of cells. The vector dx contains the weights related to connections between cells and I/O pins. The vector of unknowns, x, represents the x coordinates of cells. The details of how a linear system that models cell placement for the quadratic wire length optimization is obtained and solved can be found in the review provided by Alpert et al. (ALPERT, C. et al, 1997)

$$\begin{aligned} &\phi(x, y, z) \qquad (4.20)\\ &= \frac{1}{2} \times \sum_{i,j=1}^{n} c_{ij} \times [\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}]^2 \\ &= \underbrace{\frac{1}{2} \times \sum_{i,j=1}^{n} c_{ij} \times (x_i - x_j)^2} + \underbrace{\frac{1}{2} \times \sum_{i,j=1}^{n} c_{ij} \times (y_i - y_j)^2} \\ &\qquad + \underbrace{\frac{1}{2} \times \sum_{i,j=1}^{n} c_{ij} \times (z_i - z_j)^2} \end{aligned}$$

The system of equations is solved using a pre-conditioned Conjugate Gradient method with incomplete Cholesky factorization of matrix Q as the pre-conditioner. After solving the system of equations, all the cells are placed within the sliced cube, but probably the cells will be concentrated in the middle of the cube and most of them are placed at invalid z coordinates. The next section describes our methodology for spreading the cells and fixing their z coordinate.

4.6.1 3D Cell Shifting

Solving the spring system yields the optimal quadratic wire length but cell overlaps are not taken into account. Moreover the optimal solution tends to concentrate the cells in the center of the placeable region. To remove overlaps and spread cells over the placeable region, we extend the Cell Shifting technique (VISWANATHAN; PAN; CHU, 2005) in order to perform 3D Cell Shifting. We call *Z-Cell Shifting* (HENTSCHKE, R. et al, 2006)

the methodology to spread the cells on the Z axis. The Z-Cell Shifting methodology allocates a tier to each cell based on an even distribution of the cell area. At each iteration of 3D Cell Shifting, the cell is moved in the direction of its tiers. Note that every intermediate solution is unrealistic (cells cannot be placed in metal layers or insulator area - invalid z coordinate). More accuracy on the wirelength optimization is obtained as the cell approaches a tier.

We compute an array of threshold points $threshold[]$ of $t + 1$ positions where $threshold[0] = 0$ and $threshold[t] = depth$. The remaining threshold points are defined in such a way that the cell and 3D-Via area are evenly distributed among tiers. Formally, the threshold points must be defined according to the equation 4.21, where a and b are consecutive indexes for the vector $threshold$.

$$\sum_{(c|threshold[a] \leq z_c < threshold[b])} (CellsArea(c)) \approx IdealCa_a \tag{4.21}$$

According to the formulation presented in section 4.5.6, the face-to-back integration of two adjacent tiers will influence the cells area on a tier. Figure A.7 ilustrates that tiers integrated in a face-to-back strategy will support less cells because of the 3D-Vias space.

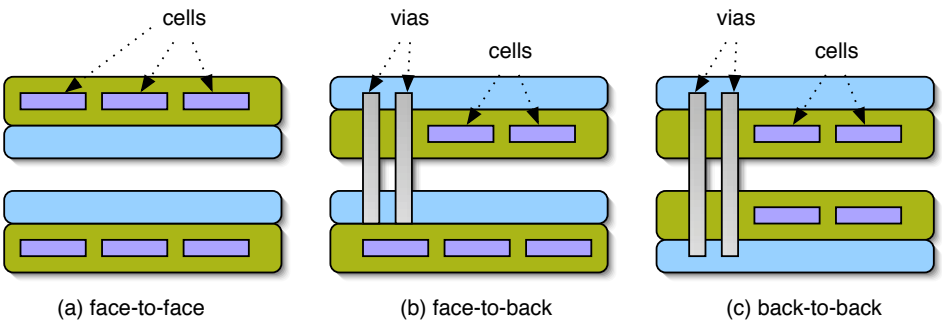


Figure 4.12: 3D integration strategies and how they impact the area distribution: (a) face-to-face and (b) face-to-back.

Note that the 3D-Via count at a tier t ($J[t]$) changes with each Cell Shifting iteration, so the proposed method will dynamically adapt itself to the imposed constraint for cell's area distribution.

After computing the threshold points the algorithm assigns each cell c to a tier t_c such that $threshold[t_c] < z_c < threshold[t_c + 1]$. A target z coordinate, $znew_c$, is computed to match the z coordinate of the corresponding slice on the sliced cube model.

The 3D Cell Shifting moves the cells into the three dimensions at the same time. The algorithm maps a cell c from the original position (x_c, y_c, z_c) to a target position $(xnew_c, ynew_c, znew_c)$ and provides a force F to be added to the system of equations described in section 4.6 that will realize the suggested movement on the next liner system solving. In order to obtain $newx_c$ and $newy_c$ we apply a similar algorithm to (VISWANATHAN; PAN; CHU, 2005). It starts by partitioning the slices into bins and assigning the cells to the bins they are in. The size of the bins is defined such that each bin holds approximately 4 cells. In our case the cells may not match any tier in intermediate steps, so we use the tier assignment provided by Z-Cell Shifting.

Figure A.8 illustrates the 3D Cell shifting process. In (a) the Z-Cell Shifting method is exemplified for 2 tiers, pointing out the threshold point computed according to the cells area. In (b) the 3D Shifting is illustrated. Two cells are moved in all axes at the same time.

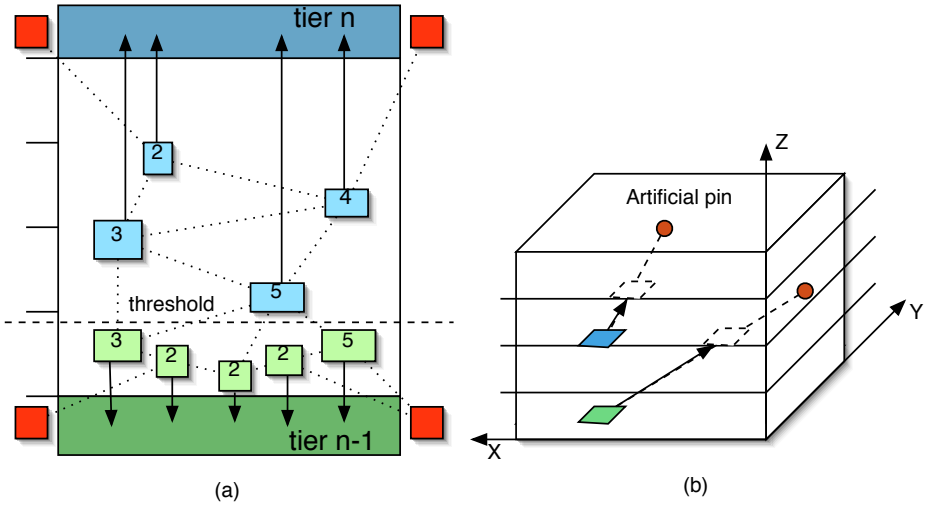


Figure 4.13: The Z-Cell Shifting methodology (a) and the 3D Cell Shifting algorithm (b).

The idea of using an α variable to limit the applied force providing a smooth movement to the destination position is also applied on Z-Place for all axes (including Z). Equation 4.22 shows the calculation of α_z, where $maxU$ is the current maximum utilization all over the circuit and $ChipHP$ represents the half perimeter of the chip ($width + height + depth$). The other axes, α_x and α_y are computed analogously.

$$\alpha_z = \frac{0.5 \times depth}{maxU \times ChipHP} \tag{4.22}$$

Viswanathan (VISWANATHAN; PAN; CHU, 2005) suggests that the artificial pin should be placed exactly in the boundary of the area, guaranteeing that all cells will be placed within the allowed area. One drawback of this method is that cells moving to a boundary position implies in extremely large forces or even infinity if the position matches exactly the boundary. In Z-Place, we want cells to be moved exactly to the upper most or lower most tier in the circuit, which overlaps with the boundary of the chip. In these cases, we place the artificial pin a little bit away from the boundary. As a drawback, a cell could be placed outside the chip. However, these cases will be corrected either by cell shifting or by rounding the cell z coordinates in the end of 3D cell shifting.

The 3D cell shifting iterates with a linear system solving until an acceptable spreading of the cells is obtained. After a few iterations the cells will be reasonably placed, as illustrated in figure 4.14.

In figure 4.14 we can observe graphically the 3D Cell Shifting process. Initially, observe that the linear system solving places the cells highly concentrated in the middle of the cube. Clls assigned to different tiers are painted in different colors. Observe that the cells are far away from their assigned tiers. In the second frame (that corresponds to an intermediate step), the cells start migrating to the assigned tier while at the same time spreading in the other axes (X and Y). It is possible that a cell might change its assigned

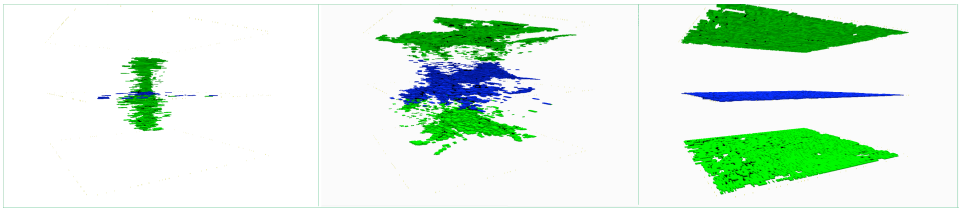


Figure 4.14: The visual effect of the 3D Cell Shifting methodology in a circuit with 3 tiers.

tier according to the movement of the other cells. Finally, the last frame shows the final arrangement of the cells after cell shifting. During these steps, the linear system is solved many times optimizing the quadratic wire length. However, it is not accurate in the initial stages since cells are overlapped and placed in invalid z coordinates. As the cells start approaching a valid z position, the quadratic wirelength optimization gets more accurate.

4.6.2 3D Iterative Refinement

After 3D Cell Shifting, the cells are placed on valid z coordinates and reasonably distributed in the X and Y axis. The tasks of the 3D iterative refinement stage are the following:

- Improve linear 3D wire length. The previous stages optimize quadratic wire length that does not correlate with linear wire length because longer connections are over-penalized.
- Iterate over the 3D-Vias count (array J) according to the upper bound constraint defined by equation 4.18. For a given 3D-Via layer v, if the 3D-Via count provided by 3D cell shifting (that ignores the constraint) is bellow the upper bound ($J[v] \leq Ubound[I[v]]$), then use the freedom to improve wire length; if it is above, work on reducing the number of 3D-Vias.
- Keep the balance of cells and 3D-Via areas according to the $IdealCa_t$ (equation 4.19).

The area of every tier is partitioned into bins of size b. Initially, b is set to a large number ((VISWANATHAN; PAN; CHU, 2005) suggests 5 times the value used for cell shifting) and gradually decreases until it reaches the size used on 3D Cell Shifting. For each value of b, all cells in the netlist are tested on all 4 neighbors in a plane and also the 5 closest bins in the upper adjacent and lower adjacent bins. The score of every movement is a real number evaluated according to the function shown in equations 4.23 and 4.24, where ΔWl_{norm}, ΔU_{norm} and ΔA_{norm} represent the normalized variation for wire length, utilization and tier area occupation respectively; m is the number of tiers; β, γ and λ real numbers between 0 and 1 such that $\beta + \gamma + \lambda = 1$. The number ΔA_{norm} is used to keep area utilization even. The best movement is taken. If no movement provides a positive evaluation, the cell is simply kept in its current position.

Overall, each parcel of the cost represent a number usually between -1 and 1. WL should be affected at the most by b, since the cell movement is no larger than b. The utilization U is at most b^2, which represents a full utilization of the bin. Finally, the last parcel A should be at most one tiers area although it is not guaranteed.

$$score = \beta \times \Delta Wl_{norm} + \gamma \times \Delta U_{norm} + \lambda \times \Delta A_{norm} \tag{4.23}$$

$$Wl_{norm} = \frac{WL}{b}; U_{norm} = \frac{U}{b^2}; A_{norm} = \frac{Ca_t + Va_t}{Ca \times 1/m} \tag{4.24}$$

Any tier migration movement may produce a delta on the 3D-Via count. Let us assume that we touched the 3D-Via layer v, changing $J[v]$ to $J_{new}[v]$. Note that only one 3D-Via layer can be touched by a single movement, since we test only neighbor tiers. There are two possible scenarios: $J_{new}[v] > UBound[I[v]]$ (a) and $J_{new}[v] \leq Ubound[I[v]]$ (b). The movement produces a variation in the 3D-Via count $\Delta 3Dvias_v = J[v] - J_{new}[v]$ that is used to change the scores. Note that a negative count means that the movement reduced the 3D-Via count.

First, let us consider situation (a), where we are above the allowed upper bound. The score is modified by encouraging the reduction and penalizing the addition of 3D-Vias (as shown in equation 4.25).

Situation (b) in which the 3D-Via count is bellow the upper bound have a degree of freedom to be explored by the algorithm in order to reduce wire length. It is possible, though, that the firstly taken cells occupy the remaining 3D-Vias resources and do not leave room for future cells that could take more benefit of the resource, which falls into a greedy algorithm. This problem is similar to the one on global routing where the first nets routed take all routing resources of a given bin edge, demanding future connections to turn around because the edge is fully congested. For this problem, an *early overflow* method, which starts penalizing routes before the full capacity is taken, is usually adopted. Similarly, for the 3D-Vias problem we define a *soft upper bound* $UboundS$ function, $UboundS \in I_{strategy} \times \mathbb{N}$ and $\forall_{i \in I_{strategy}} 0 \leq UboundS(i) \leq UBound(i)$. If $J_{new}[v] > UboundS[I[v]]$ then we modify the score similarly to situation (a) but with a multiplying factor that will reduce the impact of the $\Delta 3Dvias_v$ on the score. Since the score is a real number usually between -1 and 1, the $\Delta 3Dvias_v$ variable actually nullifies the other components of the score if it is not reduced by a factor. We use 0.3 as reducing factor, as show in equation 4.25.

$$score = score + \begin{cases} 0 & \text{if } J_{new}[v] < UboundS[I[v]] \\ 0.3 \times \Delta 3Dvias_v & \text{if } UboundS[I[v]] \leq J_{new}[v] < UBound[I[v]] \\ \Delta 3Dvias_v & \text{if } UBound[I[v]] \leq J_{new}[v] \end{cases} \tag{4.25}$$

To illustrate the possible trade-off between 3D-Vias and wire length achieved by the 3D Iterative Refinement method, an simple experiment was performed as follows. The benchmark ibm04 was picked and placed it in four different manners: no 3D-Vias constraint; an intermediate 3D-Via constraint; the tightest possible 3D-Via constraint; an hMetis partitioned solution (with approximately the optimal number of 3D-Vias) limiting our placer to 2D movements only. This different configurations produced results with variable wire length and Via count, as plotted in figure 4.15.

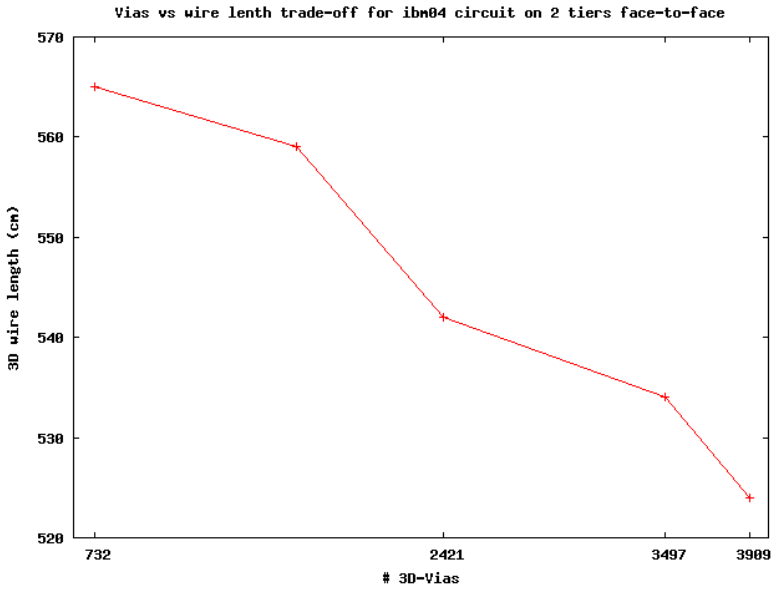


Figure 4.15: The Via count and 3D wire length (in cm) trade-off.

4.7 Experimental Results

In order to evaluate the merit of our placement tool we performed several experiments on public benchmark circuits IBM-PLACE. Z-Place was configured to constraint the number of 3D-Vias by a percentage of the tier area, according to the integration strategy. In all experiments on this section, the percentages used were: 80% for face-to-face, 20% for face-to-back, 10% for back-to-back.

Table 4.10 presents experimental results for 2 tiers integrated in face-to-face and table 4.11 for face-to-back. Note that on table 4.10 the 3D-Via size is having no effect on wire length because in all cases the 3D-Via count is bellow the allowed upper bound. This situation is expected to happen since on face-to-face the upper bound is large (80% of the area). Note that the 3D-Vias were not placed yet on this results, what would imply in an advantage of the smaller pitches.

Comparing the face-to-face and face-to-back strategies, the presented data shows, in summary, that face-to-face delivers better wire length optimization on the largest circuits, even comparing to the same via size used for face-to-back (since the face-to-back vias occupy valuable active space). There are two elements to be analyzed: first, note that the 3D-Via count for face-to-back is larger. When the algorithm is trading 3D-Vias for wire length, there is a tendency to insert more 3D-Vias in face-to-back because they are shorter than face-to-face. Second, note that on the smaller circuits there is a reduced wire length for face-to-back. Since the largest cases contain more 3D-Vias, they are more sensible to the drawbacks introduced by face-to-back 3D-Vias.

Table 4.12 presents the results for back-to-back integration on 2 tiers. Although back-to-back 3D-Vias occupy active area in both tiers, it is shorter than face-to-back and also allows a balanced area distribution of the cells. The results on table 4.12 demonstrates a advantage on wire length of the back-to-back approach.

Now let us analyze the effect of our area balancing and 3D-Via planning considering the integration strategy for 3 tiers. On table 4.13 we present the cells area and 3D-Vias area for each tier as well as 3D-Via count for each via layer. The data shows clearly the

Table 4.10: Experimental results for 2 tiers face-to-face

	f2f 1μm		f2f 5μm		f2f 10μm	
	WL	# 3D-Vias	WL	# 3D-Vias	WL	# 3D-Vias
ibm01	1.67E+06	5375	1.67E+06	5375	1.67E+06	5375
ibm02	3.60E+06	8045	3.60E+06	8045	3.60E+06	8045
ibm03	4.14E+06	10055	4.14E+06	10055	4.14E+06	10055
ibm04	5.00E+06	11721	5.00E+06	11721	5.00E+06	11721
ibm05	7.68E+06	10544	7.68E+06	10544	7.68E+06	10544
ibm06	4.99E+06	13088	4.99E+06	13088	5.03E+06	13210
ibm07	7.76E+06	19099	7.76E+06	19099	7.76E+06	19099
ibm08	8.92E+06	16659	8.92E+06	16659	8.92E+06	16659
ibm09	9.70E+06	24598	9.70E+06	24598	9.70E+06	24598
ibm10	1.65E+07	32573	1.65E+07	32573	1.65E+07	32573
ibm11	1.41E+07	30987	1.41E+07	30987	1.41E+07	30987
ibm12	2.03E+07	35192	2.03E+07	35192	2.03E+07	35192
ibm13	1.66E+07	35175	1.66E+07	35175	1.66E+07	35175
ibm14	3.12E+07	58438	3.12E+07	58438	3.12E+07	58438
ibm15	3.71E+07	68856	3.71E+07	68856	3.67E+07	69162
ibm16	4.56E+07	78418	4.56E+07	78418	4.56E+07	78418
ibm17	5.84E+07	89914	5.84E+07	89914	5.84E+07	89914
ibm18	3.99E+07	71996	3.99E+07	71996	3.99E+07	71996
avg	1.85E+07	34485	1.85E+07	34485	1.85E+07	34509

Table 4.11: Experimental results for 2 tiers face-to-back

	f2b 5μm		f2b 25μm	
ibm01	1.63E+06	6819	1.86E+06	562
ibm02	3.57E+06	9506	5.11E+06	519
ibm03	4.19E+06	12419	4.91E+06	1132
ibm04	5.32E+06	14482	6.34E+06	1630
ibm05	7.82E+06	13559	1.08E+07	3066
ibm06	5.43E+06	15980	6.00E+06	1737
ibm07	8.27E+06	23303	8.97E+06	2072
ibm08	9.38E+06	21158	1.06E+07	1788
ibm09	1.01E+07	29753	1.14E+07	1418
ibm10	1.72E+07	38019	1.91E+07	2503
ibm11	1.59E+07	38525	1.47E+07	1781
ibm12	2.13E+07	40429	2.40E+07	3296
ibm13	1.79E+07	45130	1.92E+07	2113
ibm14	3.44E+07	72026	3.74E+07	3881
ibm15	4.39E+07	88611	4.48E+07	3782
ibm16	5.13E+07	95307	4.68E+07	5050
ibm17	6.33E+07	104043	6.92E+07	7077
ibm18	4.35E+07	92636	4.88E+07	5409
Avg	2.02E+07	42317	2.17E+07	2712

effect of face-to-face integration (more vias could be reached) in contrast with face-to-back (active area occupancy by 3D-Vias). Compared to the experiments with 2 tiers, it

Table 4.12: Experimental results for 2 tiers back-to-back

	b2b 15μm		b2b 25μm	
ibm01	1.71E+06	609	1.72E+06	600
ibm02	3.71E+06	760	3.71E+06	747
ibm03	4.32E+06	1433	4.35E+06	1465
ibm04	5.27E+06	2240	5.27E+06	2244
ibm05	8.12E+06	3443	8.13E+06	3446
ibm06	4.99E+06	1605	5.04E+06	1594
ibm07	8.16E+06	1671	8.15E+06	1674
ibm08	9.21E+06	1647	9.21E+06	1618
ibm09	9.84E+06	1916	9.90E+06	1035
ibm10	1.80E+07	3142	1.80E+07	2473
ibm11	1.54E+07	2462	1.54E+07	1571
ibm12	2.11E+07	3398	2.09E+07	3187
ibm13	1.78E+07	2898	1.78E+07	1512
ibm14	3.32E+07	5371	3.34E+07	4354
ibm15	3.91E+07	6434	3.92E+07	6386
ibm16	4.63E+07	7002	4.63E+07	3524
ibm17	6.13E+07	8284	6.14E+07	5579
ibm18	4.45E+07	8023	4.45E+07	8066
avg	1.96E+07	3463	1.96E+07	2838

has an advantage on average of 1.56 to 1.85 on wire length. Table 4.14 presents results of the same algorithm with very big 3D-Vias for face-to-back (25μm pitch). The results as well as the area balancing deteriorates significantly. The average wire length go up from 1.56 to 3.25.

Tables 4.15, 4.16 and 4.17 present results for 4 tier circuits using different technologies. The first two tables (4.15 and 4.16) are generated from a face-to-face, back-to-back and face-to-face configuration, differing only on the back-to-back 3D-Via pitch that is 10μm for table 4.15 and 15μm for table 4.16. Note that the larger 3D-Via pitch led to a worse wire length in average. Finally, table 4.17 present results for a face-to-face, face-to-back and face-to-back configuration. Note that this configuration was the best for wire length and the one with more balanced 3D-Vias.

In general, the 4 tier tables present results better than 3 tiers ones by around 10%. The three strategies tried to obtain similar results in average, but individual results were significantly different suggesting that the integration strategy is somehow related to the design.

4.7.1 Summary of Partial Conclusions

The global placement step of Z-Place goal is to provide a reasonable cell placement with balanced area and feasible 3D-Via planning at the same time as optimizing wire length into 3D.

The algorithm is based on the quadratic placement algorithm extended for 3D circuits with a number of new features and algorithms that deliver a true 3D optimization while considering 3D-Via related constraints. Z-Place considers the integration strategy and provides room for face-to-back and back-to-back 3D-Vias that demand active area. It also allocates the number of 3D-Vias according to the strategy and to their required area.

Table 4.13: Experimental results with 3 tiers disposed in face-to-face (1 micra) and face-to-back (5 micra) respectively. C denotes the cells area on the tier, while V denotes the area ocupied by 3D-Vias on the same tier.

	Area Tier1		f2f	Area Tier2		b2f	Area Tier3		WL
bench	C	V	# V	C	V	# V	C	V	
ibm01	31%	0%	3249	31%	5%	4731	32%	0%	1.51E+06
ibm02	32%	0%	5616	31%	6%	6897	32%	0%	2.83E+06
ibm03	32%	0%	5693	31%	6%	8086	32%	0%	3.82E+06
ibm04	31%	0%	7211	31%	6%	10022	32%	0%	4.80E+06
ibm05	32%	0%	7169	31%	5%	9918	32%	0%	6.60E+06
ibm06	31%	0%	8150	30%	7%	10874	32%	0%	4.47E+06
ibm07	31%	0%	12012	31%	6%	16538	32%	0%	7.35E+06
ibm08	32%	0%	11799	31%	6%	15857	32%	0%	7.23E+06
ibm09	31%	0%	15379	31%	6%	20254	32%	0%	8.58E+06
ibm10	32%	0%	19441	31%	5%	25458	32%	0%	1.50E+07
ibm11	31%	0%	18793	31%	6%	25143	32%	0%	1.27E+07
ibm12	32%	0%	22349	31%	5%	29087	32%	0%	1.73E+07
ibm13	31%	0%	22772	30%	7%	32271	32%	0%	1.55E+07
ibm14	32%	0%	35472	31%	5%	49826	32%	0%	2.84E+07
ibm15	31%	0%	44799	30%	7%	62272	32%	0%	3.34E+07
ibm16	31%	0%	48724	31%	6%	66898	32%	0%	4.22E+07
ibm17	31%	0%	57994	31%	5%	76099	32%	0%	5.40E+07
Avg.:	31%	0%	20390	31%	6%	27661	32%	0%	1.56E+07

These facts provide more realism and more conclusive wire length numbers compared to known solutions in 2D.

The 3D-Cell shifting method is based on a threshold algorithm that obtains the exact cells balance while at the same time iterating with a wire length optimization. The following step, iterative refinement, provides heuristics to improve wire length and cells balance into 3D. The algorithm is constrained by an upper bound of 3D-Vias that is triggered by their required area. Within the upper bound, Z-Place is free to insert 3D-Vias. Experimental results demonstrated that by manipulating this upper bound one could play with a trade-off between number of 3D-Vias and 3D wire length. This fact lead to the conclusion that the best strategy depends on the technology used. Since Z-Place algorithm is triggered by the 3D-Via area, it will automatically adapt to the technology data, improving the wire length only if there is available space for that.

4.8 Detailed Placement

After global placement, the task of detailed placement is to legalize and improve wire length. Since global placement stage implements techniques for even distribution of the cells area and bounded planning of 3D-Vias, it is reasonable that future stages keep this distribution and do not perform any tier migration movements.

The detailed placement flow is illustrated in figure 4.16. It is composed of a Cell Sweeping stage for layout compaction, a cell legalization algorithm as described in (KHATKHATE, A. et al, 2004) and improvements algorithms that can be performed by either Simulated Annealing (KIRKPATRICK; GELATT; VECCHI, 1983) or Threshold

Table 4.14: Experimental results with 3 tiers disposed in face-to-face (1 μm) and face-to-back (25 μm) respectively. C denotes the cells area on the tier, while V denotes the area ocupied by 3D-Vias on the same tier.

	Area Tier1		f2f	Area Tier2		b2f	Area Tier3		WL
bench	C	V	# V	C	V	# V	C	V	
ibm01	40%	0%	2441	11%	14%	556	35%	0%	2.44E+06
ibm02	37%	0%	3080	8%	19%	1020	37%	0%	4.97E+06
ibm03	34%	0%	3789	7%	23%	1586	36%	0%	5.90E+06
ibm04	34%	0%	5384	10%	22%	1989	34%	0%	7.26E+06
ibm05	31%	0%	6264	9%	30%	3125	30%	0%	9.79E+06
ibm06	36%	0%	4941	8%	19%	1430	37%	0%	6.66E+06
ibm07	37%	0%	6982	9%	17%	2117	37%	0%	1.17E+07
ibm08	39%	0%	8291	11%	15%	1903	35%	0%	1.25E+07
ibm09	43%	0%	10043	9%	13%	1863	35%	0%	1.64E+07
ibm10	40%	0%	14009	14%	11%	2579	35%	0%	2.65E+07
ibm11	44%	0%	13633	13%	7%	1269	35%	0%	2.53E+07
ibm12	41%	0%	16800	14%	11%	2603	35%	0%	3.57E+07
ibm13	44%	0%	15016	9%	11%	2342	36%	0%	3.07E+07
ibm14	41%	0%	24452	11%	12%	4594	36%	0%	6.62E+07
ibm15	40%	0%	27440	9%	15%	5762	36%	0%	7.84E+07
ibm16	45%	0%	35858	14%	7%	3354	35%	0%	9.85E+07
ibm17	39%	0%	40260	12%	13%	7759	36%	0%	1.14E+08
Avg.:	39%	0%	14040	10%	15%	2697	35%	0%	3.25E+07

Table 4.15: Experimental results with 4 tiers disposed in face-to-face, back-to-back and face-to-face respectively with 3D-Via pitches 1μm (f2f) and 10μm (b2b).

	f2f	b2b	f2f	
	# V	# V	# V	WL
ibm01	2693	608	2345	1.37E+06
ibm02	3761	770	3841	2.86E+06
ibm03	4561	1506	4403	3.80E+06
ibm04	5947	1189	5206	4.69E+06
ibm05	5819	2383	5758	6.44E+06
ibm06	6487	1571	6384	4.46E+06
AVG	4878	1338	4656	3.94E+06

Accept (DUECK; SCHEUER, 1990) heuristics.

One of the important issues to be solved in a detailed placement algorithm is to define a proper data structure for the cell placement. In global placement, since cells could overlap with each other, the coordinate of the cell could be stored with the cell's data structure itself. In detailed placement the relative horizontal and vertical order of the cells is a very valuable information. For example, legalization algorithms need to assign a *row* for every cell and to sort the cells horizontally by their x position. Detailed improvement algorithms, discussed in section 4.8.2, might need to iterate with cells in the neighborhood, demanding a data structure that fastly access nearby cells. For those reasons, Z-Place uses a data structure as illustrated in figure 4.17. The placement is represented by an array of

Table 4.16: Experimental results with 4 tiers disposed in face-to-face, back-to-back and face-to-face respectively with 3D-Via pitches 1μm (f2f) and 15μm (b2b).

	f2f	b2b	f2f	
	# V	# V	# V	WL
ibm01	2563	372	2479	1.43E+06
ibm02	3757	785	3732	3.03E+06
ibm03	4771	1504	4803	4.07E+06
ibm04	5634	1149	5477	4.83E+06
ibm05	5829	2402	5921	6.86E+06
ibm06	6065	1522	6284	4.69E+06
	4770	1289	4783	4.15E+06

Table 4.17: Experimental results with 4 tiers disposed in face-to-face, face-to-back and face-to-back respectively with 3D-Via pitches 1μm (f2f) and 5μm (f2b).

	f2f	f2b	f2b	
	# V	# V	# V	WL
ibm01	2102	2778	2914	1.65E+06
ibm02	4284	5676	5348	2.88E+06
ibm03	4266	5815	5972	4.00E+06
ibm04	6035	7045	7017	4.80E+06
ibm05	6174	7558	7756	6.49E+06
ibm06	6931	7215	7788	5.02E+06
avg:	4965	6015	6133	4.14E+06

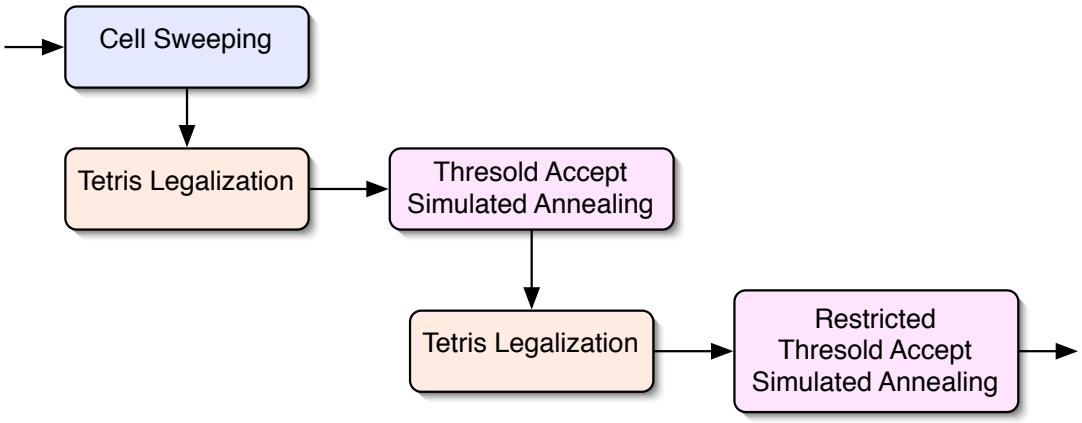


Figure 4.16: Detailed Placement flow on Z-Place

rows. Each row contains an array of cells represented by their id number.

In order to obtain more information about the cell, the id is used to index an array of cells information. It is trivial to obtain any cell neighborhood and to sort the rows. For now on, consider that every cell c has an associated row r_c and index i_c on it.

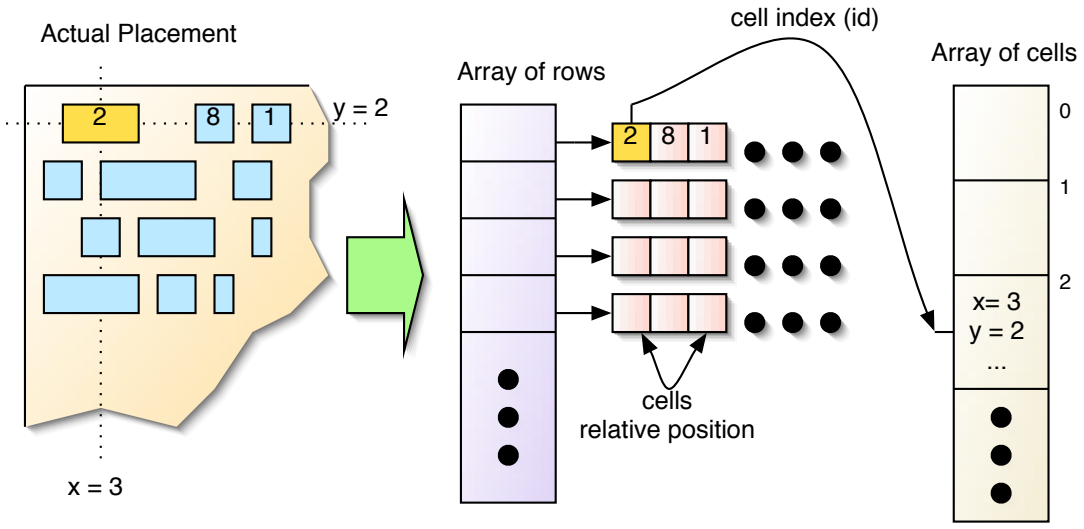


Figure 4.17: Data Structure used to store cell positions for detailed placement

4.8.1 Cell Sweeping

This stage is used to compress the cells on the left edge of the layout, eliminating most of the whitespace allocated on previous stages. For that, the cell sweeping algorithm slices the area in a very large number of vertical lines obtaining s slices. The netlist is then sorted by its x coordinate; cells are mapped to one vertical slice by linear mapping (weighted by their area). Figure 4.18 illustrates the effect of cell sweeping into the flow.

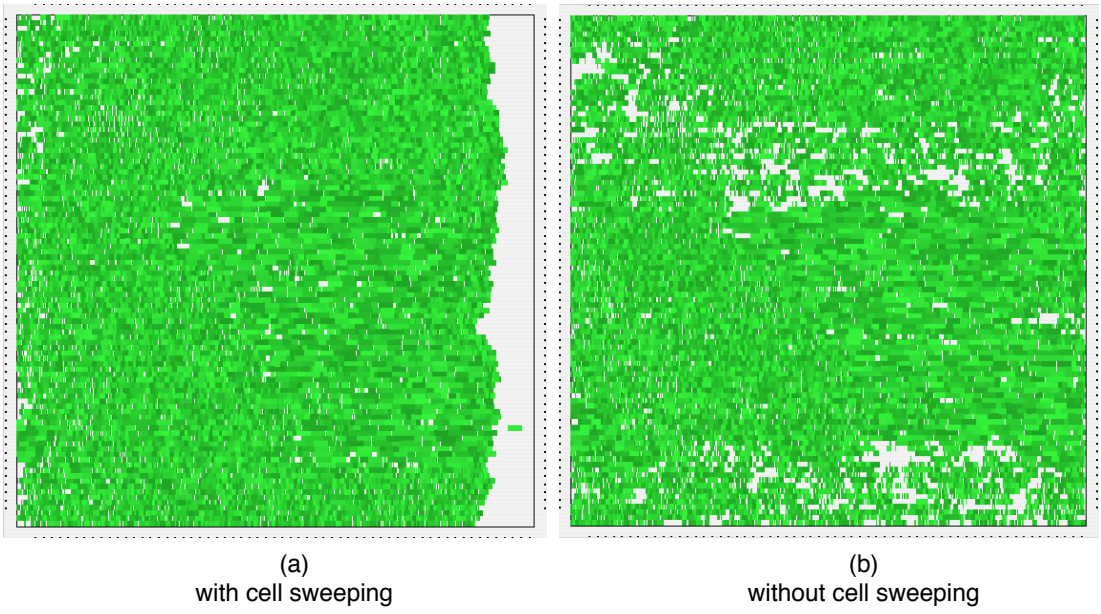


Figure 4.18: The effect of cell sweeping over the benchmark ibm01.

4.8.2 Threshold Accept Improvement

Simulated Annealing (SA) (KIRKPATRICK; GELATT; VECCHI, 1983) is a general purpose stochastic algorithm that is applied to several optimization problems of Computer Science, Engineering, Physics and Chemistry. In computer science, Simulated Annealing solves efficiently many combinatorial optimization NP-Hard problems. The simulated process is analogous to the natural annealing. First, an imperfect material is heated up such that the molecules move freely. While the temperature is slowly decreased, molecules lose mobility, joining together. The structure slowly starts to crystallize removing initial material imperfections.

Threshold Accept heuristic (DUECK; SCHEUER, 1990) was proposed as an evolution of the well known Simulated Annealing algorithm. Dueck and Scheuer argued that Threshold Accept is simpler and “apparently superior" than Simulated Annealing.

The algorithm starts from some initial solution and searches the neighborhood randomly. Compared to a greedy algorithm that accept only down-hill moves [1], Threshold Accept (TA) accepts some up-hill moves if they are under a certain threshold. Simulated Annealing (SA) works similarly, but the threshold distribution is more complex and probabilistic while in Threshold Accept it is purely deterministic. Both Simulated Annealing and Threshold Accept algorithms uses up-hill moves in order to scape from local minima.

Simulated Annealing is very successfully applied to the cell placement problem (SECHEN; VICENTELLI, 1985) (HENTSCHKE; REIS, 2003a) (TAGHAVI; YANG; CHOI, 2005). It is major and well known drawback is excessive CPU usage, since it must search a vast space in order to converge to good solutions. On the other hand, if appropriate techniques are used to prune the search space, the algorithm can converge very fast to high quality solutions.

In this thesis, the algorithm Threshold Accept is applied for cell placement. It has very similar features compared to SA, but it was preferred due to its simplicity. While Simulated Annealing literature is vast there seems to be no work on TA for cell placement. Under our experiment, both algorithms had similar quality (in terms of CPU time usage and placement quality).

For the placement problem, TA is used as an iterative algorithm. Usually it starts from a randomly generated initial placement, but other alternatives are also explored in the literature (HENTSCHKE; JOHANN; REIS, 2003) (HENTSCHKE; REIS, 2003b). The algorithm modifies the initial solution in small steps, called perturbations. The perturbations might be accepted or rejected, depending on their result and on the system threshold. The quality of the current solution must be measured by a cost function. The threshold is defined as an external parameter that will control the greedy behavior of the algorithm. It starts very relaxed, in a high threshold. As TA algorithm advances, the threshold is slowly decreased. When the threshold reaches the ZERO value, TA will be fully greedy.

The Threshold Accept algorithm is written below in pseudo-code. The basic structure is composed by two loops: an *outer loop* and *an inner loop*. The outer loop controls the threshold with the *schedule function*. The inner loop actually modifies the current placement with the *perturbation* and *cost* functions. The perturbation and cost functions are discussed on sections 4.8.2.1 and 4.8.2.2 respectively while threshold scheduling is discussed in section 4.8.2.3.

The Threshold Accept algorithm is generic, but each new implementation requires

[1] Down-hill moves are defined as perturbations to the current state of the solution that actually improves it under a cost function. Up-hill moves are the ones that leads to deterioration of the solution.

Algorithm 2 Pseudo-Code of Simulated Annealing algorithm

```
1: temp = Initial_Temperature;
2: place = Initial_Placement;
3: //Outer Loop
4: while ( Not Ending Outer Loop Condition )
5:    //Inner Loop
6:    while ( Not Ending Inner Loop Condition )
7:       new_place = Pertubation(place);
8:       delta = Cost(new_place) − Cost(place);
9:       if (Accept(delta, temp)) then
10:         place = new_place;
11:      end if
12:   end while
13:   temp = Schedule(temp)
14: end while
15: return place
```

proper tuning of the perturbation, cost and temperature schedule functions. Each of them will be analyzed carefully in the following sections.

4.8.2.1 Cell Perturbations

Z-Place has three types of cell perturbations:

- Windowed swap of cells in the same tier. A cell a is chosen randomly from the netlist. It will be swapped with a second cell b that is placed within a window. The window radius is 5 times the height of a row. Given a row r_b and an x coordinate x_b (the placement area starts at x_{ini} and goes to x_{ini} + $width$) within the window, the i_b is calculated by equation 6. It is allowed a size unbalance of the cells of up to twice the size. That movement can generate cell overlaps.

- Greedy swap of cells in the same tier. As done in (HENTSCHKE; REIS, 2003a), a few greedy moves are applied to increase convergence of the optimization.

- Cell sliding. A cell is taken randomly and slided to the left and right if there is available space. This perturbation keeps the placement legalization (e.g. it does not introduce any cell overlap).

$$i_b = \frac{1}{width} \times nr_b \times (x_b - x_{ini}) \tag{4.26}$$

The perturbation probabilities are tuned as follows: 72% to the Windowed Swap, 18% is to the Greedy Swap and 10% for the Cell Slide. Equation 4.26 assumes that the cells are well distributed in the row otherwise the suggested mapping may be out of the window. We observe, though, the window is not a strict constraint and can be broken for CPU time benefit. Both algorithms obey the basic properties: simplicity (since they work with only a few gates - one or two), randomization and consistency. However, in order to prune the search space of the algorithm, it is usual to use a windowed perturbation. The windowed perturbation obviously limit the capability of TA to move a cell for long

distances, being limited to local improvements. For this reason, windowed perturbation suits detailed placement very well, since all global decisions were taken by the previous stage.

4.8.2.2 Cost Function

The cost function is the quality measure of the current placement in the Threshold Accept algorithm. Z-Place computes the total 3D half perimeter. The process of computing 3D half perimeter for one net consists of obtaining the minimum bounding box that contains all pins of a net; the sum of the bounding box width, height and depth is used.

If Z-Place is considering critical connections, a separate module computes the critical wire length that is defined by the sum of the length of all driver to critical sink connections.

It is possible to design multi objective cost functions, such as the one proposed for iterative refinement at section 4.6.2. This matter is addressed in (SAIT; YOUSSEF; MALEH, 2001).

4.8.2.3 Threshold Schedule

This section describes how to control the threshold of the algorithm properly. A threshold scheduling (as temperature schedule for SA) is composed of three stages:

1. Finding the initial threshold.
2. How should be the variation of the threshold until it reaches Zero
3. What is the ending criteria of the inner and outer loop of TA.

At each outer loop iteration, one important definition for understanding the temperature is the TA acceptance ratio. It is calculated by the number of accepted perturbations divided by the number of inner loop iterations. A very high threshold will imply in 100% acceptance ratio. As the threshold is decreased, the acceptance ratio becomes closer to zero.

The next sections will present each of these stages in more detail.

4.8.2.4 Finding the initial threshold

The traditional literature on Simulated Annealing and Threshold Accept claim that the initial temperature/threshold should be set to a very high number, so that the algorithm will be able to scape from local minima independently of the given initial solution. This problem itself is complex because the sufficiently high number would depend on the nature of the problem. For example, suppose one is optimizing a graph coloring problem with an initial solution costing 10 colors. Now suppose that the same SA or TA engine will be applied for cell problem, with an initial solution costing 10M units (wire length for instance). Observe that the acceptance function on the Threshold accept heuristic is simply to check whether a delta caused by perturbation is bellow the threshold. Clearly, the value of initial threshold on the graph coloring must be orders of magnitude different from the initial threshold of cell problem. The same analysis is valid for Simulated Annealing.

Now suppose that we have a solution costing 7M units for our cell problem. If one use the same threshold as proposed in the last paragraph, the 7M solution will go up to higher costs (probably above 10M solution presented above) since every random perturbation will be accepted for the sake of escaping local minima (HENTSCHKE; JOHANN; REIS,

2003) (HENTSCHKE, 2002). This section discusses the idea (already existent in the literature such as (CHANG; CONG, 2003)) of starting from a initial solution without deteriorating but trying to reach the exact threshold that keeps the cost untouched (or minimally touched). That is exactly the job of detailed placement on Z-Place flow, since it is undesired to ignore the global solution provided by the former step. The idea of starting on lower thresholds can be understood as a search pruning method as well, speeding up the algorithm.

Coming back to the example, a good initial threshold for the solution costing 7M would be sufficiently low such that oscillations in the cost would still be acceptable to scape local minima but long up-hill oscillations would be prohibited. More precisely, the cost of the current solution could potentially go up, but in average it should remain 7M.

Under the model above, the algorithm that obtains the initial threshold can be simply binary searching for a threshold such that the inner loop average delta is zero (or close to zero). The algorithm 3 details this idea.

Algorithm 3 Pseudo-Code of the initial temperature calculation

1: $temp_{up} = \infty$
2: $temp_{low} = 0$
3: $place_{current} = place_{initial}$
4: **repeat**
5: $temperature = ((temp_{up} - temp_{low})/2) + temp_{low}$
6: $delta_{cont} = 0$
7: $delta_{avg} = 0$
8: **for** ($iterations = 0$ to $InnerLoop_{iterations}$)
9: $place_{new} = Pertubation(place_{current})$
10: $delta = Cost(place_{new}) - Const(place_{current})$
11: **if** $(Accept(delta, temperature))$ **then**
12: $delta_{avg} = delta_{avg} + delta$
13: $delta_{cont} + +$
14: **end if**
15: **end for**
16: $delta_{avg} = delta_{avg}/delta_{cont}$
17: **until** $delta_{avg} iscloretoZero$
18: **return** $temperature$

4.8.2.5 Threshold variation

The threshold update function is an important part of the scheduling. Let us first discuss the existing solutions for Simulated Annealing and after that adopt one solution for Threshold Accept. The literature classifies schedules as static and adaptive ones.

Static schedules are pre-determined before the simulation process starts. One possibility is a straight line schedule, as shown in figure 4.19(a). In such approach, the time is evenly allocated for all temperature slots. Another classical schedule is to multiply the current temperature by 0.9 for example (values between 0.85 to 0.99 are used). In such case, the curve looks like figure 4.19(b). It might look more attractive, since less time is spent on highest temperatures, while more time is spend at the lower temperatures. On the other hand, such approaches can be too greedy, depending on the nature of the design.

It is known that one static schedule will work better in one kind of design, while

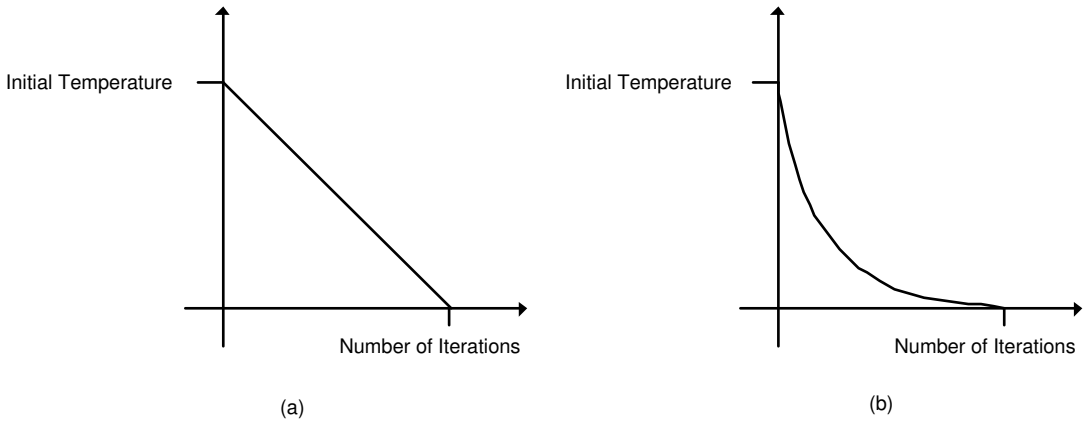


Figure 4.19: Example of static Annealing Schedules

another schedule will be better for others. For this reason, dynamic (adaptive) approaches were proposed (LAM; DELOSME, 1988). The basic idea of adaptive schedules is to keep track of the acceptance ratio. If the acceptance ratio is too high, the annealing process is oscillating between random states, suggesting a large decrease on the temperature. A low acceptance ratio can be an indicative of unfinished work of the current temperature, suggesting small changes on the temperature. A very rudimentary adaptive schedule, as suggested in (CHANG; CONG, 2003), is described as shown in figure 4.8.2.5.

Algorithm 4 Pseudo-Code of a simple adaptive annealing schedule

1: $temperature = temperature_{current}$
2: $ratio_{acceptance} = ratio_{accept}$
3: **if** $(ratio_{acceptance} > 0.99)$ **then**
4: $\quad temperature = temperature/2$
5: **else if** $ratio_{acceptance} > 0.95)$ **then**
6: $\quad temperature = temperature \times 0.8$
7: **else if** $ratio_{acceptance} > 0.9)$ **then**
8: $\quad temperature = temperature \times 0.9$
9: **else if** $(ratio_{acceptance} > 0.8)$ **then**
10: $\quad temperature = temperature \times 0.95$
11: **else**
12: $\quad temperature = temperature \times 0.98$
13: **end if**

One state-of-the-art and sophisticated dynamic schedule is found in the work of Jimmy Lam (LAM; DELOSME, 1988). Let *invtemp* be the inverse of current temperature $(1/temp)$, σ be the standard deviation of the cost function on the current temperature and ϕ be the accept ratio. Use λ as a quality factor - higher values decreases the quality for running time. The formulae to calculate the next temperature *ntemp* is given in equation 4.28.

$$ntemp = \frac{1}{invtemp + \lambda(\frac{1}{\sigma}) \times (\frac{1}{invtemp^2 \sigma^2}) \times (\frac{4\phi(1-\phi)^2}{(2-\phi(dp))^2})} \tag{4.27}$$

The formula above can be trivially rewritten for Threshold Accept as follows, where $invthresold$ is $1/threshold$ and $nthreshold$ is the computed new threshold:

$$nthreshold = \frac{1}{invthreshold + \lambda(\frac{1}{\sigma}) \times (\frac{1}{invthreshold^2 \sigma^2}) \times (\frac{4\phi(1-\phi)^2}{(2-\phi(dp))^2})} \tag{4.28}$$

In Z-Place, this function is tuned with $\lambda = 50000$. Before applying the formulae, Z-Place checks if acceptance ratio (ϕ) is larger than 0.98. In this case, the current threshold is dropped by $nthreshold = 0.4 \times threshold$. Z-Place also checks if either σ, ϕ or $threshold$ is zero which leads to $nthreshold = 0$.

4.9 3D-Via Placement

This section address 3D-Via placement and legalization. After the placement tool performs the whole flow, we introduce the 3D-Via placement, that is illustrated by figure 4.20. Between the tiers we create 3D-Via layers. For every 3D-Via layer (that is interleaved with cell layers) we must obtain a fully legalized placement of the 3D-Via objects. Instead of treating the 3D-Vias as placeable objects during the cell placement, we prefer to simplify the problem and leave the cell placement untouched. In the end of the whole flow, we fix the cell's positions and try to place the 3D-Vias in such a way that wire length is minimally affected. We suppose that each net uses exactly one 3D-Via since that resource is expensive. Every 3D-Via belongs to a net whose bounding box is delimited by all cells connected to it. The applied method tries to place the 3D-Vias inside its bounding box. If this placement is possible we place the 3D-Vias with no overhead to the wire length. For the nets in which the proposed method is not accomplished by our heuristic, the algorithm's objective function switches to a function that minimizes the distance of the 3D-Via to the nearest edge of the net bounding box. Additionally, face-to-back and back-to-back integration requires an additional legalization step to move the cells away from the 3D-Via places. This step is skipped by our algorithm, but is largely studied in the literature, as reviewed in section 3.6.3. It is important to highlight that the algorithm presented here for 3D-Via legalization suits better to face-to-face integration; for face-to-back and back-to-back it might be better to approach the problem during cell placement.

The problem is formulated (section 4.9.1) targeting at wire length overhead minimization. The goal is to study the wire length overhead caused by different 3D-Via technologies while providing means of a more realistic evaluation of the 3D placement. Section 4.9.1 presents the problem formulation and section 4.9.2 presents an algorithm based on cell legalization algorithms. We incorporated a new cost function that will attend the formulation presented in section 4.9.1.

An example of 3D-Via placement is presented in figure 4.21.

4.9.1 Problem Formulation

Let $V = \{v_1, v_2, ..., v_k\}$ be the set of 3D-Vias, z be the number of tiers, $T = \{t_1, t_2, ..., t_{z-1}\}$ be the set of 3D-Via layers and $t(v)$ be a $V \times T$ function that maps a 3D-Via to a layer. Every 3D-Via measures w for width and h for height. The *3D-Via placement problem* can be defined as a function $place \in V \times (\Re, \Re)$ that assigns a (x, y) coordinate to every 3D-Via such that no overlap is allowed between any pair $(v, w) \mid v, w \in V$, if $t(v) = t(w)$. The overlap between v and w is verified by comparing

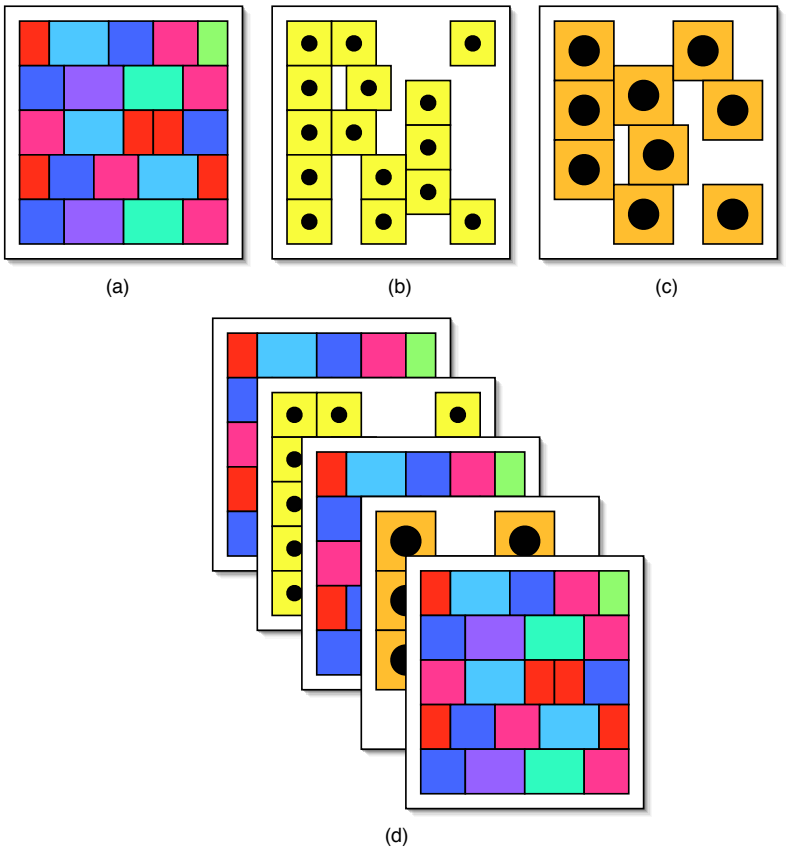


Figure 4.20: The placement of 3D-Vias; (a) placement of cells; (b) placement of face-to-face vias; (c) placement of face-to-back vias; (d) mixed integration.

the rectangles of dimensions *w* and *h* centered at $place(v)$ and $place(w)$ respectively.

Let $N = \{n_1, n_2, ..., n_m\}$ be the set of nets in the circuit and $net(v)$ be a $V \times N$ function that maps a 3D-Via to its net. Every net $n \in N$ has a minimum bounding box $B_n = (x_1, y_1) \Rightarrow (x_2, y_2)$ that contains all cells connected to n. We want to constraint the problem in order to place the center coordinate of the rectangle defined by the 3D-Vias $v \in V$ inside $B_{net(v)}$. It might not be possible, so we relax this constraint to minimize the wire length overhead. Additionally, a preferred position $P = (x_p, y_p)$ for the 3D-Via is defined and supported by the placement algorithm; P must be within the bounding box and could be obtained from a Steiner tree. In this work this position is naively set to the center of the bounding box. The objective function will be a combination of wire length overhead and displacement from P.

The problem is illustrated in figure 4.22.

4.9.2 3D Via Placement and Legalization

To solve the problem we perform a two-step method as follows. First we place all the 3D-Vias v exactly in the middle of their bounding boxes $B_{net(v)}$ as shown in figure 4.23. The second step is to perform the 3D-Vias legalization. The main difference from the standard legalization lies in the fact that any position inside the bounding box is optimal, having no cost.

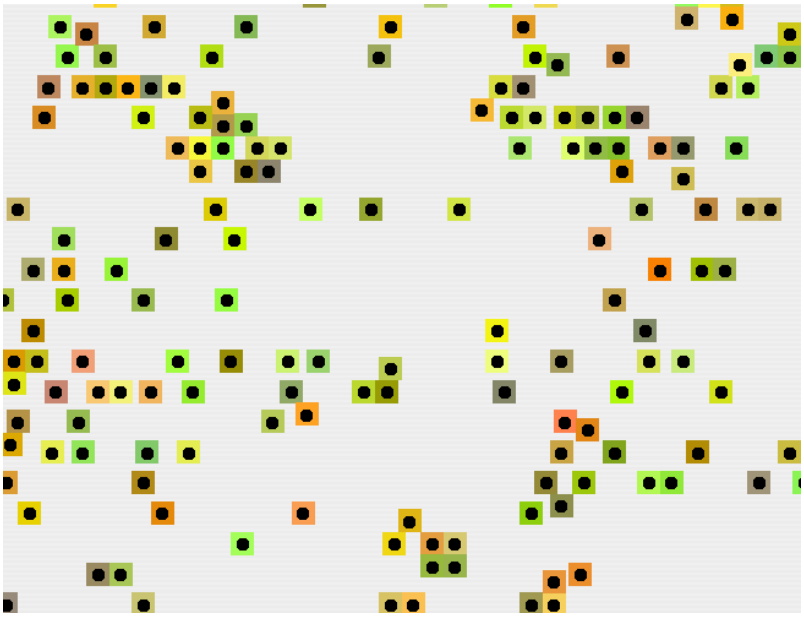


Figure 4.21: An example of legally placed 3D-Vias

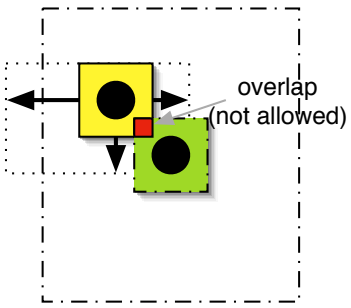


Figure 4.22: 3D-Vias ant their respective bounding boxes.

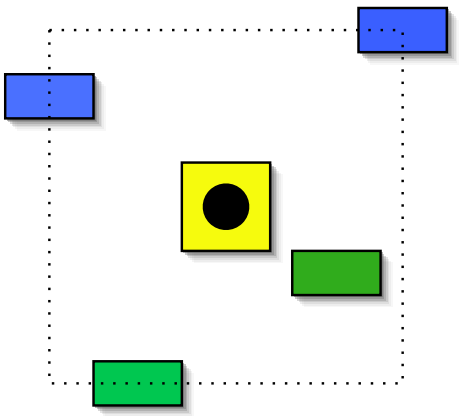


Figure 4.23: Initial Placement of the 3D-Vias in the centroid of the net bounding box.

To approach the proposed legalization problem we present a heuristic similar to the one described in (KHATKHATE, A. et al, 2004) called Tetris algorithm. It is described in detail in algorithm 5 and ilustrated by figure A.10.

Algorithm 5 Tetris algorithm to legalize Vias.

Input: A 3D-Via layer p, a set of 3D-Vias W such that W contains all 3D-Vias $w \mid t(w) = p$, a set of initial coordinates $(\forall w \in W)(xw_{ini}, yw_{ini})$, a set of bounding boxes Bw containing $B_{net(w)}$.

Output: A set of final positions (xw, yw) to replace the function $place$.

1: create a list L of all $w \in W$ sorted by xw_{ini};
2: slice the area in R rows such that the row height is h and $R = CircuitHeight/h$;
3: initialize a vector $StartingCoordinate(R)$ with 0 in all positions. The vector refers to the starting left coordinate of every row. In the beginning, the left limit is 0 so every coordinate is allowed;
4: **for** every 3D-Via $w \in W$
5: **for** every row $r \in R$
6: compute target x coordinate $x_r = max(StartingCoordinate[r], xw_{ini})$;
7: calculate cost of the movement $cost_r(w)$ as detailed in algorithm 2;
8: **end for**
9: select the row $BestRow \in R$ such that $(\forall_{r \in R})(cost_{BestRow} \leq cost_r)$;
10: set x_v to $x_{BestRow}$ and y_v as $h \times BestRow$
11: **end for**

A single movement from the original position to a legal position in a row receives a cost function that is related to the displacement from the original position and also to the wire length overhead imposed by the move. Every move to a position within the bounding box (as shown in figure A.10.c) do not impose any wire length penalty and will be preferred. The algorithm penalizes movements to positions outside the bounding boxes by computing the wl_{ovh} variable and by multiplying the cost by a constant *C*. In our experiments we set this constant to 3. The procedure for calculating the cost is given by algorithm 6, where the function $distance$ returns the distance of a point to the closest edge of a box.

Algorithm 6 Compute the cost of a 3D-Via movement.

Input: A via w with an associated B_w; a source coordinate (xw_{ini}, yw_{ini}) and a target coordinate $TargetPos = (x_r, y_r)$.

Output: The cost of the movement $cost_r(w)$.

1: $displacement = abs(xw_{ini} - x_r) + abs(yw_{ini} - y_r)$;
2: **if** $((x_r, y_r)$ is inside $B_w)$ **then**
3: $cost_r = displacement$;
4: **else**
5: $wl_{ovh} = distance(TargetPos, B_w)$;
6: $cost_r = C \times (displacement + wl_{ovh})$;
7: **end if**

4.9.3 Experimental Results

Our experimental results are divided into easy and harder instance experiments. For the easy set, we performed a min-cut partitioning (KARYPIS, G. et al, 1999) of the netlist

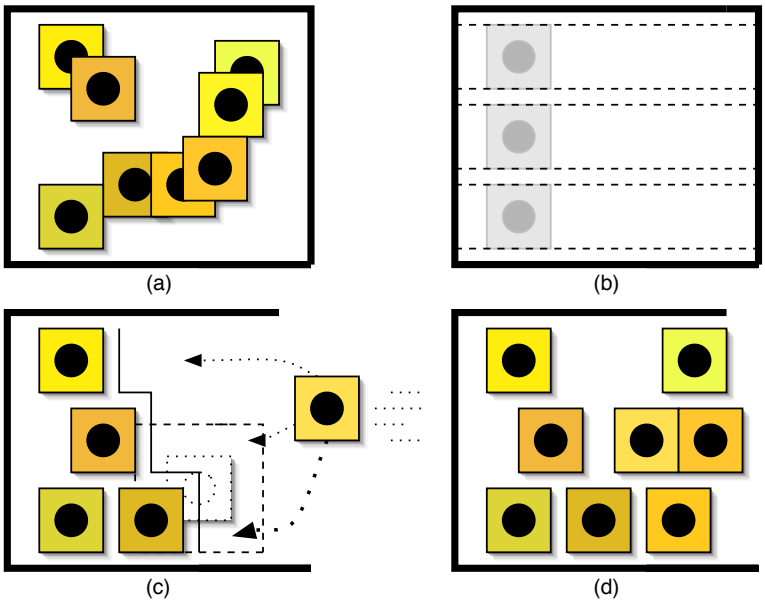


Figure 4.24: A Tetris algorithm for legalizing the 3D-Vias. (a) Represents an initial solution; (b) shows step 2 that slices the area into rows; (c) ilustrates the steps 4-10 which drops the cells into the circuit; (d) shows the final solution.

into two tiers and placed them constrained to maintain the z coordinate. This method will provide close to optimal 3D-Via count, producing an easy legalization instance. The second set of experiments leaves the placer free to insert 3D-Vias within an upper bound set to an acceptable value for total 3D-Via area. The placer will use this freedom to reduce wire length further. Tables 4.9.3, 4.9.3, 4.9.3 and 4.9.3 show the experimental results of the proposed algorithm on the easy (first two tables) and hard instances (last two tables).

The following observations are extracted from the data in the tables:

- Our placer is sensible to the 3D-Via pitch in order to use the 3D-Via resources. As the pitch increases, less 3D-Vias are used by the placer.
- Our algorithm couldn't legalize all 3D-Vias inside their bounding box for all instances. However, the wire length overhead is very low in general. For the easy instances it is less than 0.01% in most of the cases. In the hard instances it was under 0.1% in all cases with 3D-Via pitch of 5μm and 10μm, while with very large 3D-Vias (25μm) the overhead is larger but still less than 5% in most cases.
- The solutions with minimum 3D-Via count have less wire length overhead, but final wire length is worse in the cases of 5μm 3D-Via pitch.

The wl overhead averaged for all benchmarks in the hard mode was 0.05%, 0.23% and 4.34% for 5μm, 10μm and 25μm respectively. On run time, the proposed algorithm used 4.65s, 2.56s and 0.34s respectively. We also compared our algorithm with the one from (LIM, 2005). It obtained wl overhead of 0.05%, 0.16% and 2.78% while run time was 5391s, 1540s and 83.7s respectively. The excessive run time is caused mainly by the initial phase that tests all 3D-Vias in all available positions to decide which one is the best for it.

Table 4.18: Experimental results for our 3D-Via placement algorithm on easy instances for 5μm and 10μm pitch. Run times are measured in seconds.

3D-Via Pitch	5 μm					10 μm				
bench	# V	disp	# out	time	WL OVH	# V	disp	# out	time	WL OVH
ibm01	374	549	77	0.02	0.00%	374	1436	85	0.03	0.01%
ibm02	396	508	23	0.02	0.00%	396	1183	22	0.04	0.00%
ibm03	1064	1448	107	0.05	0.00%	1064	3716	117	0.12	0.00%
ibm04	735	1033	118	0.04	0.00%	735	2493	129	0.09	0.00%
ibm05	2258	3563	327	0.14	0.00%	2258	10067	366	0.28	0.01%
ibm06	1059	1468	56	0.06	0.00%	1059	3974	70	0.12	0.00%
ibm07	992	1380	89	0.06	0.00%	992	3310	91	0.15	0.00%
ibm08	1298	8467	213	0.09	0.02%	1298	23136	313	0.2	0.11%
ibm09	699	967	46	0.06	0.00%	699	2178	51	0.12	0.00%
ibm10	1490	2002	151	0.15	0.00%	1490	4839	172	0.32	0.00%
ibm11	1190	1650	93	0.11	0.00%	1190	3735	82	0.22	0.00%
ibm12	2293	3179	173	0.24	0.00%	2293	7386	176	0.51	0.00%
ibm13	1042	1511	150	0.1	0.00%	1042	3581	164	0.21	0.00%
ibm14	2121	2912	119	0.28	0.00%	2121	6626	126	0.59	0.00%
ibm15	3002	4045	162	0.38	0.00%	3002	10318	181	0.83	0.00%
ibm16	2102	2784	129	0.31	0.00%	2102	6100	131	0.67	0.00%
ibm17	2769	4090	242	0.45	0.00%	2769	9601	253	0.94	0.00%
ibm18	1676	2563	114	1.11	0.00%	1676	5669	120	0.55	0.00%
AVG:				0.20	0.00%				0.33	0.01%

We also compared the proposed algorithm with the work on (YAN, H. et al, 2005). Note that Yan et. al proposes a number of algorithms for the task. Initially they formulate the problem as an ILP (Integer Linear Programming). However, the problem size grows dramatically with addition of 3D-Vias which translates to infeasible number of variables on the IL problem. The authors present alternatives to reduce problem size. The more interesting approach is to perform first an exhaustive search of every 3D-Via into its optimal position; every 3D-Via placement with no conflict is performed. To solve conflicts, a small ILP problem is generated for the conflicting 3D-Vias and a reduced number of positions. This method was implemented on Z-Place in order to establish a comparison between it and the proposed 3D-Via legalization approach. Experimental results for the ILP solution for the hard instances are presented in table 4.22.

In average, the proposed algorithm achieve similar quality compared to the ILP algorithm with 2 orders of magnitude advantage on run time. Figures 4.25 and 4.26 present 3D-Via placements produced by the proposed algorithm and the ILP approach respectively. A zoom on figure 4.25, presented in figure 4.27 demonstrates that the proposed method is not constrained to a grid which facilitates the wire length improvement.

4.9.4 Removing and avoiding overlaps between cells and 3D-Vias

Under Z-Place flow, the global phase provides a reasonable distribution of cells and 3D-Vias such that they fit into their assigned tier. However, the current version of the tool do not remove the overlaps between cells and 3D-Vias that occupy active area. It is possible to handle this problem without much novelty using existing techniques. The

Table 4.19: Experimental results for our 3D-Via placement algorithm on easy instances for 25μm

3D-Via pitch:	25 μm				
bench	# V	disp	# out	time	WL OVH
ibm01	374	4967	93	0.01	0.02%
ibm02	396	6005	44	0.01	0.01%
ibm03	1064	35128	182	0.04	0.04%
ibm04	735	12628	168	0.04	0.02%
ibm05	2258	125801	564	0.12	0.13%
ibm06	1059	47131	121	0.05	0.03%
ibm07	992	21368	149	0.06	0.01%
ibm08	1298	76442	339	0.08	0.42%
ibm09	699	14969	81	0.05	0.00%
ibm10	1490	23324	213	0.12	0.00%
ibm11	1190	25258	125	0.09	0.00%
ibm12	2293	50069	254	0.21	0.01%
ibm13	1042	14839	201	0.08	0.00%
ibm14	2121	27557	193	0.25	0.00%
ibm15	3002	92530	232	0.34	0.00%
ibm16	2102	26730	162	0.27	0.00%
ibm17	2769	35213	269	0.38	0.00%
ibm18	1676	16937	135	0.22	0.00%
AVG:				0.12	0.05%

suggested solutions for this problem are enumerated bellow:

1. Place cells together with 3D-Vias in the detailed placement phase. A similar procedure is done by (KAYA, I. et al, 2004) assuming that 3D-Vias height is less or equal the Standard Cell row height.

2. Place cells together with 3D-Vias with mixed size placement techniques. This would be the best solution, since it would be able to handle big 3D-Vias (higher than a row) and small ones fitting them together in a same row, if possible.

3. Legalize the solution after the whole process. This method is the easier one but should lead to bad results specially in the case of small 3D-Vias, since the 3D-Via count would be high.

The algorithm 7 computes the 3D-Via layer that contains the 3D-Vias occupying active area for a given tier. Please refer to section 4.5.6 for the details on the formulation. Note that I denotes the integration strategy of a 3D-Via layer. The algorithm computes all tiers i orientation $To[i] \in \{DOWN, UP\}$ and via layers, returning the via layer correspondent to the input tier. It also uses a function $flip(i)$ that returns the opposite orientation (e.g. $DOWN$ if $To[i] = UP$ a and UP if $To[i] = DOWN$).

4.9.5 Conclusions

We presented an algorithm for the placement and legalization of 3D-Vias on a 3D-Circuit. We observed that the ideal placement of such objects is inside the bounding box

Table 4.20: Experimental results for our 3D-Via placement algorithm on hard instances for 5μm and 10μm pitch.

3D-Via Pitch	5 μm					10 μm				
bench	# V	disp	# out	time	WL OVH	# V	disp	# out	time	WL OVH
ibm01	5375	8809	849	0.25	0.08%	5375	36069	1236	0.15	0.28%
ibm02	7880	12825	984	0.41	0.04%	7880	78254	1254	0.23	0.23%
ibm03	10282	17354	1880	0.6	0.07%	10282	91116	2899	0.34	0.37%
ibm04	11721	18750	1744	0.74	0.05%	11721	87716	2579	0.43	0.21%
ibm05	10544	17296	1585	0.69	0.03%	10544	93069	2065	0.37	0.11%
ibm06	13088	21980	2304	0.79	0.07%	13210	134913	3557	0.45	0.39%
ibm07	19099	31022	2833	1.51	0.06%	19099	161804	3854	0.85	0.25%
ibm08	16659	26590	2675	1.34	0.05%	16659	119381	3733	0.74	0.18%
ibm09	24598	41715	3668	2.15	0.06%	24598	265514	6070	1.22	0.38%
ibm10	32573	52750	4232	3.65	0.04%	32573	237096	5999	1.99	0.15%
ibm11	30987	51465	4405	3.08	0.05%	30987	302353	6190	1.73	0.25%
ibm12	35192	57523	3696	4.06	0.03%	35192	301419	4780	2.18	0.14%
ibm13	35175	59633	4888	3.8	0.05%	35175	371957	8073	2.14	0.28%
ibm14	58438	93185	8131	8.57	0.04%	58438	461453	9976	4.69	0.16%
ibm15	69229	116074	9694	10.09	0.04%	69075	790675	15782	5.72	0.26%
ibm16	78242	128163	9576	13.07	0.03%	78242	665982	12689	7.14	0.15%
ibm17	90680	148705	10027	16.49	0.03%	90680	798764	14540	8.92	0.11%
ibm18	72456	114284	10401	12.46	0.04%	72456	549949	14863	6.77	0.15%
AVG:				4.65	0.05%				2.56	0.23%

of the net that it is connected to, since the wire length overhead is null. Our heuristic was not able to place all 3D-Vias inside their bounding boxes, which is expected since the net bounding boxes are short (after cell placement). On the other hand, the algorithm could accommodate the 3D-Vias in such a way that wire length overhead is very low, validating the 3D-Via legalization methodology and leaving space for more research on 3D placement. Compared to an existing approach, it obtains similar results with orders of magnitude advantage on run time.

4.10 Whole Z-Place Experimental Results

This section presents experimental results that include all global, detailed and 3D-Via placement algorithms, establishing comparisons with other 3D and 2D solutions. First, a run time contribution for each step of Z-Place flow was also evaluated. The results, illustrated in figure 4.28, report run time for the reading of the netlist (netlist), global placement (global), legalization of global placement (legalize1), threshold accept improvement (TA), legalization of detailed placement (legalize2), greedy improvement (greedy), 3D-Via placement (3D-Via), other intermediate steps.

Table 4.23 presents experimental results targeting a 2 tier face-to-face technology with 1μm 3D-Via pitch and 20μm 3D-Via length. This technology was chosen to be the one with no Through Via and less cost compared to solutions with more tiers; it also demonstrated very good potential on the global placement analysis (section A.3.1.3). The table

Table 4.21: Experimental results for our 3D-Via placement algorithm on hard instances for 25μm

3D-Via pitch:	25 μm				
bench	# V	disp	# out	time	WL OVH
ibm01	1550	186374	1097	0.02	5.68%
ibm02	1945	275265	921	0.03	2.48%
ibm03	2391	282265	1149	0.03	2.70%
ibm04	3045	368423	1637	0.06	3.27%
ibm05	3442	655449	2067	0.06	5.21%
ibm06	2612	333387	1374	0.04	2.18%
ibm07	4627	898320	2786	0.1	6.02%
ibm08	4735	714555	2594	0.1	3.55%
ibm09	5512	637455	2720	0.13	3.01%
ibm10	9044	1191727	4700	0.27	2.83%
ibm11	7088	1048209	4203	0.2	3.16%
ibm12	9766	1147941	4251	0.27	1.93%
ibm13	8341	1391079	5040	0.24	3.65%
ibm14	15468	2400982	8561	0.6	3.06%
ibm15	15070	5481844	10434	0.6	8.80%
ibm16	20155	2829965	9746	0.87	2.47%
ibm17	23856	4263605	11261	1.12	3.25%
ibm18	21445	9065612	13464	0.96	14.92%
AVG:				0.32	4.34%

presents the 2D Area used, the final wire length on 3D, a flattened wire length (ignores the length of vertical wires) and run time.

Table 4.24 establishes a comparison of Z-Place to other methods. First, the algorithm from (LIU, G. et al, 2005) with Z-Place; since Liu uses flattened wire length, we compared with Z-Place flattened wire length. Secondly, we perform comparisons with 2D solutions such as Fastplace, Fastplace using a additional improvement tool called Domino and Z-Place targeting 1 tier. The comparisons with 2D tools consider the Z-Place 3D wire length.

Table 4.25 presents results targeting a 3 tiers face-to-face and face-to-back technology with 1μm face-to-face 3D-Via pitch and 5μm face-to-back 3D-Via pitch. 3D-Via lengths are 20μm and 15μm for face-to-face and face-to-back respectively. Table 4.25 estabilishes comparisons with 2D solutions.

The following can be observed and concluded from the tables:

- In average there is a wire length improvement in the order of 10% to add one tier.
- On 2D, Fastplace and Z-Place have similar WL results.
- The aid of an specialized detailed placer such as Domino can improve significantly the quality of the results.
- Compared to existing algorithms, we have a small average advantage (2%); on the other hand, the compared algorithm supports only 2 tiers face-to-face.

The 2 tier technology have the best configuration considering cost and reliability; it is able to provide at least 10% WL improvement. We observe that more improvement could

Table 4.22: Experimental results obtained by running the ILP algorithm for 3D-Via placement.

3D-Via Pitch	5 μm			10 μm			25 μm		
bench	# V	CPU	WL OVH	# V	CPU	WL OVH	# V	CPU	WL OVH
ibm01	5630	32	0.10%	5630	13	0.29%	1550	2	7.06%
ibm02	8045	55	0.05%	8045	23	0.15%	1945	3	2.70%
ibm03	10023	85	0.07%	10023	35	0.22%	2391	4	1.30%
ibm04	11680	123	0.06%	11680	48	0.17%	3045	6	1.45%
ibm05	9819	96	0.03%	9819	41	0.08%	3442	6	2.45%
ibm06	13053	129	0.08%	13131	47	0.24%	2612	7	2.11%
ibm07	19218	329	0.06%	19218	126	0.18%	4627	12	5.63%
ibm08	16552	274	0.05%	16552	115	0.13%	4735	16	3.09%
ibm09	24760	525	0.06%	24760	190	0.21%	5512	16	2.15%
ibm10	32714	1013	0.04%	32714	381	0.11%	9044	30	2.08%
ibm11	31096	811	0.05%	31096	277	0.18%	7088	21	1.73%
ibm12	35285	1229	0.03%	35285	427	0.11%	9766	35	1.27%
ibm13	35478	1106	0.05%	35478	386	0.18%	8341	31	2.15%
ibm14	58514	3085	0.04%	58514	1083	0.12%	15468	81	2.14%
ibm15	69030	3794	0.05%	69206	1237	0.15%	15070	84	5.83%
ibm16	77904	5687	0.04%	77904	1907	0.11%	20155	142	1.56%
ibm17	90378	7643	0.03%	90378	2552	0.08%	23856	161	2.02%
ibm18	72562	4963	0.05%	72562	1844	0.12%	21445	149	5.72%
		1721	0.05%		596	0.16%		45	2.91%

be approached with the implementation of published techniques (such as (SPINDLER; JOHANNES, 2006)) to the placer; the research on placement area is very vast and the combination of existing methods to the Z-Place have a substantial improvement potential.

In general, the experimental results demonstrate a trade-off between WL and actual dolar cost of the circuit, since the addition of more tiers improve WL but introduce reliability problems and increase the cost. If one is able to pay the price to improve performance, a 3D solution is a simple and effective choice.

4.11 Critical Paths Handling

In the circuit netlist, each net has a set of weights that determines the criticality of each sink, defined by a pair ($driver$,$sink_i$) for all sinks i in the net. The criteria of criticality could be either defined by timing analysis, power analysis or any combination of these methods. Both detailed and global placement algorithms described in this text can handle point-to-point weights naturally. On Quadratic Placement the multi-pin net is broken into point-to-point connections that have their own separate weight. Those weights can be incorporated into the star model simply by setting both the weights from the driver to the star node and the star to the critical sink; in the clique model, the weight can be directly incorporated into the direct connection from the driver to the critical sink. On the detailed placement with Threshold Accept, the weight can be incorporated by computing a separate wire length for the critical wires as done in (SWARTZ; SECHEN,

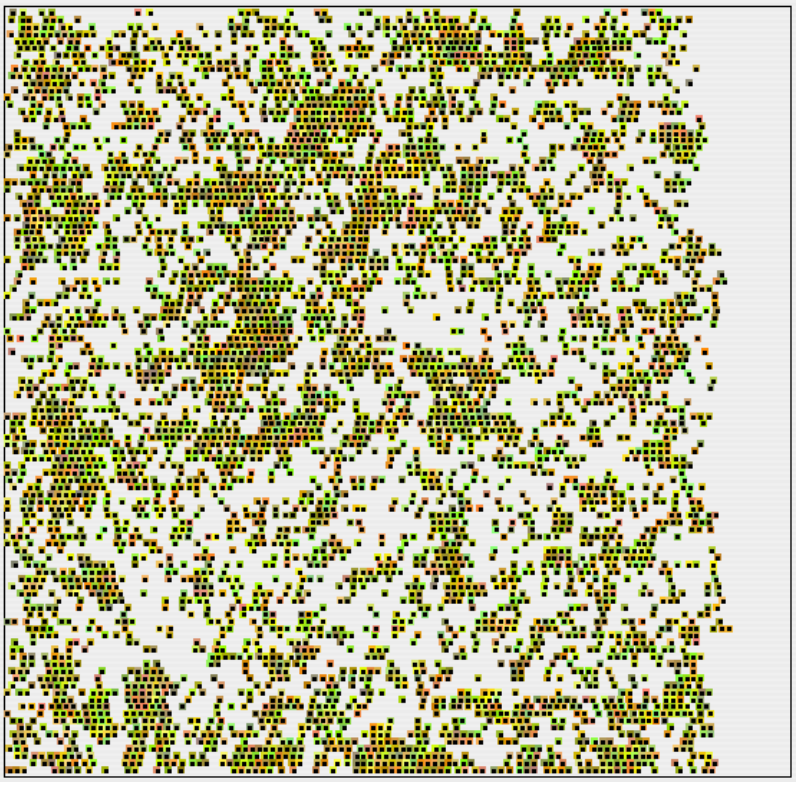


Figure 4.25: An example of a 3D-Via Placement with the proposed Tetris algorithm for ibm01 with 5375 3D-Vias

1995), assuming that this connection will be actually routed separately. The extra weight is multiplied only on the critical connection. The fact that a critical connection is modeled in the placement level similarly to its actual routing topology improves the capability of the whole flow to converge to a better solution (SANTOS, 2006).

Since net and point-to-point weighting on the placement level are very mature technique to improve the length of critical wires, it is not a subject of this thesis. In this work, we exploit the problem of keeping critical paths with no 3D-Via connections. There are several issues on 3D-Vias that would make them not attractive to critical wires:

- A connection from a transistor to a 3D-Via require extra wires and vias going though all the metal layers. This scenario impose significant capacitance and resistance for the wire;
- The electrical characteristics of the 3D-Via itself could be harmful.
- Existing timing-closure methodologies assume that the timing critical connections can be modeled by 2D Steiner Trees.

In order to address the proposed problem, we define the 3D-Via avoidance problem formally in section 4.11.1 and we propose an algorithm that will work under Z-Place and other tools based on force directed placement on section 4.11.2. Experimental results are presented on section 4.11.3 and partial conclusions on section 4.11.4.

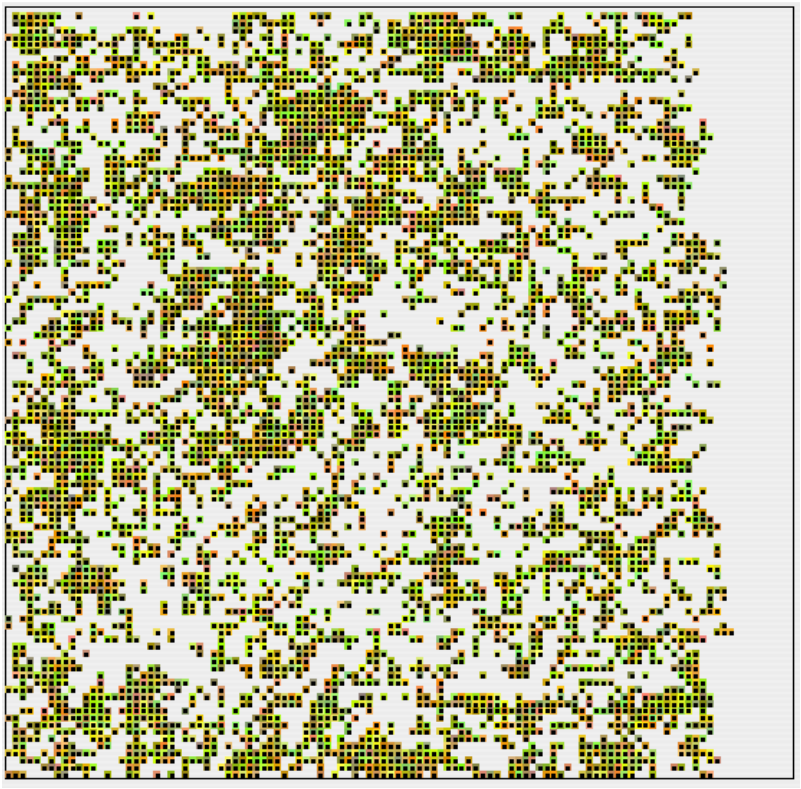


Figure 4.26: An example of a 3D-Via Placement with ILP algorithm for ibm01 with 5375 3D-Vias

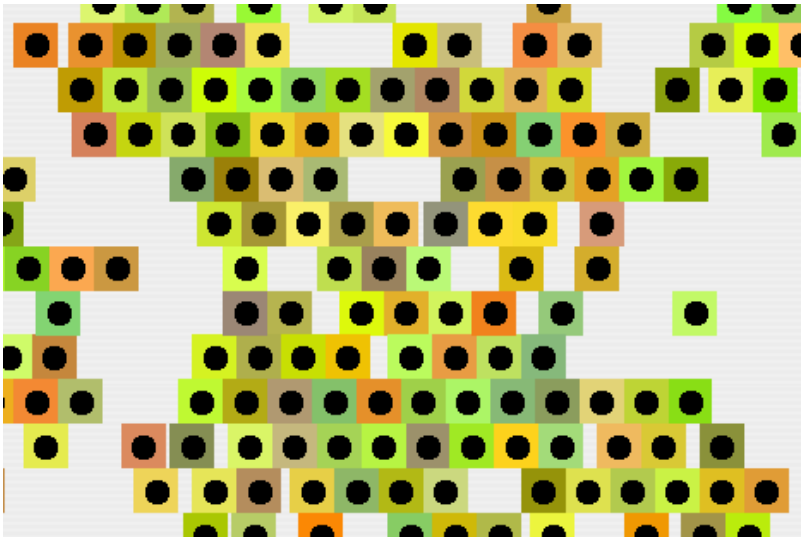


Figure 4.27: A detail on the 3D-Via placement obtained by the Tetris algorithm

4.11.1 Problem Definition

Initially, let us define the required information and how it can be obtained. The approach starts by identifying the k most critical paths of the circuit and defining a function

Algorithm 7 Algorithm to compute the via layer of a given tier t.

Input: A tier index t.
Output: The correspondent via layer index l.

1: **if** $I[0]$ = $b2b$ **then**
2: $To[0]$ = $DOWN$;
3: **else**
4: $To[0]$ = UP;
5: **end if**
6: **for** $i = 1$ to $numTiers - 1$
7: **if** $I[i-1]$ = $f2b$ **then**
8: $To[i]$ = $To[i-1]$
9: **else**
10: $To[i]$ = $flip(To[i-1])$
11: **end if**
12: **end for**
13: Initialize a $tier2ViaLayer$ vector with $numTiers - 1$ positions.
14: Initialize all positions of $tier2ViaLayer$ with -1
15: $numViaLayers$ = $numTiers - 1$
16: **for** $l = 0$ to $numViaLayers - 1$
17: $orTop$ = $To[l]$
18: $orBottom$ = $To[l-1]$
19: **if** $orBottom$ = $DOWN$ **then**
20: $tier2ViaLayer[l]$ = l;
21: **end if**
22: **if** $orTop$ = TOP **then**
23: $tier2ViaLayer[l+1]$ = l;
24: **end if**
25: **end for**
26: **return** $tier2ViaLayer[t]$;

$cp \in C \times K$ that maps every cell to a critical path, where C is the set of cells to be placed and K is the set of numbers $\{1, 2, ...k\}$. The cell placement problem takes the set C and place each cell $c \in C$ into a (c_x,c_y,c_z) coordinate inside a sliced placement cube model with t tiers such that $c_z \in \{1, 2.., t\}$.

The **3D-Via avoidance** problem is defined as a constraint to the cell placement process defined by equation 4.29.

$$\forall_{i,j \in C}(cp(i) = cp(j) \rightarrow i_z = j_z) \tag{4.29}$$

4.11.2 Proposed Algorithm

The algorithm for 3D-Via avoidance is applied into the global placement phase, more specifically on the Quadratic Placement engine. The other steps performed after Quadratic Placement (Iterative Refinement (section 4.6.2), Legalization and Threshold Accept improvement (section 4.8.2) do not move critical cells into the Z axis).

In order to keep the cells of a same critical path together, the netlist is updated by inserting an artificial node called $criticalstar$ ($cs_i \mid i \in K$) for each critical path. Every cell

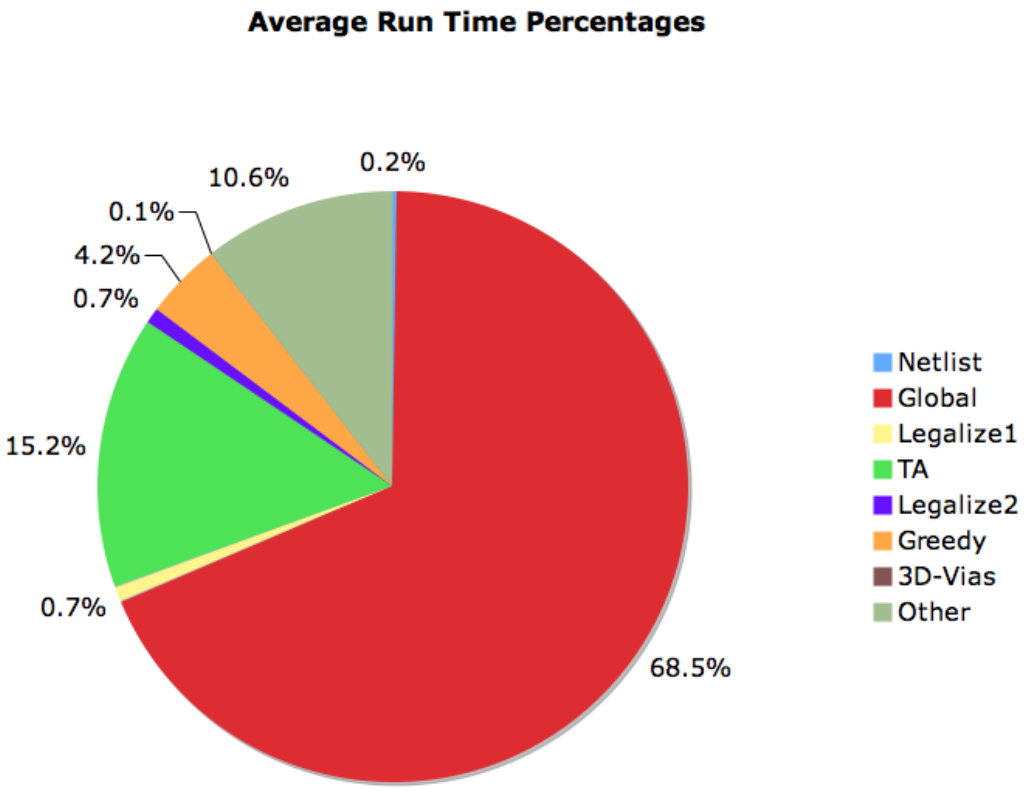


Figure 4.28: Run time contribution of Z-Place steps averaged from ibm01 to ibm12.

$c \in C$ such that $cp(c) = i$ is connected to cs_i with a special kind of artificial connection called *Z-Grouping*.

A *Z-Grouping* connection has a distinct weight for the Z axis in contrast with the weight on X or Y. In fact, the weight on Z axis is set to a very high number, such as 500 times a regular wire, while on the X and Y it could be set to 0 or some small number (such as 0.5 times a regular wire) in order to address the critical wire length reduction problem as well. Note that that for the 3D-Via avoidance problem the weight on X and Y can be set to any value, but it might affect wire length and the number of 3D-Vias. In order to keep the wire length and number of 3D-Vias as close as possible to the number without the critical 3D-Via avoidance, the weight on X and Y axes are set to smaller values.

Figure A.9 illustrated the method and the desired effect.

4.11.3 Experimental Results

In order to verify the effectiveness of the proposed algorithm, the following experimental setup was proposed. First, a benchmark set with timing information was generated with the aid of commercial tools to synthesize the source VHDL description and map it to a logic level netlist. A timing analyzer for the same commercial tool set was used to identify the 100 most critical paths at the logic level. The details of the generated benchmarks can be observed in table 4.2.

Initially, a baseline execution of Z-Place did not include any Z-Grouping net. After that, 100 Z-Grouping nets were introduced with 0.5 weight on X and Y axes and 500 on

Table 4.23: Experimental results for 2 tiers face-to-face. Run times are measure in seconds.

	2D area	WL	WL Flattened	run time (s)
ibm01	1209856	1.61E+06	1.50E+06	183
ibm02	1517568	3.25E+06	3.10E+06	425
ibm03	1865920	4.20E+06	4.00E+06	375
ibm04	2377280	5.05E+06	4.81E+06	495
ibm05	2483712	7.64E+06	7.44E+06	740
ibm06	2038784	4.98E+06	4.73E+06	601
ibm07	3612960	7.72E+06	7.33E+06	1071
ibm08	3697920	8.88E+06	8.54E+06	1409
ibm09	4307568	9.62E+06	9.13E+06	1463
ibm10	7064640	1.63E+07	1.57E+07	2115
ibm11	5536320	1.42E+07	1.36E+07	1928
ibm12	7627968	2.02E+07	1.95E+07	2150
ibm13	6515184	1.66E+07	1.58E+07	2452
ibm14	12082176	3.16E+07	3.04E+07	5985
ibm15	11771712	3.68E+07	3.54E+07	7650
ibm16	15744768	4.56E+07	4.40E+07	10292
ibm17	18636320	5.86E+07	5.68E+07	10644
ibm18	16751808	3.99E+07	3.84E+07	11971

Table 4.24: Comparison of Lim (2 tiers), Fastplace (1 tier), Fastplace + Domino (1 tier), Z-Place (1 tier) with Z-Place in 2 tiers face-to-face.

	Liu (flattened)	Imp.	Fastplace	Imp.	Fastplace + Domino	Imp.	Z-Place (1 tier)	Imp
ibm01	1.57E+06	5%	1.89E+06	17%	1.70E+06	5%	1.86E+06	15%
ibm02	3.36E+06	9%	3.93E+06	21%	3.69E+06	13%	3.90E+06	20%
ibm03	4.22E+06	5%	5.27E+06	25%	4.95E+06	18%	5.32E+06	27%
ibm04	5.31E+06	10%	6.15E+06	22%	5.79E+06	15%	6.34E+06	26%
ibm05	8.24E+06	11%	1.06E+07	39%	1.03E+07	35%	1.03E+07	35%
ibm06	4.68E+06	-1%	5.41E+06	9%	5.04E+06	1%	5.41E+06	9%
ibm07	8.11E+06	11%	9.10E+06	18%	8.64E+06	12%	9.32E+06	21%
ibm08	8.38E+06	-2%	9.80E+06	10%	9.34E+06	5%	9.88E+06	11%
ibm09	8.72E+06	-5%	1.08E+07	12%	1.02E+07	6%	1.11E+07	15%
ibm10	1.60E+07	2%	1.90E+07	16%	1.81E+07	11%	2.00E+07	23%
ibm11	1.27E+07	-7%	1.55E+07	9%	1.46E+07	2%	1.61E+07	13%
ibm12	2.12E+07	9%	2.46E+07	22%	2.35E+07	16%	2.49E+07	23%
ibm13	1.54E+07	-3%	1.89E+07	14%	1.77E+07	7%	1.98E+07	19%
ibm14	2.91E+07	-4%	3.57E+07	13%	3.39E+07	7%	3.67E+07	16%
ibm15	3.45E+07	-3%	4.44E+07	21%	4.22E+07	15%	4.55E+07	24%
ibm16	4.05E+07	-8%	4.69E+07	3%	4.43E+07	-3%	5.02E+07	10%
ibm17	—	—	6.74E+07	15%	6.44E+07	10%	7.17E+07	22%
ibm18	—	—	6.74E+07	69%	4.36E+07	9%	4.91E+07	23%
AVG:		2%		20%		10%		20%

Table 4.25: Experimental results for 3 tiers face-to-face. Run times are measure in seconds.

	2D area	WL Vias	WL Flattened	run time
ibm01	817152	1.60E+06	1.48E+06	188
ibm02	1040256	2.79E+06	2.57E+06	533
ibm03	1272320	3.87E+06	3.63E+06	419
ibm04	1596192	4.78E+06	4.48E+06	631
ibm05	1665280	6.61E+06	6.33E+06	863
ibm06	1368576	4.52E+06	4.19E+06	730
ibm07	2420736	7.37E+06	6.87E+06	1171
ibm08	2479008	7.39E+06	6.91E+06	1460
ibm09	2960048	8.72E+06	8.11E+06	1352
ibm10	4791552	1.51E+07	1.44E+07	2073
ibm11	3798432	1.29E+07	1.21E+07	2328
ibm12	5175264	1.76E+07	1.67E+07	2412
ibm13	4359680	1.57E+07	1.48E+07	3070
ibm14	8050944	2.90E+07	2.75E+07	6908
ibm15	7963648	3.61E+07	3.42E+07	8539
ibm16	10426272	4.35E+07	4.15E+07	11986
ibm17	12383136	5.53E+07	5.30E+07	13365
ibm18	11312752	3.90E+07	3.70E+07	15271

Table 4.26: Comparison of Fastplace (1 tier), Fastplace + Domino (1 tier), Z-Place (1 tier) with Z-Place in 3 tiers face-to-face / face-to-back. Run times are measure in seconds.

	Fastplace	Imp	Fastplace + Domino	Imp	Z-Place 1 (1 tier)	Imp
ibm01	1.89E+06	18%	1.70E+06	6%	1.86E+06	16%
ibm02	3.93E+06	41%	3.69E+06	32%	3.90E+06	40%
ibm03	5.27E+06	36%	4.95E+06	28%	5.32E+06	37%
ibm04	6.15E+06	29%	5.79E+06	21%	6.34E+06	33%
ibm05	1.06E+07	60%	1.03E+07	56%	1.03E+07	56%
ibm06	5.41E+06	20%	5.04E+06	12%	5.41E+06	20%
ibm07	9.10E+06	24%	8.64E+06	17%	9.32E+06	27%
ibm08	9.80E+06	33%	9.34E+06	26%	9.88E+06	34%
ibm09	1.08E+07	24%	1.02E+07	17%	1.11E+07	27%
ibm10	1.90E+07	25%	1.81E+07	20%	2.00E+07	33%
ibm11	1.55E+07	21%	1.46E+07	13%	1.61E+07	26%
ibm12	2.46E+07	40%	2.35E+07	34%	2.49E+07	42%
ibm13	1.89E+07	20%	1.77E+07	13%	1.98E+07	26%
ibm14	3.57E+07	23%	3.39E+07	17%	3.67E+07	27%
ibm15	4.44E+07	23%	4.22E+07	17%	4.55E+07	26%
ibm16	4.69E+07	8%	4.43E+07	2%	5.02E+07	15%
ibm17	6.74E+07	22%	6.44E+07	16%	7.17E+07	30%
ibm18	6.74E+07	73%	4.36E+07	12%	4.91E+07	26%
AVG:		30%		20%		30%

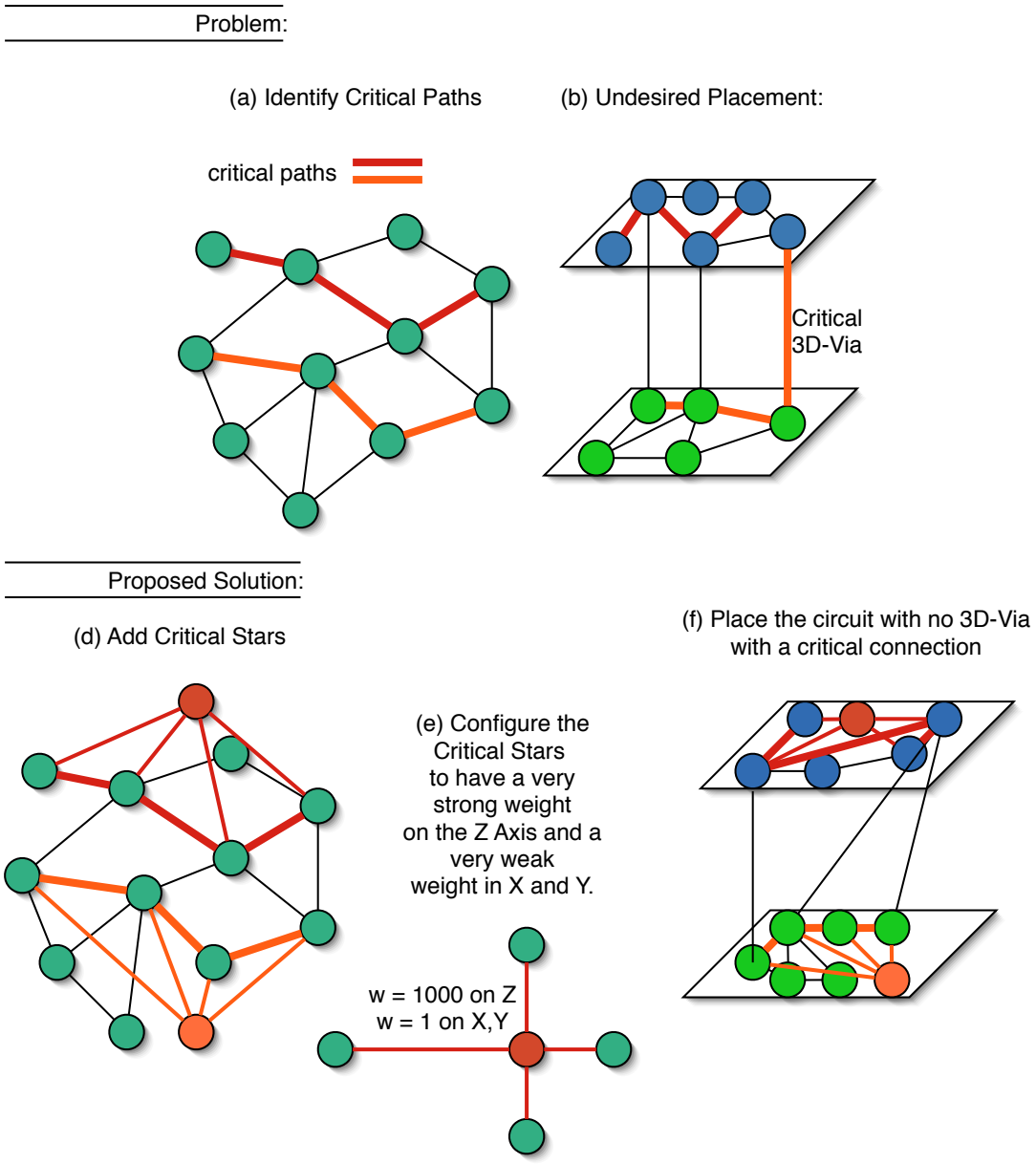


Figure 4.29: An illustration of the proposed method for avoiding 3D-Vias with critical connections.

the Z. Table 4.11.3 contains the experimental results.

Analyzing the data presented on table 4.11.3 the following data can be observed:

- The proposed method is very effective to avoid critical 3D-Vias; all experimental results (except for the circuit b13) resulted in none critical 3D-Via (see column #CV).
- The circuit b13 is a small benchmark (275 cells). Note that smaller benchmarks are more constrained than larger benchmarks. On our experiments, 100 critical paths might be too much for this circuit, making the problem unfeasible.
- Wire length is not affected (0.03% affect in average).

Table 4.27: Baseline experimental results on benchmarks with timing information; # t denotes number of tiers, # V denotes 3D-Vias count, # CV denotes critical 3D-Vias count, C WL denotes critical wire length

		Baseline				With Z-Grouping Nets			
	# t	# V	# CV	C WL	WL	# V	# CV	C WL	WL
b11	2	206	544	4727	49843	225	0	4520	48540
b11	3	413	1269	3640	42967	407	0	4207	42373
b11	4	385	716	3286	41113	405	0	4312	40395
b12	2	461	299	2698	78557	460	0	3290	77013
b12	3	501	175	3438	80853	474	0	4456	83664
b12	4	395	255	3674	76486	364	0	3015	73409
b13	2	88	29	2673	17758	68	0	3636	18635
b13	3	149	84	2662	17559	143	34	2529	17255
b13	4	110	91	2415	16344	125	17	2637	17300
b14	2	1384	1065	8299	379860	1451	0	9734	372845
b14	3	2020	2307	6096	328408	2186	0	6753	321635
b14	4	1472	1167	7425	348034	1913	0	7276	337371
b15	2	3110	687	5807	1.04E+06	3143	0	6059	993602
b15	3	4142	881	6035	946681	4291	0	7067	917239
b15	4	3314	981	6967	901196	3959	0	4679	819516
b17	2	9061	989	16941	3.71E+06	8894	0	16769	3.64E+06
b17	3	12586	1550	13329	3.34E+06	12260	0	12824	3.23E+06
b17	4	10457	1172	10640	3.19E+06	10996	0	12825	2.93E+06
b20	2	3221	1925	11725	912468	3275	0	13264	948918
b20	3	4605	2272	9300	816211	4234	0	11921	913147
b20	4	3980	1930	8993	825921	4215	0	12705	958814
b21	2	2582	809	19036	1.13E+06	2667	0	18979	1.17E+06
b21	3	4692	1851	12451	839501	4543	0	13577	906789
b21	4	3178	1181	15783	867797	4185	0	17262	832459
b22	2	4473	1316	14948	1.60E+06	4380	0	14769	1.58E+06
b22	3	6895	1830	15512	1.32E+06	6224	0	15026	1.37E+06
b22	4	5373	1408	16086	1.30E+06	5240	0	13823	1.31E+06
b14-1	2	1575	848	8277	362468	1567	0	8857	364230
b14-1	3	2078	1142	8670	353513	2356	0	6892	305643
b14-1	4	1443	1049	7118	324952	1815	0	7061	332907
b15-1	2	2489	1365	7788	1.24E+06	2411	0	5492	1.12E+06
b15-1	3	4452	1011	4309	868396	4611	0	5804	842928
b15-1	4	3859	1469	3151	820633	3858	0	4372	832314
b17-1	2	9408	1180	16738	3.38E+06	9125	0	17356	3.29E+06
b17-1	3	13351	1229	17150	3.27E+06	13250	0	15913	2.97E+06
b17-1	4	10682	1135	16843	3.09E+06	11979	0	19531	2.83E+06
b20-1	2	3081	1383	13346	1.03E+06	3104	0	14039	1.04E+06
b20-1	3	4703	1677	10699	863206	3938	0	14689	1.02E+06
b20-1	4	3510	1333	12727	887554	3480	0	13654	879506
b21-1	2	3153	1274	11780	995118	3199	0	11096	956224
b21-1	3	4307	1844	10191	895383	4033	0	11220	954954
b21-1	4	3314	912	11499	862450	3486	0	11711	871389
b22-1	2	4571	1449	16277	1.50E+06	4952	0	17434	1.54E+06
b22-1	3	7394	2182	12864	1.33E+06	7210	0	12990	1.36E+06
b22-1	4	5546	1560	15233	1.32E+06	5413	0	16693	1.46E+06

- Number of 3D-Vias is only 2.36% affected in average.
- Critical wire length is improved by 4.6%, possibly due to the weight on X and Y axes.

4.11.4 Partial Conclusions

This section proposes a method to avoid the critical 3D-Vias simply by adding artificial nets in the circuit and making them with a strong weight only for the Z axis. The method could reduce the number of 3D-Via to zero in all tested cases (except for one small circuit) with no effect to circuit wire length and number of 3D-Vias. We conclude that the method explores a degree of freedom existing in the placement problem, where many solutions are equivalently good in terms of wire length and other classical metrics. Our algorithm actually introduced a new metric by the number of critical 3D-Vias and demonstrated that this objective does not conflict with 3D wire length optimization.

5 FAST AND EFFICIENT MAZE ROUTING STEINER TREES

5.1 Introduction

The task of the routing stage is to provide routes that connect circuit elements. In the previous chapter, placement algorithms were proposed for the reduction of the distance of the elements to be connected. In the routing stage, the objects are fixed and the routing task it to provide the shortest possible route that connects the circuit elements. In the presence of delay critical objects, the wire topology can be improved to trade wire length for delay.

Traditionally, the routing stage is performed at the end of the synthesis flow, just after placement is complete. Since the wires represent a significant amount of delay and power in a design, the wire lengths must be estimated during the whole synthesis flow in order to target a feasible implementation under pre-stabilished design constraints. In the upper design levels, such as system level, wires are estimated using inaccurate techniques since the circuit information is very preliminary. As the hardware synthesis advances, new information is provided for estimation. Convergence of the optimization process will be affected by the quality of the wiring estimates as well as placement algorithms, routing algorithms and their ability to optimize delay to the critical elements of the circuit while keeping the overall design routable. Convergence will also be affected based upon how well estimates made in the early stages match values obtained finally in the lower levels.

Although wire estimation seeks similarity with actual wires, higher level routing estimation and actual routing stages are performed with different algorithms.

Steiner Trees can be used for early estimation because, as they account for the individual positions of each pin and represent actual connected sets of paths, they are better than simple half-perimeter measures. Steiner Tree algorithms are also commonly used in Global Routing to actually define global routing solutions for each net. In particular, minimum length rectilinear Steiner trees (MRSTs) represent optimal routes in terms of wire length. However, wiring topologies strongly affect delay to the critical sinks and improperly designed MSRTs could lead to very high delays (CONG; LEUNG; ZHOU, 1993). Even small changes in topology may significantly affect delay to sensitive sinks. Additionally, most Steiner tree algorithms are not flexible enough to handle blockages, congestion and actual routing constraints.

The routing algorithms are flexible and robust enough to handle constraints such a blockages, congestion or other real design issues. The most commonly used algorithms for routing are based on maze routing (LEE, 1961). They are well known and are characterized by the following: (a) high flexibility since it provides a cost function that can

model any constraint, (b) producing optimal point-to-point paths and (c) good tree topologies. Those properties are obtained at the expense of high CPU and memory usage. By judicious implementation techniques and proper exploitation of the methods outlined in this thesis (including existing and new techniques), it is possible to improve the CPU times of these routers considerably without compromising their quality.

Instead of the usual distinction of Steiner trees and global routing algorithms, we understand that a unique flexible algorithm could do a better job delivering more accuracy to the estimates. All these stages have their own requirements for accuracy, route constraints, cost/penalty functions and run time. The algorithm must be flexible to attend the requirements of every stage, achieve good wire length and have capability to improve timing for selected elements. Run time should also be a concern; the run time requirement of early design stages is tighter than the requirement of the later stages such as detailed routing.

The Maze Router algorithm, discussed above, works over a routing space modeled as a graph with costs and constraints. In the scope of this thesis, we propose techniques to be incorporated into a Maze Router that will deliver good Steiner tree topologies according to the needs of each net. In order to extend this algorithm to a full circuit, the nets need to be routed one-by-one on a certain ordering; nets that were previously routed are fixed and considered as obstacles. In the literature there are many techniques that are able to handle ordering issues and rip-up and re-route of nets (FLACH; HENTSCHKE; REIS, 2004). Those techniques are not subject of this thesis.

Note also that the flexibility provided by Maze Routers to operate in any graph makes it suitable to work on 2D or 3D circuit with only a few changes to correctly model the 3D-Vias.

This chapter describes the AMAZE algorithm that serves the purpose of Steiner tree routing. Initially, we formulate the problem we are addressing, which considers only one net with critical and non-critical sinks on section 5.2. In order to understand the problem, we review the basics of Steiner tree with respect to delay and wire length optimization in section 5.3. A review of existing algorithms for Steiner tree construction is presented in section 5.4. Those algorithms include a basic study of path search methods that are used by Maze Routers building a theoretical basis to understand the AMAZE algorithm. AMAZE algorithm is described in section 5.5. It is composed by the combination of techniques the serves the purpose of wire length, delay and run time improvements for Maze Routers. The section 5.5.1 explains how the traditional A* algorithm is extended to handle Steiner tree routing. Section 5.5.2 details our biasing method to improve wire length. Sections 5.5.3 and 5.5.4 presents our techniques for delay improvement, while section 5.6 describes our contributions for run time improvement. Section 5.7 presents experimental results and discussions on them. Finally, section 5.9 discusses the applicability of the algorithm to 3D circuits.

5.2 Problem Definition

A net is a set V of points v located at positions (x_v, y_v) on a grid. These points need to be electrically connected in the circuit. Point s also called the driver, transmits a signal to all other sinks. Let subset K be the set of critical sinks in the net. For each critical sink $k \in K$ it is required that the delay of transmitting the signal from the driver s to k be minimized. Critical sinks can be identified by incremental timing analysis, that is available at the logic and placement levels of most CAD tools. For the rest of the sinks

$t \in V - K$ delay is not considered and wire length should be minimized at most.

5.3 Tree Topologies

This section presents kinds of Steiner tree topologies and how they impact wire length and delay. Since wires are restricted to the Manhattan geometry for most of the VLSI technologies, we are particularly interested in the rectilinear Steiner trees. In order to understand a Steiner tree, we start by defining a Spanning Tree and the process of obtaining a Steiner Tree and finally a rectilinear Steiner tree. Figure 5.1 exemplifies graphically the three mentioned kinds of tree.

A Spanning Tree is an acyclic graph (tree) that connects all the *n* nodes. From a Spanning Tree, it is possible to reduce the size of the edges by creating new nodes (called Steiner points). A tree with Steiner points is called Steiner Tree. A Rectilinear Steiner Tree is made by reorganizing the edges to be vertical or horizontal only. Doglegs are allowed.

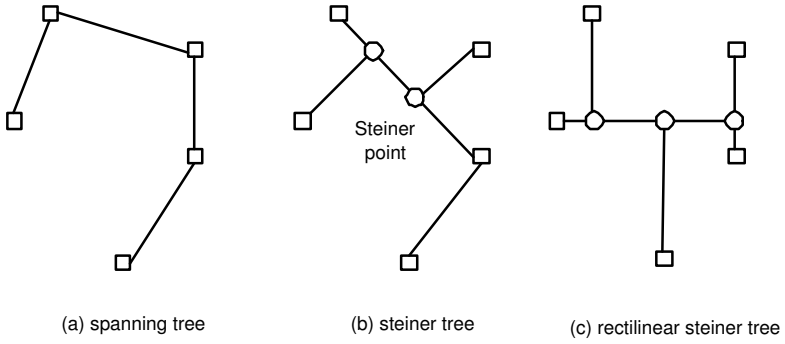


Figure 5.1: Example of spanning tree (a), Steiner tree (b) and rectilinear Steiner tree (c)

5.3.1 Delay Analysis

Elmore delay (ELMORE, 1948) is one of the most accepted models to compute delay of tree shaped wires due to its simplicity, fidelity (BOESE; KAHNG; MCCOY, 1995) and reasonable accuracy. Equation 5.1 describes the method, where D_k is the delay from the driver to a particular sink k, is R is the resistance of a piece of wire and $C_{downstream}$ is the downstream capacitance starting at this particular piece of wire. The downstream capacitance is a function of the tree topology; it determines the number of paths that share a same piece of wire, increasing downstream capacitance. The length of the piece of wire (w) influences the accuracy of the model and CPU time to compute it: the larger wire the worse accuracy and better run time. The sum should account for all wires on the path from the driver to the sink k. Supposing that the length of the path from the driver to k is l, then l/w pieces are accounted for. Refer to figure 5.2 for a didactic explanation of the model. It demonstrates a tree being sliced into pieces of wires and how the topology affects the downstream capacitance. Note that on the topology of tree 1 (figure 5.2.a) the paths for both sinks are shared by a single wire, while the tree on the topology 2 (figure 5.2.b) the paths are completelly separeted.

$$D_k = \sum(R \times C_{downstream}) \tag{5.1}$$

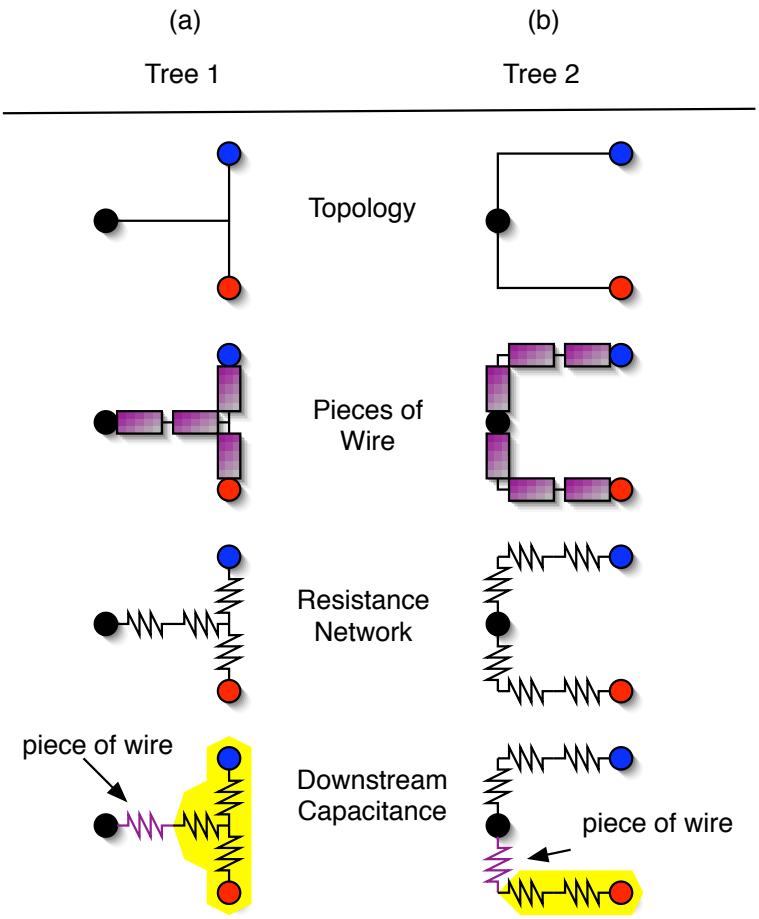


Figure 5.2: A didactic view of Elmore delay and the downstream capacitance.

Starting from equation 5.1, it can be broken into equation 5.2 by separating the driver resistance from the wire resistance where C_{total} is the total wire capacitance and R_{driver} is the output resistance of the driver cell. The first parcel is called gate delay while the second is wire delay.

$$D_k = R_{driver} \times C_{total} + \sum(R \times C_{downstream}) \tag{5.2}$$

By analyzing equation 5.2 the following facts can be understood:

- The driver resistance is an element of the delay calculation for any sink.
- The overall capacitance influences the delay according to the driver resistance; the higher the driver resistance the more important is the overall capacitance participation on the delay.

- The relation between the wire resistance and driver resistance will determine whether the overall capacitance or downstream capacitance are responsible to a larger chunk of the delay.

On the recent technologies, the cells' resistance is going down while wire resistance per unit is increasing (since wire width decreases). Additionally, drivers of delay critical nets are resized having their resistances even smaller. This scenario imposes a very significant wire resistance compared to the driver resistance, enforcing the growing importance of wire delay (right hand side of the equation 5.2). The following features on the topology of a tree can be addressed to improve delay for a particular sink k.

- The path length l of the connection from the net driver to k along the tree.
- The amount of capacitance plugged into the wire on the path to k
- How close to the driver is a branch of the tree with a high amount of load. Note that the closer is the branch to the wire the sooner it is not accounted into the downstream capacitance.

In order to address these features, previous works proposed tree topologies categories and algorithms. These topologies are reviewed in section 5.3.2. Usually, high performance connection topologies tend to distribute sinks on different branches of the trees in order to avoid excessive load for a wire on the path to a critical sink.

5.3.2 Topologies for Delay and Wire Length Trade-off

Delay and wire length are strongly related to Steiner tree topologies. Various kinds of rectilinear Steiner tree topologies are shown for a net in Figure A.14. Figure A.14.b shows a minimum length Steiner tree (MRST), which minimizes wire length but may present high delay to nodes that accidentally get connected far from the driver. Figure A.14.c shows a minimal Steiner arborescence (MSA) (RAO, S. et al, 1992)(CONG; LEUNG; ZHOU, 1993) or shortest path tree (SPT) which is a tree with shortest paths from the source to any sink. Such a tree minimizes source to sink distances at the expense of total wire length but do not necessarily minimize overall delay, affected by its increased wire length and capacitance sharing along the paths. Figure A.14.d shows a bounded radius Steiner tree (BRST) (CONG, J. et al, 1992) (ALPERT, C. et al, 1995) (CONG; KOH; MADDEN, 2001) in which the maximum distance from the source to any sink is bounded, exhibiting a compromise between MRSTs and SPTs. The Star tree topology shown in Figure A.14.e has separate wires for each sink, resulting in optimal path length and minimum sharing, but possibly huge total wire length and therefore non optimal delay, depending technology parameters and on the positions and number of pins.

It is clear that these topologies trade-off wirelength for delay. In real designs, there are two main strategies to control algorithms such that timing closure is achieved: slack satisfaction and critical path optimization. Although slack management is very precise, it is more complex, and many CAD tools rely on the optimization of critical paths, that are easily identified by incremental timing analysis. Optimization of critical paths surely reduce the worst logic delay and is an effective method of improving circuit speed. This way, we can relax the requirement of reducing the delay for all the sinks and concentrate on a few critical sinks that participate in critical paths. In this scenario, (KHANG; ROBINS, 1995) (BORAH; OWENS; IRWIN, 1997) (BOESE; KAHNG; MCCOY, 1995)

proposed the *Identified Critical Sink Routing Trees* in which a Star tree like topology is used for one identified critical sink while the rest of the sinks are connected by a minimum wire length Steiner tree (resulting in smaller Elmore delay to the critical sink). An example of such tree is given by Figure A.14.f.

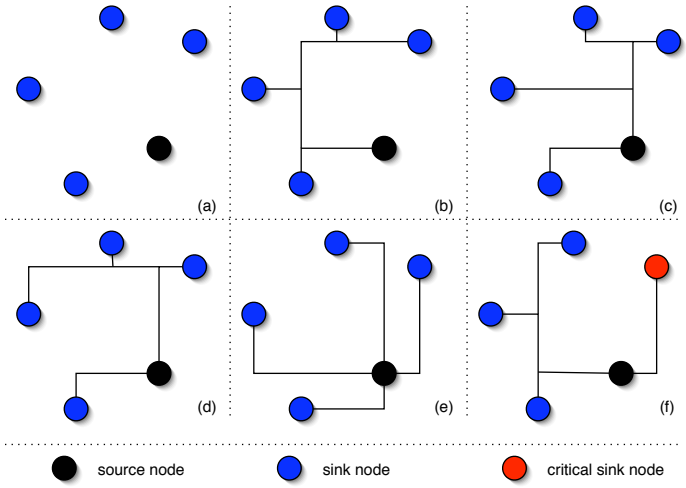


Figure 5.3: Steiner Topologies for delay optimization; (a) net; (b) Minimum Steiner Tree (MST); (c) Minimum Arborescence (MSA); (d) Intermediate topology between (b) and (c) - Bounded Radius Steiner Tree (BRST); (e) Star topology; (f) Critical Sink Approach (CSA)

5.4 Review of existing algorithms

5.4.1 Algorithms for Steiner Tree Construction

The the minimum length Steiner tree (MRST) problem is np-complete (GAREY; JOHNSON, 1977). There are several approaches to find exact Steiner trees in exponential time. Among them, the GeoSteiners (ZACHARIASEN, 1999) is the fastest. Heuristic methods are able to deliver close to minimum length Steiner trees in polynomial time, such as (MANDOIU; VAZIRANI; GANLEY, 1999) (GRIFFITH, J. et al, 1994).

Performance-driven Steiner tree algorithms have been well studied and a variety of methods have been proposed. The first category of algorithms is based upon the Minimum Spanning Tree and shortest path algorithms. Hou et. al. (HOU; HU; SAPATNEKAR, 1999) have shown techniques to find the shortest length Steiner tree under pin delay constraints. Alpert et. al. (ALPERT, C. et al, 1995), Boese et. al. (BOESE; KAHNG; MCCOY, 1995) and Cong et al. (CONG, J. et al, 1992), have described several such algorithms. The Spanning Tree could be generated based upon different criteria. Boese et. al. (BOESE; KAHNG; MCCOY, 1995) proposed the use of an Elmore routing tree for the same purpose. The AHHK algorithm described by Alpert et. al. (ALPERT, C. et al, 1995), just like (CONG, J. et al, 1992), builds a Steiner tree that trades off between shortest path and Minimum Spanning Tree (fig A.14.d). These methods only see the graph containing net pins to build Spanning Trees, and usually employ a separate Edge-Overlapping procedure for conversion to Rectilinear Steiner Trees (RST). Lillis et. al. (LILLIS, J. et al, 1996) generate many topologies to minimize the Elmore delay for attending a required delay budget at each sink. The variety of topologies provides a wide

range trade-off between wirelength and delay. Constructive algorithms have been proposed by Cong et. al. (CONG; LEUNG; ZHOU, 1993), Hong. et. al. (HONG, X. et al, 1993), and Xu et. al. (XU, J. et al, 2002). The Minimum Steiner Arborescence (MSA - fig A.14.c) generated by Cong et. al. (CONG; LEUNG; ZHOU, 1993) tends to have a high total wire length. Boese et. al. (BOESE; KAHNG; MCCOY, 1995) proposed the SERT-C algorithm for individual critical sink routing, in which the wire to the critical sink is not shared and the rest of the tree is build as short as possible disregarding the delay of the sinks other than the critical one (fig A.14.f).

In general all the methods described so far do not account for blockages, congestion etc. With the exception of (BOESE; KAHNG; MCCOY, 1995) and (BORAH; OWENS; IRWIN, 1997), they also generally minimize the source to sink distance for all sinks in the net without discriminating between critical and non-critical sinks.

Algorithms based on path search on the other hand use intelligent methods that incorporate the desired properties into the search process to generate the tree. Commonly used path search algorithms include basic Dijkstra and the A* algorithm. Dutt et. al (DUTT; ARSLAN, 2006) present an algorithm to perform incremental routing using Dijkstra algorithm to connect nodes to an existing tree in a restricted interval that satisfies the timing constraint. Hur et. al. (S-W.; JAGANNATHAN; LILLIS, 2000) present a method based on a multi-graph model for performance driven routing of two pin nets with wire sizing. Prasitjutrakal and Kubitz (PRASITJUTRAKUL; KUBITZ, 1990) have also proposed a basic timing-aware router. While this method uses Elmore delay to drive the A* search, their choice of the next target to be added to the tree is restrictive and in some situations can compromise the quality of the results in the presence of blockages.

5.4.2 Review of Path Search Algorithms

The shortest-path search in a graph is a well known problem in computer science. It is also very useful in real-live applications, such as telephone lines routing, gaming, circuit routing, network routing, etc.

Given a graph *G = (V,E)*, a source node $s \epsilon V$, a target node $t \epsilon V$ and a cost function *C* $E \rightarrow \Re$ the goal is to find the sequence of edges such that the sum of costs associated to this edges is minimal.

This problem is not np-complete. Dijkstra (SHERWANI, 1998) proposed a $O(n^2)$ shortest-path algorithm, where *n* is the number of nodes in the graph. However, the worst case is very unlikely to happen. In routing, the graph is a regular grid and costs range in similar values. Dijkstra algorithm will typically search a circular area around the source node, until the circle reaches the target, as shown in figure 5.4 (a).

Dijkstra's algorithm is based on two operations over nodes: **open** and **expand**. Open operation is performed every time a node is reached by the search. It consists of calculating the distance from the source to the opening node *n* called *g(n)* and inserting it in a list called **openlist**. The openlist must be kept sorted by *g(v)*. For this reason, the open operation is performed in *O(log(n))* time, since the openlist is usually implemented with a priority queue (or binary heap). A node is expanded if the search selects it as a candidate to be on the shortest path. The expand operation consists of marking the node as expanded (so that it won't be expanded twice) and opening all its reachable neighbors. The search algorithm is given described in algorithm 8.

In order to speedup the search, A* (HART; NILSSON; RAPHAEL, 1968) proposed the usage of a heuristic estimation of the whole path length passing though *n* called *f(n)*. It is computed by adding *g(n)* to a heuristic estimation from *n* to the target, called $k(n,t)$ (or

Algorithm 8 Dijkstra algorithm for shortest path

```
1: Open(s)
2: found = false
3: while ( OpenList is not empty )
4:    Node = TopOfTheList
5:    Pop TopOfTheList from OpenList
6:    if (Node = t) then
7:       found = true;
8:       break; //the shortest path was found
9:    end if
10:   Expand(Node)
11: end while
12: if (found) then
13:   Retrace the path from t
14: end if
```

simply $h(n)$). By keeping the openlist sorted by f instead of g, the *less promissing* nodes to be part of the shortest path (the ones with higher h) would be delayed for expansion. If h is underestimated (is less or equal to the actual distance), the estimation is said to be **admissible** and the path found will be the shortest possible. This property of A* is called **admissibility**. If $h(v)=0$ for every node then A* behaves like the Dijkstra's algorithm, expanding nodes closest to the source first. The heuristic estimator should also be **consistent**. Consistency is defined as follows: $k(v_1,t) = k(v_1,v_2) + k(v_2,t)$ for nodes v_1 and v_2. If h is not consistent the algorithm must be modified to allow multiple expansion of nodes.

The higher is the heuristic estimator (e.g. the closer is the heuristic estimator to the actual distance), the less nodes are expanded by A*. Figure 5.4 shows graphically the searched area by Dijkstra and A* with different estimators. Dijkstra algorithm will always search a large area, even if the shortest path can be easily recognized. A* algorithm will depend on the quality of the estimator. If the estimator is *zero* than A* searches the same area as Dijkstra. If the estimator is *good* than A* will search a smaller area than Dijkstra. As the estimator quality, A* approximates to a DFS search direct to the target.

In routing applications, Manhattan Distance (*ManhD*) is commonly used as heuristic estimator. It is consistent and admissible. Let C^* be the optimum cost of reaching the target t from s. Algorithm A* expands each and every vertex v with $f(v) < C^*$ and no node with $f(v) > C^*$ (HART; NILSSON; RAPHAEL, 1968). We say that a **tie** happens whenever two nodes have the same value of $f(n)$ in the open list. A **critical tie** is a tie with the additional constraint that $f = C^*$. Simple ties are not a concern, given that all nodes with $f(v) < C^*$ must be expanded to ensure admissibility. Yet critical ties are very significant, specially for routing. The use of Manhattan Distance as estimator and no additional costs for congestion or obstacles in initially empty areas make all estimates perfect, so all nodes inside the box bounded by the source and the target have $f(n) = C^*$. To get the most efficiency from A* a mechanism is needed to avoid expanding all nodes with critical ties. In (HART; NILSSON; RAPHAEL, 1968) the authors point out that critical ties can be arbitrarily broken but always in favor of the target. Additionally, vertices that are closer to the target can be chosen to break intermediate critical ties for efficiency purposes. So, if two nodes have the same value of $f(n)$, theoretically the one with higher $g(n)$ should be expanded first. In routing this causes the effect of Depth First

Searches (DFS) from s to t in empty areas. Now, in a regular and uniform grid, each expanded node that is not aligned to the target in x or y will open two neighbors with the same value of $f = C^*$ and the same value of g, what we will refer to as a **depth tie**. While depth priority can be used to get efficiency, depth ties represent a true degree of freedom for selecting between alternate paths every time this choice happens. A separate mechanism must be implemented to do that, and this will be addressed in section 5.5.2.

There is a final concern regarding ties. The routing grid cannot be assumed to be regular or to have the same step size in x and y. In a Hannan grid, two neighbors of a node may exhibit different values of g, and in this case we would lose the ability to recognize at this point that these are alternate paths that we must choose from. To cope with this, instead of using g (depth) as the critical tie breaking criterion, the number of expansion steps can be employed. For each vertex v a value $cs(v)$ is stored that indicates how many expansions were needed to reach v is from the source. We pick that vertex v with the highest $cs(v)$, and the definition of a depth tie is adapted accordingly.

In summary, a maze router using the A* algorithm can be made very efficient for routing. When running in empty and not congested areas whose costs are uniform and known, with the perfect Manhattan distance estimator and critical tie breaking based on $cs(v)$ the algorithm expands only the nodes that lie in the optimal path, most like in a DFS. A Hannan grid provides reduced number of nodes and still preserves the degree of freedom regarding the choice of nodes that exhibit stepped depth ties. Additional speed-up methods are addressed on section 5.6. For global routing with congestion information, variable costs will slow down the search and change the occurrence of all types of ties. In extreme situations, bidirectional search and dynamic estimation methods such as LCS* (JOHANN; REIS, 2000) can be applied.

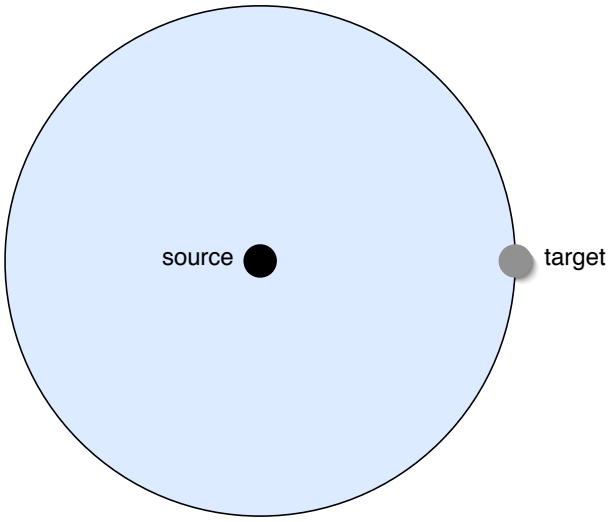
5.5 Amaze Algorithm to generate Steiner Trees

5.5.1 A* for Steiner routing

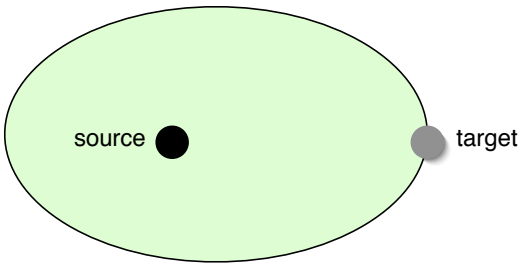
Given a set of pins of the same net, a maze router will start from a particular pin and grow the tree inserting one pin at a time with the path search method. We initialize the tree to be the source and then each sink is added one by one to the existing tree, as shown in figure 5.5. Since we must have the capability to connect sinks to any point on an existing tree the path search mechanism must be extended to accommodate multiple sources. This way all vertices on the existing tree are potential starting points for the new connection.

One more feature is to select which target is going to be connected to the tree at a time. Prim's algorithm for Minimum Spanning-Tree connects the closest pin to the existing tree at each step. We follow the same heuristic. In order to ensure that the A* algorithm will select the closest sink to the tree even in the presence of blockages whose impact on the path length cannot be estimated, we explore the concept of multiple targeted search. The accurate selection of the closest target leads to smaller wire length, as shown in the figure 5.6. In situation (b) we considered a static estimation of the closest target. In situation (c) the target was selected dynamically during the search. In situation (c) the wirelength is shorter because the paths were shared.

Adding the capability of dealing with multiple sources and targets to the classical A* is straightforward. The algorithm 9 presents the steps of the algorithm in detail. All the sources are open initially (see steps 2 - 8) and the priority-queue of the open nodes will automatically select the most promising node to start with. Multiple targets can be handled simply by stopping the algorithm whenever any target is reached (see steps 16-



(a) Dijkstra Algoritm / A* without estimation



(b) A* Algorithm with a good estimator

(c) A* Algorithm with an exact estimator

Figure 5.4: Comparison between the searched area of Dijkstra's algorithm and A* with different estimators

18). The heuristic function *h(v)* will point to the target *t* that is the closest to *v*. The concept of chosen target *ct(v)* is introduced and stored in the references *r* that are stored on the open list. Every open node has an associated chosen target that is the closest target to it measured by Manhattan distance.

Note that the algorithm 9 makes use of temporary and global variables. The open list, the graph G and the sets S and T are the global variables. All other variables, such as $g(s)$ are temporary variables; a value g will be definitive only when the node is actually expanded (after step 13). For each expanded node, the algorithm stores the previous node (parent) in order to retrace back the path. Only at expand time the final parent can be defined; note that a temporary parent is obtained at open time on step 26. However, at expand time, on step 14, the parent of node v is finally stored in the global variable $G.pred(v)$.

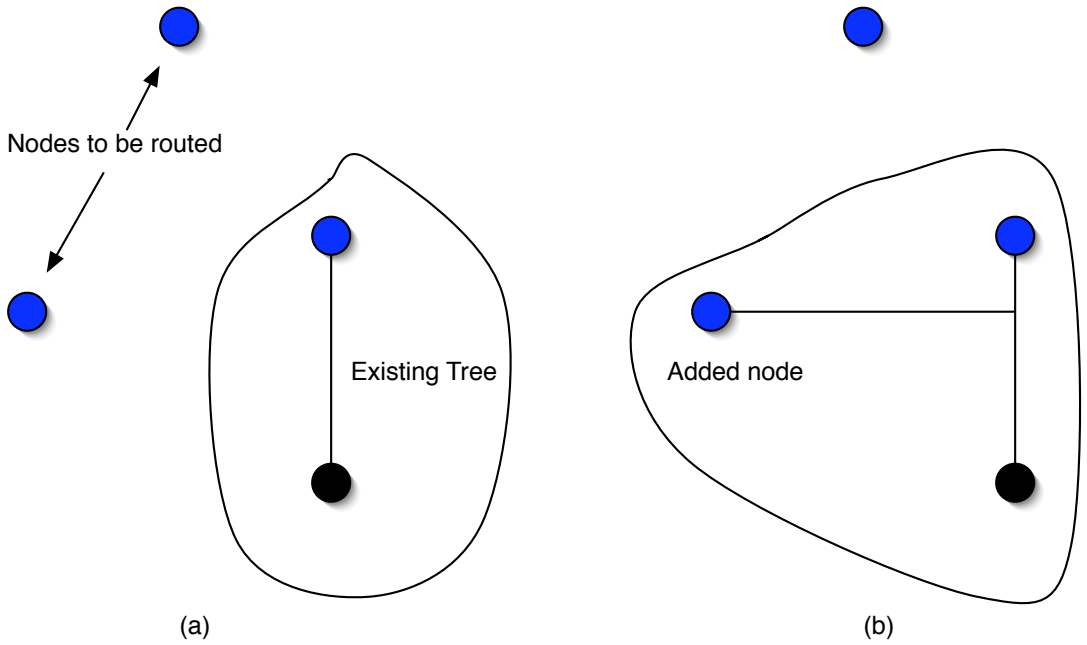


Figure 5.5: A step of the AMAZE algorithm; (a) provides the current configuration of the routing tree and the remaining nodes to be routed; (b) provides the next configuration, after one adding a node to the tree.

An important property if that *f* monotonically increases during the search (monotonicity property). This property was already demonstrated (HART; NILSSON; RAPHAEL, 1968) if *h* consistency holds.

Theorem 1 demonstrates the consistency of the multiple target heuristic estimator.

Theorem 1. *For two nodes* n_1, n_2, *each with a different assigned target* $ct(n_1)$ *and* $ct(n_2)$ *respectively,* $k(n_1, ct(n_1)) \leq k(n_1, n_2) + k(n_2, ct(n_2))$

Proof. At node n_1, we know that the closest target is $ct(n_1)$, so $k(n_1, ct(n_1)) \leq k(n_1, ct(n_2))$. By the consistency property for the same target, $k(n_1, ct(n_2)) \leq k(n_1, n_2) + k(n_2, ct(n_2))$. Joining the equations, we conclude the proof. □

Another important property if that f monotonically increases during the search (monotonicity property). This property was already demonstrated (HART; NILSSON; RAPHAEL, 1968) if *h* consistency holds.

In the presence of delay critical sinks, we first route critical sinks in order of criticality and than we route the non-critical ones. When routing a critical node, regular nodes are not considered targets, but they are used for biasing calculation (section 5.5.2). The algorithm for generating a steiner tree with priority to the critical nodes is given in algorithm 10.

5.5.2 Biasing technique for wire length optimization

As already stated, often there are situations when two or more vertices (for instance v and n) have the same value of f as well as cs ($f(v) = f(n)$ and $cs(v) = cs(n)$). Uniformity of the grid will lead to such situations. The choice of the appropriate v will

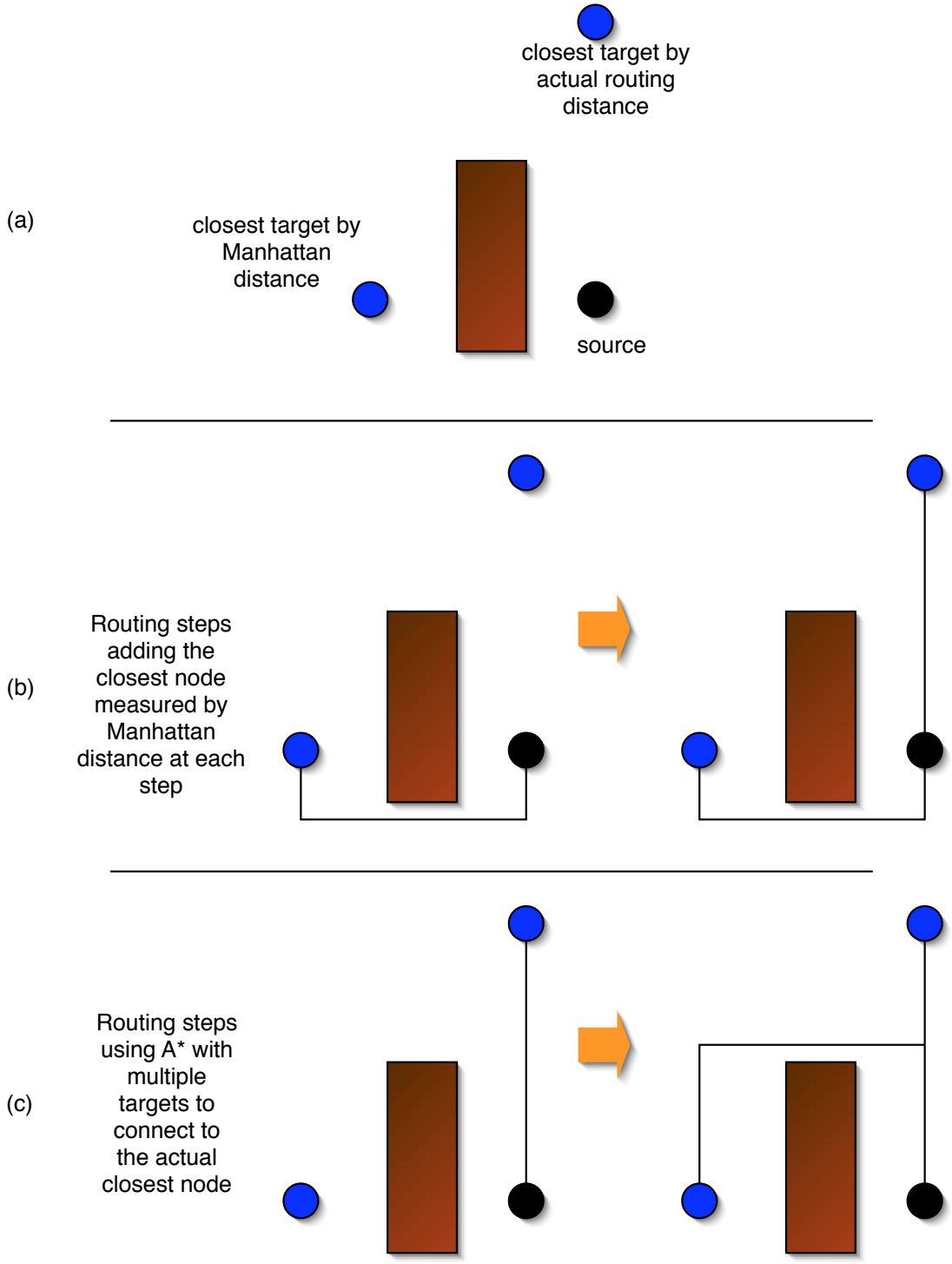


Figure 5.6: The reason for using multiple targets in A* instead of getting the closest node; (b) shows the steps taken by a routing algorithm that gets the next node using Manhattan Distance; (c) shows the steps by getting the actual closest node instead. The final tree in (c) is smaller than in (b).

determine whether the upper-left L shaped wire or the lower-right L shaped wire will be selected, as illustrated by figure A.11. These cases are degrees of freedom provided

Algorithm 9 A* Mult

Input: A graph $G = (V, E)$, a set S of sources and a set T of targets.

Output: A shortest-path with cost C^* from some $s \in S$ to some $t \in T$ such that ($\forall s_1 \in S \forall t_1 \in T$ the cost C of the shortest-path from s_1 to $t_1 \leq C^*$). New sources are created in S and t is removed from T.

1: Create an open list L and insert a reference $R(s)$ into it.
2: **for** every $s \in S$
3: $g(s) = 0$ ($s \in S$)
4: $ct(s) = ct$ such that $\forall t \in T(ManhD(s, ct) \leq ManhD(s, t))$
5: $h(s) = ManhD(s, ct(s))$
6: Mark the predecessor pointer $p(s)$ as invalid
7: Create a reference $R(s)$ with $g(s)$, $h(s)$, $ct(s)$, $p(s)$
8: Insert reference $R(S)$ into L
9: **end for**
10: **while** (L is not empty)
11: Remove the first reference r from L
12: Get the vertex v from r
13: If v is closed than **continue** to the next iteration.
14: Set $G.pred(v) = r.p(v)$
15: Mark v as closed
16: **if** ($v \in T$) **then**
17: Set $t = v$ as the reached target
18: **break**
19: **end if**
20: **for** each vertex u that is adjacent to v in G and is not closed
21: $g(u) = g(v) + c(v, u)$
22: Set $ct(u)$ as in step 4
23: $h(u) = ManhD(u, ct(u))$
24: $p(u)=v$
25: Create $R(u)$ with $g(u)$,$h(u)$, $ct(u)$, $p(u)$
26: Insert $R(u)$ into L
27: **end for**
28: **end while**
29: **if** a target is found **then**
30: Retrace back the path from the reached target t until some $s \in S$ is reached. Use $G.pred(i)$ (i are intermediate nodes in the path); Insert all i in S. Move t from T to S.
31: **else**
32: Report that no $t \in T$ is reachable from S
33: **end if**

by the A* search. Either choice is valid and will lead to an admissible (shortest) path. Instead of letting the coding style implicitly decide on the choice we introduce a **biasing** technique to direct the search according to the needs of future connections to unrouted sinks. Clearly, the choice (a) in figure A.11 is best for wire length. The biasing technique will attempt to select the wire accordingly.

Biasing uses a reference point called **biasing point** that is used to choose the vertex for expansion. The biasing point is calculated by first determining a set of affected targets.

Algorithm 10 A* Steiner with critical nodes

Input: Graph *G*, the net driver *s* and a set *T* of sinks with an associated criticality.
Output: A Steiner tree connecting *s* to all targets in *T*.

1: S = s
2: **while** (T is not empty)
3: Call A* (S, T_c) such that T_c contains only the nodes from T with highest criticality
4: $T = T - t$ where t is the A* chosen target
5: $S = S \bigcup V_{st}$ where V_{st} is the set of all vertices in the path P returned by A*, including t
6: **return** S
7: **end while**

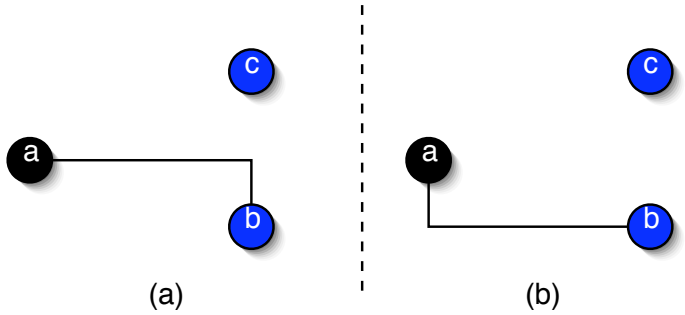


Figure 5.7: Illustration of a routing situation favorable to wirelength minimization (a) and a favorable situation for the isolation of the paths (b).

Let T_p be the set of vertices in the quadrant with origin p (p is the predecessor of the candidate node v) that is diagonally opposite to t and T_t be the set of vertices in the quadrant with origin t that is diagonally opposite to p. The biasing point will not be affected by vertices in $T_p \cup T_t$, since p or t will be closer to these vertices than any other vertex in the routing box bounded by p and t, as illustrated by Figure A.12, situation (a).

Likewise, any vertex in the set of vertices (denoted by T_u) that are closer to the tree than the routing box bounded by v and t will not affect the biasing point either, as shown in figure A.12 situation (b). Situation (c) represents one node that will affect the biasing calculation since it is neither behind the source/target nor closer to the tree than the routing box.

Therefore, $T_{affected} = T - (T_p \cup T_t \cup T_u)$. If (x_t, y_t) is the coordinate of a target $t \in T_{affected}$ whose distance from the vertex v is d_t the bias point is computed as a weighted centroid of the affected nodes according to equation 5.3. It is reasonable to assume that nodes closer to the vertex v have a higher probability of using the route being performed in future connections.

$$(x_c, y_c) = \left(\frac{\sum_{t \epsilon T_{affected}} \frac{x_t}{d_t}}{\sum_{t \epsilon T_{affected}} \frac{1}{d_t}}, \frac{\sum_{t \epsilon T_{affected}} \frac{y_t}{d_t}}{\sum_{t \epsilon T_{affected}} \frac{1}{d_t}}\right) \tag{5.3}$$

The algorithm that calculates the biasing value *b(v)* is shown bellow (algorithm 11).

The bias value $b(v)$ is calculated by the distance from v to the biasing point. Vertices

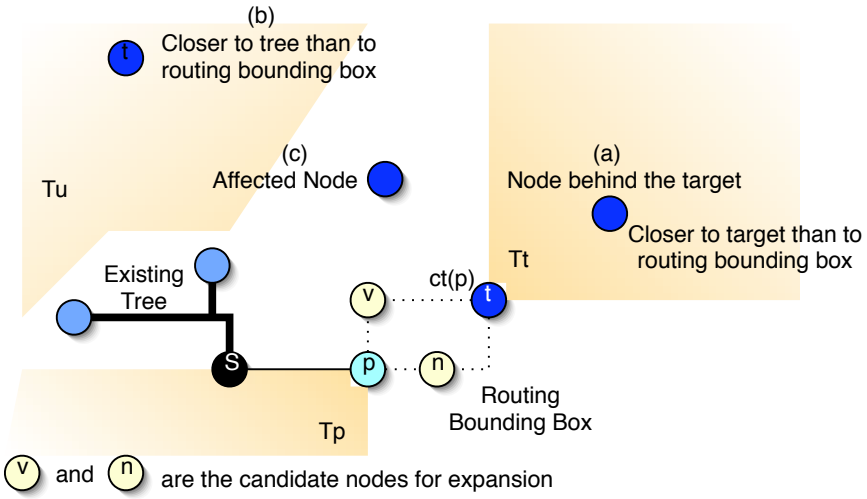


Figure 5.8: An illustration of the biasing technique and the affected target. (a) shows one target that is excluded because it is behind the target. (b) shows one target that is closer to the existing tree than to the routing bounding box. (c) is an example of a target that will affect the biasing point. In this situation, the path will go into the direction of node v.

Algorithm 11 Biasing Value

Input: The node to be expanded v, the parent n, the closest target t, the complete set of targets T, graph G.

Output: The value $b(v)$.

1: Compute the unaffected region R_p that is the opposite quadrant of p from t.
2: Compute the unaffected region R_p that is the opposite quadrant of t from n.
3: Let T_p and T_t be the set of targets in R_p and R_t.
4: Determine the set of vertices T_u that are close to the tree than to the rectangle defined by s and t.
5: Compute the centroid (X_c, Y_c) as explained before.
6: **return** $b(v) = |x_p x_c| + |y_p y_c|$

with the same $f(v)$ and $cs(v)$ in the open list are sorted by increasing values of $b(v)$ (the node with smaller $b(v)$ is selected). Figure A.13 illustrates the effect of the biasing technique on a 7-pins net. In this case, the pin placement favored the sharing of some wires and the biasing technique provided a 15% improvement in wire length.

Though biasing helps to reduce the total wire length, it could potentially result in sharing a path P from source to critical target with paths to other targets thereby increasing the capacitive load on P, which, combined with a significant value of wire resistance, can slow it down. To isolate critical paths and make them less likely to be shared we suggest the use of repulsive biasing for critical targets. Repulsive biasing sorts the open list in decreasing order of $b(v)$ and tends to route wires that connect the source to critical targets in such a way that ample space is available for routing non-critical wires.

On the run time analysis, it is important to notice that biasing does not increase the complexity of the algorithms. A detailed analysis of how it can be efficiently implemented is provided on section 5.6.4. Also, in favor of the biasing technique, reducing wire length

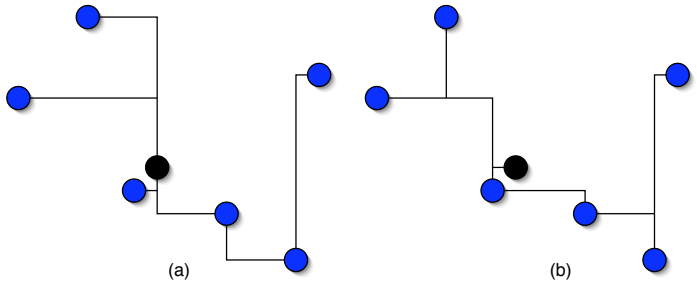


Figure 5.9: Visualization of trees (a) without biasing and (b) with biasing.

Table 5.1: Impact of the biasing technique in average for WL (measure in μm and run time (s). Imp rows represent the improvement achieved by using the biasing technique.

	Random				Placed Circuit	
	3 pins		15 pins			
	wl	t (s)	wl	t (s)	wl	t (s)
	Considering the costs of Vias					
Off	314	1.38	915	7.24	3937	45.6
On	304	1.38	915	7.24	3937	45.6
Imp	3.0%	0%	0.7%	-2.8%	0.9%	0.3%
	Ignoring the vias					
Off	311	1.34	905	6.38	3899	42.0
On	301	1.23	881	6.58	3827	43.3
Imp	3.3%	-9.8%	2.6%	-3.1%	1.8%	-3.0%

reduces the overall number of expanded nodes, since wire length is shorter, resulting in possible run time savings.

We observe that the biasing technique is able to improve the wirelength by breaking ties in the open list. The number of ties on a search is highly dependent on the modeling of the routing space. Cost functions that model congestion, for example, will reduce the amount of ties occurred in a search. Also, modeling of the vias will minimize the number of bends but also reduce the degrees of freedom for the biasing technique, since Z shaped connections cost more than L shaped ones.

In order to evaluate the impact of the biasing technique in wire length and run time we performed experiments in two sets of benchmarks: random and placed circuits. For the random set we generated one thousand trees varying the number of pins from 3 to 15 in a space of 300μm x 300μm. The placed circuit set was extracted from the ibm02 circuit from ISPD 2004 placement benchmarks suite after full placement. This circuit has 19584 nets; 54% are 2-pin nets (those are being ignored) followed by 9%, 9%, 9%, 2%, 1.5%, 1.5%, 2%, 2%, 2.5% for 3, 4, 5, 6, 7, 8, 9, 10 and 11 pins; 7.5% of the nets are between 12 and 134 pins. The average results are presented in Table 5.1. For both options, we study the impact of modeling the vias or not. Analyzing the table, we report average gains in the order of 1-3% considering the vias and 2.5-3% ignoring the vias, reinforcing the conclusion that the modeling of the vias will reduce the degree of freedom for the biasing technique and consequently increase wire length.

We observed that the average numbers are very small considering the visual impact

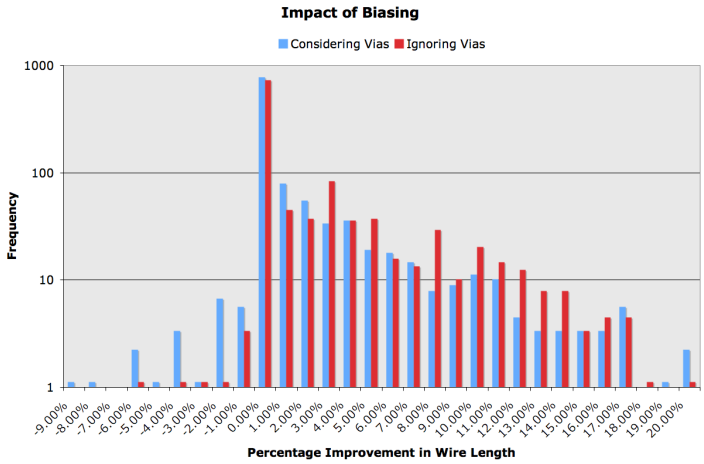


Figure 5.10: Histogram showing the impact of biasing on nets taken from a placed circuit. The big majority of nets is not affected by biasing, but almost none is affected negatively while improvements can be in the order of 20%.

(exemplified by figure A.13) of the biasing technique. We then plotted a histogram of the biasing improvement for the placed circuit nets with 3 or more pins. The histogram, on Figure 5.10, shows that in the big majority of cases the impact is 0%. Analyzing each case, we could observe that in fact the case where wire sharing does not help is the most common. However, for the ones that are affected by wire sharing, biasing can improve the wire length of a net by up to 20%. We also observed in the histogram the same reduction on the improvement of the biasing technique with the modeling of the vias.

5.5.3 Sharing Factor on Maze Search

From the theory established in the earlier sections it is clear that delay performance can be effectively managed by controlling the amount of sharing. We introduce the capability of wire length to delay trade-off using a parameter called the *sharing factor* (*sf*). The sharing factor has a value between 0 and 1 and is used to designate some parts of the tree as prohibited for connection, avoiding sharing of these wires. Critical wires connect the driver to critical sinks. From the formulation presented in the previous section, sharing a small amount in the beginning of this wire may be less harmful than sharing the whole critical wire.

In step 1 of A* Mult algorithm we ascertain, for every node s_k on the tree whether or not it is part of a critical path. For every node n that is part of the tree, we store the distance d_n of the node from the source along the path of the tree. This distance can be obtained by the g from previous searches. If k is a critical sink, then all wires that are closer to s_0 than $sf \times d_k$ are available for sharing. Clearly, if *sf=1* then all wires in the path from s_0 to s_k are available for sharing and if *sf=0* then no wire in the path is available for sharing. Figure A.15 shows the effect of different values of s_f on the types of routes generated.

5.5.4 Path Length Factor on Maze Search

Path length is the term used for the distance of a critical sink k to the net driver s along the tree. If the path length is optimal for k then optimal wire resistance (disregarding

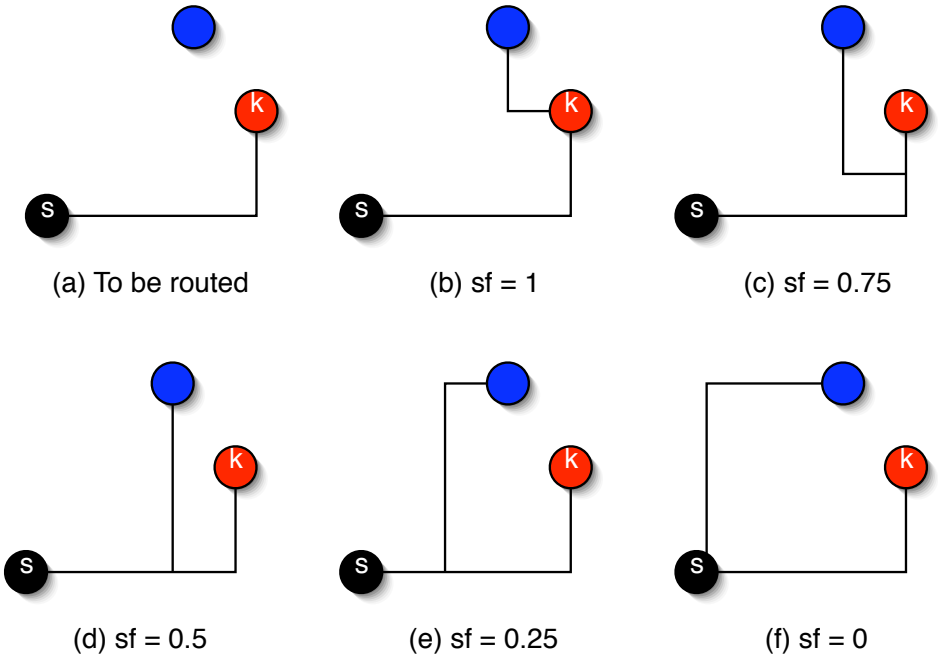


Figure 5.11: Effect of sharing factor on maze search.

possible wire sizing) is obtained for it. For example, consider figure A.16, where k_1 and k_2 are connected to the driver with optimal path length. Observe that node k_3 can be connected into the path to k_1 with optimal path length as well, but connected to k_2 it would have a small detour.

The previous sections assumed that minimum path length to the critical sinks could be achieved easily by performing their connections first. It is always true for a single critical sink but may not be for multiple critical sinks if sharing is available. In such cases, we propose the use of a path length factor (plf) in order to guarantee that the minimum path length for the critical sinks is achieved. The idea was based on (CALDWELL, 1997) and extended to attend critical sinks only. For a node $n \in S$, the path length factor uses d_n (distance from the source along the tree). At each shared node, a number between 0 and 1 is multiplied by d_n. The product of this multiplication is set as initial g of the node (affecting step 3 of algorithm 9). The effect of initializing g with a constant value like 0 is that the obtained routes will not correspond to shorter wires from the source; the proposed algorithm will optimize overall path length instead. A trade-off of the two goals can be obtained with the plf. If the path length factor is set to 0, then a MST-like tree will be generated. All intermediate values trade-off wire length for path length.

5.6 Run Time Improvement Techniques

5.6.1 Application specific grid

In general, the use of a coarser grid improves the speed of the algorithm at the expense of quality. During early estimation speed is of essence since estimates need to be generated in the inner loops of the synthesis algorithms. In this case we use a Hanan grid (HANNAN, 1992) to model the space thereby reducing the grid to the smallest size that

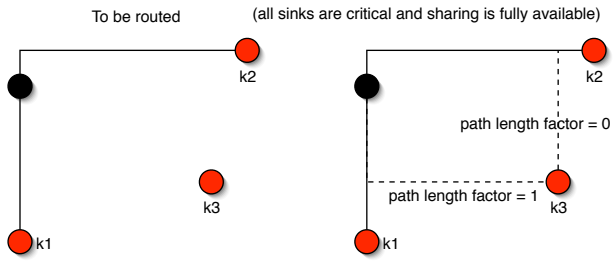


Figure 5.12: Effect of path length factor on maze search.

accommodates all the sinks. This leads to better memory usage as well as shorter runtime. During actual routing we use a finer grid thereby increasing the routability and improving wire length at the expense of CPU time.

Costs are also supported by a maze router and can be used at the expense of run time as well. Congestion cost can be modeled by a cost function. It is important to notice that those costs must remain static during the A* expansion in order to guarantee admissibility. Applications such detailed routing could take benefit of increasing costs dynamically during the expansion to cope with design rules.

Another feature of the maze routers is the ease to model multi-layer routing. This is merely an extension of the maze router to a three dimensional scenario with via costs assigned appropriately. As discussed in many sections of the routing chapter, the situation where *f* is unchanged during the whole search is the best for performance of the algorithm. To achieve that, in the case where the space is modeled in 3D, via costs must be included in the *h* function. If the *Z* coordinate of the node *n* being open is the same as its target (*ct*) and in any of the situations described below, the value of twice the via cost can be included in the *h* function without harming the algorithm admissibility:

- $n.X \neq ct.X$ and $n.Y \neq n.Y \neq ct.Y$
- $n.X = ct.X$ and $n.Z$ is horizontal layer
- $n.Y = ct.Y$ and $n.Z$ is vertical layer

This improvement was tested on Steiner trees extracted from ibm02 placement benchmark and we got 17% reduction in the number of expanded nodes.

5.6.2 Simplified heuristic function (h) calculation

Every time a node is open (steps 21-26), it has an associated *closest target* that is used to calculate its *h* value (as demonstrated in steps 22 and 23). Consider the scenario where a node *p*, with associated closest target *ct*, is being expanded and is opening its neighbors *n*. We propose that, instead of looking at all of the targets, only *ct* be checked. The value f'_n denotes the f function of node n forcing it to point at ct without measuring if ct is actually the closest target from n. Consider two possibilities: $f'_n = f_p$ and $f'_n \neq f_p$. Theorem 2 explores the first possibility.

Theorem 2. *If* $f'_n = f_p$, *then* $f_n = f_p$ *with the same ct.*

Proof. 1. $f'_n = f_p$

2. $f'_p \leq f_p$ (monotone property)

3. $f'_n \leq f_n$ (combination of 1 and 2)
4. $f'_n \geq f_n$ (this step is a requirement of A* - it must choose the target that leads to smaller f).
5. $f_n = f_p$ (combination of 3 and 4)
6. The target for f'_n can be used.

□

Considering that $f'_n \neq f_p$, the associated target ct for n_i must be recalculated because it might have changed. Since the f_n will never be smaller than f_p this calculation could be postponed until the situation that there are no more nodes in the open list with the current value of f. This mechanism is implemented as follows. An auxiliary list of open nodes is created, where no f value is computed. The A* search does not look at this structure unless the current f value changes. In this case, the auxiliary open list is flushed and an open list is built afresh.

5.6.3 Specialized Open List

The open list data structure must be a priority queue sorted by *f (v)*, *cs(v)* and *b(v)* respectively. The insertion and removal operations in a standard STL priority queue implemented as a binary heap take $O(logn)$ time. While this is not a strong performance limitation by itself, it demands any open node *v* to have calculated *f (v)*, *cs(v)* and *b(v)* values. In our algorithm, we want to postpone the calculation of *b(v)* to expansion time (see section 5.6.4), for it is more expensive. Therefore, the binary heap turns out not to be the best data structure. We propose a specialized open list supported by theorems 3 and 4, which can be classified as a *bounded height priority queue*, a structure similar to what is used in bucket sorting.

Theorem 3. *During an A* search (with a single source node) search, the difference of the smallest f to the largest f in the open list at all times is restricted to twice the maximum cost in the graph.*

Proof. Consider the initial situation where the only open node is the source node s with f_s. After it is expanded, neighbors n are open and inserted in the open list. The one with largest f will cost $f_s + \Delta g + \Delta h$ where Δg is the highest cost edge connected to s that points in the opposite direction of the target ct. In the worst case, $\Delta h = \Delta g$ by the admissibility property and Δg is the highest cost in the graph (*HighestCost)*. In this case, the f range in the open list is given by $f_s + 2 \times HighestCost - f_s = 2 \times HighestCost$. As the search advances, the situation will be repeated for the expanded nodes p that follow. While p has $f_p = f_s$ the worst case f will be given by the same equation $f_p + 2 \times HighestCost$. When a node k with a higher value for f_k is expanded, the largest f will be given by $f_k + 2 \times HighestCost$, but the range is still given by $2 \times HighestCost$ since the smallest f in the open list is now f_k. □

Theorem 4. *Given multiple sources, the f range is given by the maximum between the f range of the initially open nodes and $2 \times HighestCost$.*

According to the monotone property, the minimum *f* value f_{min} is given by the first node expanded in the search. Based on the maximum range for the *f* function (*fr*) from theorems 3 and 4, we implement a circular array of *fr* positions, indexed by (*f* - f_{min}) *mod*

(fr+1). At each node, there is another array indexed by *cs* and finally, at each node of this two dimensional array there is a linked list of references to nodes. This infrastructure provides constant time insertion. On access or removal the array must be traversed until a non-empty list is found. The time complexity of such operations will be $O(fr \times csr)$ in the worst case and *O(*1*)* on average.

At the *expand* procedure, a linked list is located for the current value of f and *cs*. If this list contains only one node it is returned to A*Mult. When the list contains two or more nodes, the biasing technique is applied to decide which should be expanded first. The advantages of such a structure is that the biasing point and *b(v)* values are computed only if necessary, e.g. if A* reached both nodes and must decide which one to expand.

5.6.4 Biasing Implementation

In order to efficiently implement biasing, we need a mechanism to calculate both target excluding criteria efficiently. The first criterion is to exclude nodes that are behind the source or the target. For *t* candidate targets, it can be performed in O(*t*) time. The second criterion is to exclude nodes that are closer to the existing tree than to the routing bounding box. Considering that the tree has *n* nodes, this test would cost O(*nt*) time. However, the distance to the tree can be pre-computed by the A* Mult algorithm. In the step 3 of the algorithm we calculate the distance from every node in the tree to every target. This computation can be stored to be used by the biasing technique dropping the complexity to O(*t*).

5.7 Experimental Results

The experimental results are divided into the following sections:

- wire length experiments: to compare trees that AMAZE generates when optimizing WL alone to the optimal topologies, to heuristics for Steiner trees construction and to maze routers.
- delay experiments: to compare the trees generated by the AMAZE critical sink approach and AMAZE critical arborescence approach (path length factor is set to 1) to heuristics like AHHK (ALPERT, C. et al, 1995) and P-Trees (LILLIS, J. et al, 1996).
- wire length versus delay trade-off analysis.
- analysis of the impact of blockages on the generated topologies.

On all subsections we applied randomly generated trees since we understand it, statistically, covers all possible patters. Placed circuits may exhibit an uneven distribution of the patters that is sometimes consequence of that particular circuit or placement algorithm.

5.7.1 Steiner Wire Length Experiments

Initially we verified the wire length of our trees disregarding delay optimization. We compared AMAZE to three other algorithms: 1) the GeoSteiner software (ZACHARIASEN, 1999) that finds the optimal Steiner tree in an acceptable time; 2) the Labyrinth

Table 5.2: Wire length comparison of AMAZE with optimal trees (GeoSteiners) and other heuristics

Alg.	#nodes:	8	14	20	26	49	100
GeoSteiners	Wirelength	81.68	197.61	337.13	496.52	1289.23	3751.65
	CPU (s)	0.01	0.02	0.05	0.29	0.16	0.92
AMAZE WL	Wirelength	101.60%	101.80%	102.10%	102.20%	102.70%	102.40%
	CPU (s)	0	0	0.02	0.01	0.06	0.4
Labyrinth	Wirelength	111.10%	111.10%	111.70%	112.40%	112.60%	——
	CPU (s)	0.01	0.04	0.08	0.16	0.55	——-
AHHK (c=0)	Wirelength	101.30%	102.60%	102.40%	102.20%	102.60%	102.60%
	CPU (s)	0.01	0.01	0.01	0.02	0.07	0.53

maze router (KASTNER; SARRAFZADEH, 2005); 3) the AHHK algorithm (ALPERT, C. et al, 1995) configured for best wire length.

Table 5.2 presents the average results for 30 randomly generated trees ranging from 8 nodes to 100 nodes. All values are normalized to the GeoSteiner solutions (optimal trees). In summary, we can observe that AMAZE generates near optimal trees (within 2% in average) while run time is even better than the AHHK heuristic. We can also observe that Labyrinth could not find good trees and requires significant more run time. The case with 100 nodes in a tree could not finish because of memory requirements. Such a result enforces the importance of the techniques described in this thesis to the routing community.

5.7.2 Steiner Delay Experiments

We have conducted experiments with randomly generated nets, varying the numbers of total sinks (n_t) and critical sinks (n_k). For each (n_t, n_k) pair we averaged the results of 100 randomly generated nets. We compared three configurations of our algorithm to the P-Trees (LILLIS, J. et al, 1996) and to a special version of AHHK (ALPERT, C. et al, 1997). The first configuration (AMAZE WL) is set to minimize WL only. As demonstrated in the previous section, our trees are close to optimal in terms of WL. The second and the third configuration (AMAZE) are set to demonstrate the effectiveness of the sharing factor while fixing the path length factor to 0 and 1 respectivelly.

In these tests we used our own implementation of the AHHK algorithm, configured to use 0, 0.25, 0.5, 0.75 and 1 as the control factor, as suggested in (ALPERT, C. et al, 1997). When implementing the edge overlapping procedure, however, we selected a method that results in less shared wires, as it produced tress with smaller delays. Therefore, this implementation, called DAHHK from now on, has better delay results and yields to higher wire lengths when compared to the results obtained from (ALPERT, C. et al, 2006).

The P-Trees are configured to have required arrival times of 0 ps in the critical sinks, while the remaining sinks have no timing requirements, and are generated from the executable provided by the authors.

All these algorithms enable the user to choose the best tree considering the wire length and delay trade-off. In all cases, we pick the best delay configuration, disregarding wire length. We claim that the most critical nets of the circuit must have minimum possible delay, and congestion will not be strongly affected since these nets represent a small portion of the wires.

Table 5.3: Delay Comparison of AMAZE to DAHHK and P-Trees in a (300 μm $\times$ 300 μm) area

Alg	#nodes	3		5		7			9			11		
	#critical	1	all	1	all	1	3	5	1	3	5	1	3	5
AMAZE	WL (mm)	98	98	447	447	565	565	565	666	666	666	741	741	741
WL	Delay (ps)	45	54	85	109	120	161	173	165	222	235	194	260	274
AMAZE	WL (mm)	391	392	589	788	735	1078	1218	841	1280	1407	919	1320	1465
$plf = 0$	Delay (ps)	30	40	32	49	34	50	59	33	52	66	32	51	69
AMAZE	WL (mm)	391	391	589	786	735	1059	1200	841	1287	1400	919	1305	1454
$plf = 1$	Delay (ps)	30	40	32	48	34	49	57	33	51	64	32	51	67
DAHHK	WL (mm)	297	297	538	540	784	801	802	971	1019	1025	1202	1232	1234
	Delay (ps)	45	53	53	76	56	78	87	55	81	89	55	76	87
P-Trees	WL (mm)	391	391	642	730	863	1011	1051	1104	1262	1281	1314	1459	1504
	Delay (ps)	30	40	37	48	41	53	56	45	57	60	46	57	60
AMAZE	WL	-32%	-32%	-9%	-49%	6%	-32%	-50%	13%	-26%	-37%	24%	-6%	-18%
vs DAHHK	Delay	32%	26%	38%	39%	40%	37%	34%	40%	36%	28%	42%	32%	22%
AMAZE	WL	0%	0%	8%	-8%	15%	-5%	-14%	24%	-2%	-9%	30%	11%	3%
vs P-Trees	Delay	0%	0%	14%	1%	19%	8%	-2%	27%	11%	-7%	30%	11%	-12%

In our experiments we have used wire resistance and capacitance values from an actual fabrication line and source resistance and sink capacitance values from libraries currently used in industrial designs. For the source resistance we chose a strong cell, since it will be driving a critical net. This is, in fact, a favorable scenario for the AMAZE algorithm. By increasing the driver resistance by 30$\times$ the AMAZE improvement over DAHHK went down by approximately 1-8% (proportional to the size of the tree) while the improvement over P-Trees went down by approximately 0-4%. With this increased driver resistance, the gate delay (driver resistance times total capacitance) was roughly around 10-25% of the whole delay. In order to compute the Elmore delay, we used a 3Π RC circuit to model the wires. The validity of the data obtained from the delay model was confirmed by a set of electrical simulations performed on sample trees.

Tables 5.3 and 5.4 report the experimental results. On table 5.3 the trees where randomly generated in a window of 300 μ m $\times$ 300 μ m. On table 5.4 the window is reduced for 100 μ m $\times$ 100 μ m while we excluded 5 critical pins configurations. Additionally, on table 5.4 only AMAZE with path length factor set to 1 is used.

The following facts can be observed from Table 5.3:

- The AMAZE best setting for delay is AMAZE with path length factor (plf) 1.
- AMAZE scales very well with the addition of a non-critical sink. Note that the delay of configurations with one critical sink is always close to 30ps.
- AMAZE improvement is better (in delay) for cases with 1 critical sink. It does not scale well with the addition of other critical sinks. It delivers good results compared to P-Trees and DAHHK with up to 3 critical sinks.
- AMAZE delay is consistently better than DAHHK (ranging from 26% to 42% in average).
- AMAZE delay for 1 or 3 critical sinks is consistently better than P-Trees (ranging from 1% to 30% in average).
- Overall AMAZE does better on large trees.

On table 5.4 we can observe similar advantages of AMAZE to P-Trees and DAHHK algorithms.

Table 5.4: Delay Comparison of AMAZE to DAHHK and P-Trees in a (100 μm $\times$ 100 μm) area

Alg	# nodes	3		5		7		9		11	
	# critical	1	3	1	3	1	3	1	3	1	3
AMAZE	WL (mm)	98	98	152	152	188	188	221	222	244	244
WL	Delay (ps)	14	16	26	34	39	50	48	68	67	91
AMAZE	WL (mm)	126	125	200	275	243	344	276	395	313	441
$plf = 1$	Delay (ps)	10	12	10	15	11	15	10	16	11	18
DAHHK	WL (mm)	98	98	184	186	255	260	330	345	414	432
	Delay (ps)	14	16	17	24	18	25	16	25	19	27
P-Trees	WL (mm)	125	125	223	247	311	344	382	435	490	540
	Delay (ps)	10	12	12	15	13	16	13	17	14	18
AMAZE	WL (mm)	-28%	-28%	-9%	-48%	5%	-32%	16%	-14%	24%	-2%
vs DAHHK	Delay (ps)	30%	26%	37%	37%	40%	37%	40%	37%	42%	33%
AMAZE	WL (mm)	-1%	0%	10%	-11%	22%	0%	28%	9%	36%	18%
vs P-Trees	Delay (ps)	0%	0%	11%	3%	18%	5%	26%	8%	20%	-2%

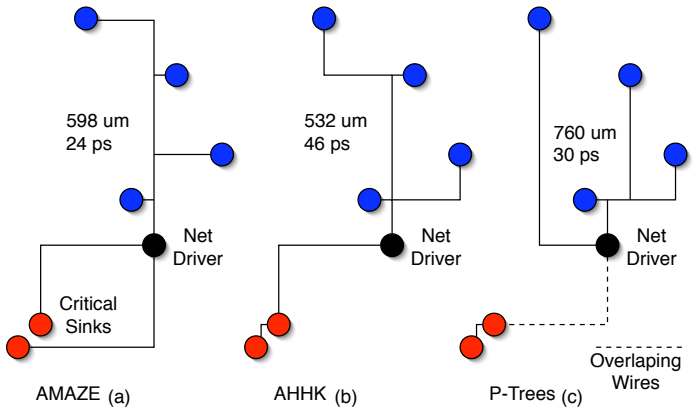


Figure 5.13: The best tree for delay of (a) AMAZE, (b) DAHHK and (c) P-Trees.

Figure 5.13 compares nets produced by the 3 algorithms. We observe that, in the AMAZE algorithm, the path to the critical sinks has minimum length and minimum sharing, while the rest of the tree is optimized for wire length. The best effort of the DAHHK algorithm found the minimum path length to all sinks (arborescence), however it did not help much for the delay of the critical sinks, since the path is fully shared by both sinks. For the P-Trees algorithm, among 35 different topologies evaluated, the one that was best for minimizing delay to critical sinks have separated wires to the critical sinks. The P-Trees's drawback that impacted the delay is the fact that the overall wire length is too large (affecting the product of driver resistance per total capacitance). Another drawback is the generation of overlapped wires, that need to be somehow resolved in actual routing.

5.7.3 Steiner Trade-off analysis

We provide a comparison of the wire length and delay trade-off produced by the three studied algorithms (ours, DAHHK and P-Trees) in Figure 5.14. The delay and wire length ranges are average numbers for 7 pin nets with one critical sink. The best delay generally leads to the worst wire length (*wl*). The algorithms will trade-off *wl* for delay in this range.

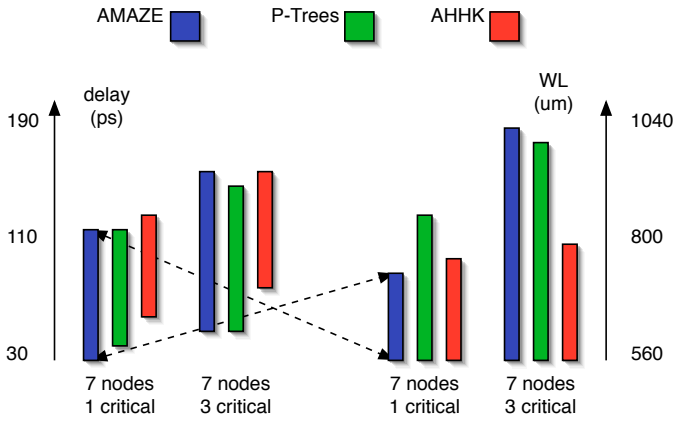


Figure 5.14: Delay and wire length range of the studied algorithms. Within this range you can trade WL for delay. The worst case wire length leads to best delay vice-versa.

The following facts can be observed from Figure 5.14.

- In the case of one critical sink, AMAZE worst case wire length is better than the other studied algorithms, while delay obtained is the best.
- AMAZE and P-Trees have a wider range to trade-off wire length and delay, since they can control the amount of sharing more effectively.
- The addition of a critical sink increases significantly the trade-off range, since extra wire is inserted to isolate the wire.

5.7.4 Blockage analysis

Since AMAZE is a maze router it naturally handles blockages. As a potential problem, we observe that our critical wire isolation approach (sharing factor) will be less effective if a blockage is in the middle of the way. In such case, AMAZE will select the possibility with minimum length and this might lead to a worse topology of the overall tree since the wires will not be shared. We enforce that AMAZE should be executed with varied sharing factors (path length factor can be set to 1) to avoid extremely long wires.

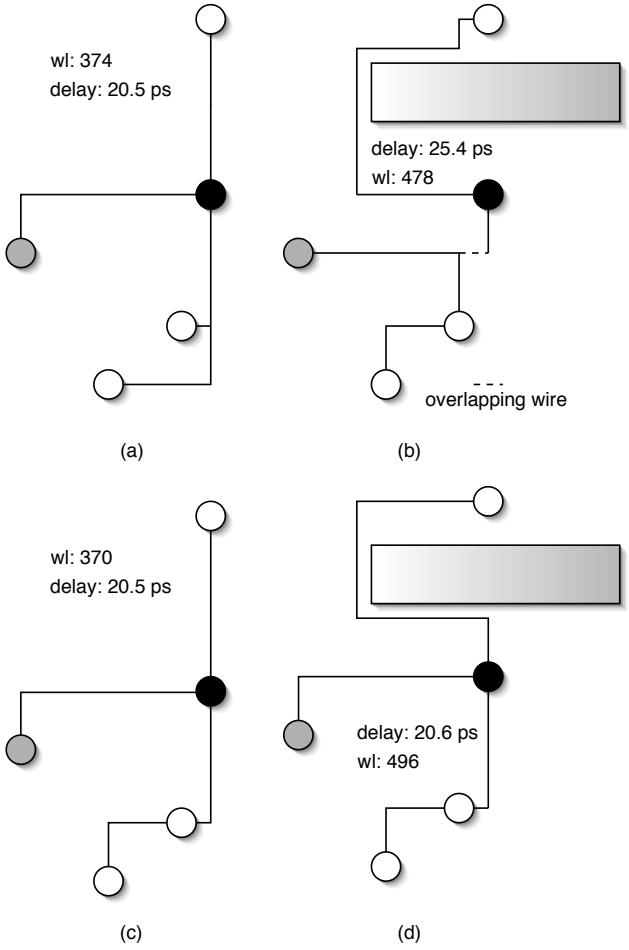


Figure 5.15: Example of blockage handling with both AMAZE and P-Trees; (a) and (b) refer to P-Trees solution without and with a blockage respectivelly; (c) and (d) refer to the AMAZE.

We compared our algorithm to the P-Trees. Since DAHHK trees cannot handle blockage directly we left it out of the comparison. Our experimental setup was as follows: We generated trees ranging from 5 to 9 pins, where 1 to 3 of the sinks were critical. For each configuration we generate 100 trees randomly with one blockage. Given the Hannan grid of the tree, sliced horizontally and vertically in the Hannan points, the blockage is constrained to interrupt at least one of the slices. After that, we run the AMAZE (with $plf = 1$) and P-Trees algorithms for each of the trees picking the best tree for delay to the critical sinks. We then compare the obtained trees with and without the blockage. If the solution of both algorithms are identical to the non-blockage version, we discard the

Table 5.5: Steiner delay comparison with blockages between AMAZE and P-Trees

Alg	# nodes	5		7		9	
	# critical	1	2	1	2	1	2
AMAZE $pfl = 1$	WL (μm)	595	709	752	935	851	1024
no block	Delay (ps)	33	39	35	43	41	53
AMAZE $pfl = 1$	WL (μm)	646	751	774	962	870	1047
with block	Delay (ps)	36	43	36	44	43	53
P-Trees	WL (μm)	672	692	904	1036	1163	1261
no block	Delay (ps)	36	43	43	53	45	54
P-Trees	WL (μm)	735	733	958	1046	1175	1274
with block	Delay (ps)	38	46	45	53	48	55
AMAZE	WL (μm)	-8.5%	-5.7%	-3.0%	-2.7%	-2.1%	-2.2%
Degradation	Delay (ps)	8.5%	10.2%	-3.1%	-3.6%	-0.5%	-4.0%
P-Trees	WL (μm)	-9.3%	-5.95%	-6.0%	-1.0%	-1.0%	-0.9%
Degradation	Delay (ps)	-5.2%	-6.8%	-6.5%	-1.06%	-5.49%	-3.13%
Impr. over	WL (μm)	12.1%	-2.37%	19.1%	8.0%	25.96%	17.82%
P-Trees	Delay (ps)	6.0%	5.8%	20.4%	16.6%	31.4%	21.91%

tree from the experiment set. We observed that half of the generated trees fall into this criteria.

Table 5.5 presents all the experimental results where the following facts can be observed: (1) AMAZE is consistently better than P-Trees in average for all tested configurations, from 5% to 32% in delay. (2) Besides the better delay of critical sinks, AMAZE could deliver better wirelength up to 26%. (3) Both algorithms suffer from a degradation for the blockage insertion. AMAZE degrades less in the cases with one critical sink and more than 5 pins.

Finally we present an example comparing the same trees with AMAZE and P-Trees in figure 5.15. In this example, the non-blocked trees are practically the same (figure 5.15.a and 5.15.c) except for a small piece of wire that is not enough to increase the delay to the critical sink. In the presence of the blockage, AMAZE simply made a minimal detour while the P-Trees ended up adding more capacitance to the critical path (note that a part of the path from the driver is composed of overlapping wires, but not all of it).

To conclude with, we can observe that AMAZE has some important features that help in the blockage handling: (1) The critical sinks are routed first. This fact prevents that blockage detours made by wires to non-critical sinks eventually change the topology of the critical connection. This feature is not present in the P-Trees, as show in the graphical example of figure 5.15.a and 5.15.b. (2) Each sink is routed separately, isolating the detour to the blocked branch only; (3) Even in an extreme case (a very long detour is needed), AMAZE will not change the strategy of isolating the critical wire. It might happen thought that wirelength increases too much. In these cases, the solution is to relax the isolation of critical wires by playing with the sharing factor.

5.8 Application to Timing Driven Routing

The techniques presented in this thesis called *sharing factor*, *path length factor* and *biasing* provide means of controlling, respectively, the amount of sharing of critical wires,

path length of critical wires and overall wire length. By understanding the impact of those parameters to the delay of critical elements in a circuit and playing with them it is possible to obtain a variety of tree topologies. For instance, an arborecence can be built simply by setting all sinks to critical and the path length factor to 1. A star tree can be built using the sharing factor 0. All intermediate values provide topologies that trade-off sharing of critical wires, path length and wire length in such a way that an appropriate topology is found.

In our experiments, we target at obtaining a path with minimum delay. For that, our best setting is to fix the path length factor to one (minimum path length) and vary the sharing factor from 0 to 1 in steps of 0.25. This way, 5 different topologies are generated and we simply pick the best for delay.

The algorithm is also flexible to handle other applications. Arborecences can be generated by setting all sinks as critical and the path length factor to 1. By playing with intermediate values of this factor and providing multiple levels of criticality (as more detailed in algorithm 10), delays to less critical sinks can be improved. This methodology can be applied to obtain the best tree that satisfies a certain slack.

As regarding speed, our algorithm also provides means of obtaining very fast routing combining existing techniques such as heuristic search and Hannan grid with some new ones such as an improved h function that predicts vias, an improved data structure for the open nodes with constant time insertion and log time access, an auxiliary openlist to delay computation of f function for some *expensive* cases, and a method for fast expansion of nodes in the critical path. To evaluate run time, we performed 415 AMAZE runs on 5, 7, 9 and 15 pin nets and achieved respectively 0.61s, 0.94s, 1.24s and 2.45s. We also performed the same experiment inserting one random blockage as described in the previous section obtaining, respectively, 0.69s, 1.01s, 1.35s and 2.7s. There is a small degradation with the blockage since the heuristic based on manhattan distance will not be exact in some cases, demanding some more nodes to be expanded in order to assure the minimal path. The hardware platform used was a Mac Dual G5 with 1GB of RAM. Due to the speed of our results, we conclude that AMAZE can be used for higher level estimation such as placement and for routing as well.

5.9 AMAZE for 3D circuits

AMAZE, as a flexible routing algorithm, can be applicable to the routing of 3D circuits. We leave the experiments as well as the research for the advantages and contributions it could bring to the 3D routing community as future work. Figure 5.16 illustrates an hypothetical application of the algorithm in a situation where all critical connections remain on the same tier. Note that on this situation AMAZE will provide easily the isolation of the critical connections. Additionally, the cost of wires on the Z axis can be modeled accordingly; note that the addition of extra costs will not harm AMAZE run time since they can be incorporated in the h function.

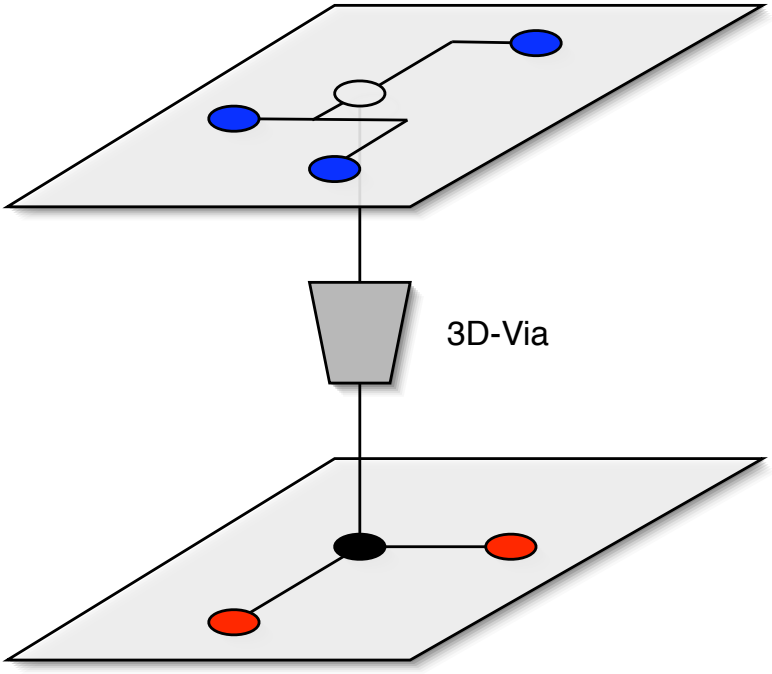


Figure 5.16: AMAZE algorithm applied to 3D Routing.

6 CONCLUDING REMARKS

It is well known that the connections of a circuit is a limiting factor for timing, power and area of the circuit. This work presented methods that are applicable for wire length reduction of existing and future designs.

6.1 3D Placement

A 3D Placement algorithm is presented. It is well known that 3D Placement is a way to reduce wire lengths, but since this area is brand new it lacks research. There is a considerable amount of issues to be solved by 3D Placement and some of them are completely ignored by a large amount of the existing literature. This thesis addressed some of 3D-Via related issues in a complete placement flow called Z-Place.

By analyzing the available technologies for 3D integration we observed that there is a variety of 3D-Via pitches and lengths, which impacts cell placement significantly. On the large pitch technologies, the appropriate methodology is to minimize the 3D-Vias, while on small pitches there is room to trade 3D-Vias for wire length. For these reasons, we concluded that the cell placer must be adaptable to the scenario provided by the technology in order to take full advantage of it.

Consider the case of large pitch 3D-Vias first. Z-Place has a novel technique for manipulating the I/O pins and placing them into 3D that favors the 3D-Via minimization. By obtaining its logic distance with BFS searches, Z-Place can improve a min-cut partition of the netlist and consequently obtain less 3D-Vias. This improvement comes with the cost of CPU time.

Considering the case of small pitch 3D-Vias, Z-Place uses the I/O pins distribution as a starting point to its Quadratic Placement engine that is a very scalable and largely used algorithm. The global placement engine optimizes 3D wire length, which considers vertical costs as well. While wire length optimization can introduce a large number of 3D-Vias, our algorithm keeps track of 3D-Via count as well, considering that some kinds of 3D-Vias consume resources differently.

More algorithms are also proposed for the Z-Place flow. A 3D-Via placement algorithm places 3D-Vias such that they do not overlap with each other. The new algorithm has very good speed and low wire length overhead. Compared to an existing ILP approach, it achieves a similar quality with 2 orders of magnitude advantage on run time.

We also proposed a technique to address timing. The idea is to keep cells of a same critical path on a same tier. In order to do that, an artificial pin connecting all cells in a critical path is inserted in the netlist. The connection has a high weight for the Z axis in contrast with a low (or zero) weight for X or Y. Experimental results on a benchmark set of 14 circuits placed into 2,3 and 4 tiers demonstrated the effectiveness of the approach by

reducing the number of critical 3D-Vias from several hundreds to zero in all benchmarks. The total wire length and number of 3D-Vias were only slightly affected.

Overall, the 3D Placement led to the following conclusions:

- In average there is a improvement in the order of 10% when adding one tier.
- On 2D, Fastplace and Z-Place have similar WL results.
- Compared to an existing algorithm, we have a small average advantage (2%); on the other hand, the compared algorithm supports only 2 tiers face-to-face.
- The 2 tier technology have the best configuration considering cost and reliability; it is able to provide at least 10% WL improvement.

In general, the experimental results demonstrate a trade-off between WL and cost of the circuit, since the addition of more tiers improve WL but introduce reliability problems and increase the cost. If one is able to pay the price to improve performance, a 3D solution is a simple and effective choice.

6.2 Routing

We addressed the application of several techniques in a path search based algorithm called AMAZE so that it becomes able to compete with state-of-the-art methods for building Steiner Trees. For delay, we have presented important properties of such trees under the Elmore delay model. AMAZE has also the ability to trade-off wire length for delay, minimizing the delay for critical sinks while reducing the overall wire length for the rest of the tree. Our biasing technique is the key to achieve *wl* reduction by maximizing wire sharing. On the other hand, repulsive biasing, path length and sharing factors were introduced to isolate critical paths so that delay to the identified critical sinks is minimized.

We outperformed algorithms used in the industry and in the state-of-the-art academic research, such as our implementation of AHHK (by 25%-40% in average) and P-Trees (by 1%-30% in average) (for P-Trees we downloaded the official binary). We also observed that a wide range of topologies are available for the AMAZE algorithm, which is able to find the appropriate tree according to the needs of the net being routed. On the blockage analysis, we observed that AMAZE maintained a good behavior and outperformed P-Trees from 5% to 32%. Congestion could also be incorporated in AMAZE algorithm by providing costs to the routing graph. Cost functions should be carefully used since they will eliminate critical ties that eventually happen during the A* search, affecting the benefits obtained in this thesis by exploring those ties.

Novel properties of path search that can be used to obtain better run time were demonstrated. Those properties include an open list with a bounded number of positions that can be read in constant time. By cleverly combining those properties with heuristic search, the routing algorithm can take great advantage of the heuristic estimator (behaving like a direct DFS search) and have run times compatible with heuristic steiner tree algorithms that are considered fast.

We believe that due to their speed and quality, our algorithms will find applications both in obtaining wiring estimates that are required during early design like placement, as well as during the actual routing. For example, applications such as global routing could potentially benefit from the added flexibility and trade-off provided by our algorithm. Designs dominated by a small number of highly critical paths could benefit greatly by optimizing these paths for delay at the expense of WL in the rest of the design.

REFERENCES

3D ICs Industry Summary. Available at: <http://www.tezzaron.com>. Visited on: June 2007.

ABABEI, C. et al. Placement and Routing in 3D Integrated Circuits. **Design and Test of Computers**, [S.l.], p.520–531, Nov.-Dec 2005.

ABABEI, C.; MOGAL, H.; BAZARGAN, K. Three-Dimensional Place and Route for FPGAs. In: CONFERENCE ON ASIA SOUTH PACIFIC DESIGN AUTOMATION, ASP-DAC, 2005. **Proceedings...** [S.l.]: IEEE Press, 2005.

AGNIHOTRI, A. R.; ONO, S.; MADDEN, P. H. Recursive bisection placement: feng shui 5.0 implementation details. In: INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, ISPD, 2005, San Francisco, CA, USA. **Proceedings...** New York: ACM Press, 2005. p.230–232.

ALPERT, C. et al. Prim-Dijkstra Tradeoffs for Improved Performance-Driven Routing Tree Design. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v.14, p.890–896, 1995.

ALPERT, C. et al. Quadratic placement revisited. In: DESIGN AUTOMATION CONFERENCE, DAC, 1997. **Proceedings...** New York: ACM Press, 1997. p.752–757.

ALPERT, C. et al. Timing-driven Steiner trees are (practically) free. In: DESIGN AUTOMATION CONFERENCE, DAC, 2006, San Francisco, CA, USA. **Proceedings...** New York: ACM Press, 2006. p.389–392.

BALAKRISHNAN, K. et al. Wire congestion and thermal aware 3D global placement. In: ASIA SOUTH PACIFIC DESIGN AUTOMATION, ASP-DAC, 2005, Shanghai, China. **Proceedings...** New York: ACM Press, 2005. p.1131–1134.

BANERJEE, K. et al. 3D-ICs: a novel chip design for improving deep submicrometer interconnect performance and systems on-chip integration. **Proceedings of IEEE**, New York, v.89, p.602–633, 2001.

BOESE, K. D.; KAHNG, A. B.; MCCOY, B. A. Near Optimal Critical Sink Routing. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v.14, p.1417–1436, 1995.

BORAH, M.; OWENS, R. M.; IRWIN, M. J. A fast algorithm for minimizing the Elmore delay to identified critical sinks. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, [S.l.], v.16, p.753–759, 1997.

BRENNER, U.; VYGEN, J. Legalizing a placement with minimum total movement. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v.23, 2004.

CALDWELL, A. Personal conversation about using the information of previous searches to improve delay of subsequent searches. 1997.

CALDWELL, A. E.; KAHNG, A. B.; MARKOV, I. L. Can recursive bisection alone produce routable placements? In: DESIGN AUTOMATION CONFERENCE, DAC, 2000, Los Angeles, CA, USA. **Proceedings...** New York: ACM Press, 2000. p.477–482.

CHANG, C.; CONG, J. Multilevel Global Placement With Congestion Control. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v.22, April 2003.

CHOU, Y.-C.; LIN, Y.-L. Effective Enforcement of Path-Delay Constraints in Performance-Driven Placement. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v.21, p.15–21, 2002.

CONG, J. et al. Provably Good Performance Driven Routing. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v.11, p.739–752, 1992.

CONG, J.; KOH, C.-K.; MADDEN, P. Interconnect layout optimization under higher order RLC model for MCM designs. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v.20, p.1455–1463, 2001.

CONG, J.; LEUNG, K.-S.; ZHOU, D. Performance-driven interconnect design based on distributed RC delay model. In: DESIGN AUTOMATION CONFERENCE, DAC, 1993, Dallas, Texas, USA. **Proceedings...** New York: ACM Press, 1993. p.606–611.

CONG, J.; WEI, J.; ZHANG, Y. A Thermal-Driven Floorplanning Algorithm for 3D ICs. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 2004, San Jose, CA, USA. **Proceedings...** [S.l.: s.n.], 2004.

CONG, J.; ZHANG, Y. Thermal via planning for 3-D ICs. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 2005, San Jose, CA. **Proceedings...** Washington: IEEE Computer Society, 2005. p.745–752.

DAS, S.; CHANDRAKASAN, A.; REIF, R. Design tools for 3-D integrated circuits. In: ASIA SOUTH PACIFIC DESIGN AUTOMATION, ASP-DAC, 2003, Kitakyushu, Japan. **Proceedings...** New York: ACM Press, 2003. p.53–56.

DAS, S.; CHANDRAKASAN, A.; REIF, R. Calibration of Rent's rule models for three-dimensional integrated circuits. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, New York, v.12, p.359–366, 2004.

DAS, S.; CHANDRAKASAN, A.; REIF, R. Timing, energy, and thermal performance of three-dimensional integrated circuits. In: GREAT LAKES SYMPOSIUM ON VLSI, GLSVLSI, 2004, Boston, MA, USA. **Proceedings...** New York: ACM Press, 2004. p.338–343.

DAS, S. et al. Technology, performance, and computer-aided design of three-dimensional integrated circuits. In: INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, ISPD, 2004, Phoenix, Arizona, USA. **Proceedings...** New York: ACM Press, 2004. p.108–115.

DAVIS, W. et al. Demystifying 3D ICs: the pros and cons of going vertical. **Design and Test of Computers**, [S.l.], p.498–510, Nov.-Dec. 2005.

DENG, Y.; MALY, W. 2.5-Dimensional VLSI System Integration. **IEEE Transactions on Very Large Integration (VLSI) Systems**, New York, v.13, p.668–677, June 2005.

DENG, Y.; MALY, W. P. Interconnect characteristics of 2.5-D system integration scheme. In: INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, ISPD, 2001, Sonoma, CA, USA. **Proceedings...** New York: ACM Press, 2001. p.171–175.

DUECK, G.; SCHEUER, T. Threshold Accepting: a general purpose optimization algorithm appear superior to simulated annealing. **Journal of Computational Physics**, [S.l.], p.161–175, 1990.

DUTT, S.; ARSLAN, H. Efficient timing-driven incremental routing for VLSI circuits using DFS and localized slack-satisfaction computations. In: DESIGN, AUTOMATION AND TEST IN EUROPE, DATE, 2006. **Proceedings...** [S.l.]: IEEE, 2006. p.768–773.

EISENMANN, H.; JOHANNES, F. M. Generic global placement and floorplanning. In: DESIGN AUTOMATION CONFERENCE, DAC, 1998, San Francisco, CA, USA. **Proceedings...** New York: ACM Press, 1998. p.269–274.

ELMORE, W. C. The Transient Response of Damped Linear Network with Particular Regard to Wideband Amplifiers. **Journal of Applied Physics**, [S.l.], v.19, p.55–63, 1948.

FLACH, G.; HENTSCHKE, R.; REIS, R. Algorithms for improvement of RotDL router. In: SOUTH SYMPOSIUM ON MICROELECTRONICS, SIM, 2004. **Proceedings...** Ijui: UNIJUI, 2004. p.65–70.

FORMAL Grammar Definition. Available at: <http://http://en.wikipedia.org/wiki/Formal_grammar/>. Visited on: Jan. 2007.

GAREY, M. R.; JOHNSON, D. S. The Rectilinear Steiner Tree is NP-Complete. **SIAM Jour. on App. Math**, [S.l.], v.23, p.826–834, 1977.

GAREY, M. R.; JOHNSON, D. S. **Computers and Intractability**. [S.l.]: W. H. Freeman, 1979.

GOPLEN, B.; SAPATNEKAR, S. Efficient Thermal Placement of Standard Cells in 3D ICs using a Force Directed Approach. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 2003. **Proceedings...** Washington: IEEE Computer Society, 2003. p.86.

GOPLEN, B.; SAPATNEKAR, S. Thermal via placement in 3D ICs. In: INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, ISPD, 2005, San Francisco, CA, USA. **Proceedings...** New York: ACM Press, 2005. p.167–174.

GRIFFITH, J. et al. Closing the gap: near-optimal steiner trees in polynomial time. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v.13, p.1351–1365, 1994.

GUPTA, S. et al. **Techniques for Producing 3D ICs with High-Density Interconnect**. Available at: <http://www.tezzaron.com/>. Visited on: Aug. 2005.

HALL, K. M. An r-dimensional quadratic placement algorithm. **Management Science**, [S.l.], v.17, p.219–229, 1970.

HANNAN, M. On Steiner Problem with Rectilinear Distance. **SIAM Jour. on App. Math**, New York, v.30, p.255–265, 1992.

HART, P. E.; NILSSON, N. J.; RAPHAEL, B. A Formal Basis for the Heuristic Determination of Minimum Cost paths. **IEEE Transactions on System Science and Cybernetics**, [S.l.], v.SSC-4, p.100–107, 1968.

HEALY, M. et al. Multiobjective Microarchitectural Floorplanning for 2-D and 3-D ICs. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, [S.l.], v.26, p.38–52, 2007.

HENTSCHKE, R. **Algoritmos para o Posicionamento de Células em Circuitos VLSI**. 2002. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

HENTSCHKE, R. et al. Quadratic placement for 3d circuits using z-cell shifting, 3d iterative refinement and simulated annealing. In: INTEGRATED CIRCUITS AND SYSTEMS DESIGN, SBCCI, 2006, Ouro Preto, MG, Brazil. **Proceedings...** New York: ACM Press, 2006. p.220–225.

HENTSCHKE, R. et al. 3D-Vias Aware Quadratic Placement for 3D VLSI Circuits. In: IEEE COMPUTER SOCIETY ANUAL SYMPOSIUM ON VLSI, ISVLSI, 2007, Porto Alegre, RS, Brazil. **Proceedings...** Los Alamitos: IEEE Computer Society, 2007. p.67–72.

HENTSCHKE, R. et al. Maze routing steiner trees with effective critical sink optimization. In: INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, ISPD, 2007, Austin, Texas, USA. **Proceedings...** New York: ACM Press, 2007. p.135–142.

HENTSCHKE, R. F.; JOHANN, M.; REIS, R. A Study on the Performance of Fast Initial Placement Algorithms. In: INTERNATIONAL CONFERENCE ON VERY LARGE SCALE INTEGRATION, VLSI-SOC, 2003, Darmstadt, Germany. **Proceedings...** Darmstadt: Technische Universitat Darmstadt, 2003. p.204–209.

HENTSCHKE, R. F.; REIS, R. Improving Simulated Annealing Placement by Applying Random and Greedy Mixed Perturbations. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, SBCCI, 2003. **Proceedings...** Los Alamitos: IEEE Computer Society, 2003. p.267–272.

HENTSCHKE, R. F.; REIS, R. Plic-Plac: a novel constructive algorithm for placement. In: INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, ISCAS, 2003. **Proceedings...** [S.l.]: IEEE, 2003. p.461–464.

HENTSCHKE, R.; JOHANN, M.; REIS, R. An Algorithm for I/O Partitioning Targeting 3D Circuits and Its Impact on 3D-Vias. In: INTERNATIONAL CONFERENCE ON VERY LARGE SCALE INTEGRATION, VLSI-SOC, 2006, Nice, France. **Proceedings...** [S.l.: s.n.], 2006.

HENTSCHKE, R.; REIS, R. A 3D-Via Legalization Algorithm for 3D VLSI Circuits and its Impact on Wire Length. In: INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, ISCAS, 2007. **Proceedings...** Los Alamitos: IEEE Computer Society, 2007. p.2036–2039.

HONG, X. et al. Performance-driven Steiner tree algorithm for global routing. In: DESIGN AUTOMATION CONFERENCE, DAC, 1993, Dallas, Texas, USA. **Proceedings...** New York: ACM Press, 1993. p.177–181.

HOU, H.; HU, J.; SAPATNEKAR, S. Non-Hanan Routing. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v.18, p.436–444, 1999.

HU, B.; ZENG, Y.; MAREK-SADOWSKA, M. mFAR: fixed-points-addition-based vlsi placement algorithm. In: INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, ISPD, 2005, San Francisco, CA, USA. **Proceedings...** New York: ACM Press, 2005. p.239–241.

HUA, H. et al. Exploring compromises among timing, power and temperature in three-dimensional integrated circuits. In: DESIGN AUTOMATION CONFERENCE, DAC, 2006, San Francisco, CA, USA. **Proceedings...** New York: ACM Press, 2006. p.997–1002.

HWANG, C.; PEDRAM, M. Timing-driven placement based on monotone cell ordering constraints. In: ASIA SOUTH PACIFIC DESIGN AUTOMATION, ASP-DAC, 2006, Yokohama, Japan. **Proceedings...** New York: ACM Press, 2006. p.201–206.

ISPD04 - IBM Standard Cell Benckmarks with Pads. Available at: <http://www.public.iastate.edu/∼nataraj/ISPD04_Bench.html>. Visited on: Jan. 2008.

JOHANN, M.; REIS, R. Net by net routing with a new path search algorithm. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, SBCCI, 2000, Manaus. **Proceedings...** Los Alamitos: IEEE Computer Society, 2000. p.144–149.

KAHNG, A. B.; REDA, S.; WANG, Q. APlace: a general analytic placement framework. In: INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, ISPD, 2005, San Francisco, CA, USA. **Proceedings...** New York: ACM Press, 2005. p.233–235.

KAHNG, A. B.; WANG, Q. An analytic placer for mixed-size placement and timing-driven placement. In: INTERNATIONAL CONFERENCE ON COMPUTER AIDED DESIGN, ICCAD, 2004. **Proceedings...** [S.l.: s.n.], 2004. p.565–572.

KARYPIS, G. et al. Multilevel Hypergraph Partitioning: applications in vlsi domain. **IEEE Transactions on Very Large Integration (VLSI) Systems**, New York, v.7, p.69–79, March 1999.

KASTNER, R.; SARRAFZADEH, M. **Labyrinth**. Available at <http://www.ece.ucsb.edu/∼kastner/labyrinth>. Visited on: Jan. 2005.

KAYA, I. et al. Wirelength Reduction Using 3-D Physical Design. In: INTERNATIONAL WORKSHOP ON INTEGRATED CIRCUIT AND SYSTEM DESIGN - POWER AND TIMING MODELING, OPTIMIZATION AND SIMULATION, PATMOS, 2004, Santorini, Greece. **Proceedings...** [S.l.: s.n.], 2004.

KHANG, A.; ROBINS, G. **On Optimal Interconnects for VLSI**. Boston, MA: Kluwer Academic, 1995.

KHATKHATE, A. et al. Recursive bisection based mixed block placement. In: INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, ISPD, 2004, Phoenix, Arizona, USA. **Proceedings...** New York: ACM Press, 2004. p.84–89.

KIRKPATRICK, S.; GELATT, C. D.; VECCHI, M. P. Optimization by Simulated Annealing. **Science**, [S.l.], v.220, n.4598, p.671–680, 1983.

KONG, T. T. A novel net weighting algorithm for timing-driven placement. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 2002. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p.172–176.

LAM, J.; DELOSME, J.-M. Performance of a new annealing schedule. In: DESIGN AUTOMATION CONFERENCE, DAC, 1988, Atlantic City, New Jersey, USA. **Proceedings...** Los Alamitos: IEEE Computer Society Press, 1988. p.306–311.

LEE, C. An algorithm for path connections and its applications. **IRE Transactions on Electronics Computers**, [S.l.], p.346–365, September 1961.

LI, Z. et al. Efficient thermal-oriented 3D floorplanning and thermal via planning for two-stacked-die integration. **ACM Trans. Des. Autom. Electron. Syst.**, [S.l.], v.11, n.2, p.325–345, 2006.

LILLIS, J. et al. New performance driven routing techniques with explicit area/delay tradeoff and simultaneous wire sizing. In: DESIGN AUTOMATION CONFERENCE, DAC, 1996, Las Vegas, NV, USA. **Proceedings...** New York: ACM Press, 1996. p.395–400.

LIM, S. K. Physical design for 3D system on package. **Design and Test of Computers**, [S.l.], p.498–510, Nov.-Dec. 2005.

LIU, G. et al. 3D placement algorithm considering vertical channels and guided by 2D placement solution. In: INTERNATIONAL CONFERENCE ON ASIC, ASICON, 2005. **Proceedings...** [S.l.: s.n.], 2005. p.24–27.

LUO, T.; NEWMARK, D.; PAN, D. Z. A new LP based incremental timing driven placement for high performance designs. In: DESIGN AUTOMATION CONFERENCE, DAC, 2006, San Francisco, CA, USA. **Proceedings...** New York: ACM Press, 2006. p.1115–1120.

MANDOIU, I. I.; VAZIRANI, V. V.; GANLEY, J. L. A new heuristic for rectilinear Steiner trees. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 1999, San Jose, CA, USA. **Proceedings...** Piscataway: IEEE Press, 1999. p.157–162.

MOORES Law: made real by intel innovation. Available at: <http://www.intel.com/technology/silicon/mooreslaw/>. Visited on: Aug. 2005.

NAM, G. et al. The ISPD2005 placement contest and benchmark suite. In: INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, ISPD, 2005, San Francisco, CA, USA. **Proceedings...** New York: ACM Press, 2005. p.216–220.

OBENAUS, S.; SZYMANSKI, T. Gravity: fast placement for 3-d vlsi. **ACM Transacions on Design Automation of Electronic Systems**, New York, v.8, p.69–79, March 1999.

OBERMEIER, B.; JOHANNES, F. M. Temperature-aware global placement. In: ASIA SOUTH PACIFIC DESIGN AUTOMATION, ASP-DAC, 2004, Yokohama, Japan. **Proceedings...** Piscataway: IEEE Press, 2004. p.143–148.

OBERMEIER, B.; RANKE, H.; JOHANNES, F. M. Kraftwerk: a versatile placement approach. In: INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, ISPD, 2005, San Francisco, CA, USA. **Proceedings...** New York: ACM Press, 2005. p.242–244.

OGAWA, Y.; PEDRAM, M.; KUH, E. S. Timing-driven placement for general cell layout. In: INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, ISCAS, 1990. **Proceedings...** [S.l.: s.n.], 1990. p.872–876.

PATTI, R. Three-dimensional integrated circuits and the future of system-on-chip designs. **Proceedings of IEEE**, [S.l.], v.94, p.1214–1224, 2006.

PATTI, R. Personal Conversation Regarding Tezzaron Technology. 2006.

PRASITJUTRAKUL, S.; KUBITZ, W. J. A Timing-Driven Global Router for Custom Chip Design. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 1990. **Proceedings...** [S.l.: s.n.], 1990. p.48–51.

RAHMAN, A.; REIF, R. System-level performance evaluation of three-dimensional integrated circuits. **IEEE Transactions on Very Large Scale Integration (VLSI) Systems**, [S.l.], v.8, December 2000.

RAHMAN, A.; REIF, R. Thermal analysis of three-dimensional (3-D) integrated circuits (ICs). In: INTERNATIONAL INTERCONNECT TECHNOLOGY CONFERENCE, 2001. **Proceedings...** [S.l.: s.n.], 2001. p.157–159.

RAO, S. et al. The rectilinear steiner arborescence problem. **Algorithmica**, [S.l.], v.7, December 1992.

RIESS, B. M.; ETTELT, G. G. SPEED: fast and efficient timing driven placement. In: INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, ISCAS, 1995. **Proceedings...** [S.l.: s.n.], 1995. p.377–380.

ROY, J. et al. Capo: robust and scalable open-source min-cut floorplacer. In: INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, ISPD, 2005, San Francisco, CA, USA. **Proceedings...** New York: ACM Press, 2005. p.224–226.

S-W., H.; JAGANNATHAN, A.; LILLIS, J. Timing-driven maze routing. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v.19, p.234–241, 2000.

SAIT, S.; YOUSSEF, H.; MALEH, A. Fuzzy Simulated Evolution for Power and Performance Optimization Of VLSI Placement. In: INTERNATIONAL JOINT INNS-IEEE CONFERENCE ON NEURAL NETWORKS, IJCNN, 2001. **Proceedings...** [S.l.: s.n.], 2001.

SANTOS, G. B. V. dos. Personal conversations regarding timing driven steiner tree topologies. 2006.

SECHEN, C.; VICENTELLI, A. S. The Timberwolf Placement and routing package. **IEEE Journal of Solid State Circuits**, [S.l.], v.SSC-20, April 1985.

SHERWANI, N. A. **Algorithms for VLSI Physical Design Automation**. Norwell, MA, USA: Kluwer Academic Publishers, 1998.

SPINDLER, P.; JOHANNES, F. Fast and Robust Quadratic Placement Combined with an Exact Linear Net Model. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 2006. **Proceedings...** [S.l.: s.n.], 2006.

SUNTHARALINGAM, V. et al. Megapixel CMOS image sensor fabricated in three-dimensional integrated circuit technology. In: SOLID-STATE CIRCUITS CONFERENCE, ISSCC, 2005. **Proceedings...** [S.l.]: IEEE, 2005. v.1, p.356–357.

SWARTZ, W.; SECHEN, C. Timing driven placement for large standard cell circuits. In: DESIGN AUTOMATION CONFERENCE, DAC, 1995, San Francisco, CA, USA. **Proceedings...** New York: ACM Press, 1995. p.211–215.

TAGHAVI, T.; YANG, X.; CHOI, B.-K. Dragon2005: large-scale mixed-size placement tool. In: INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, ISPD, 2005, San Francisco, CA, USA. **Proceedings...** New York: ACM Press, 2005. p.245–247.

TEZZARON Homepage. Available at: <http://www.tezzaron.com>. Visited on: Aug. 2005.

TSA, C.-H.; KANG, S.-M. Cell-level placement for improving substrate thermal distribution. **IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems**, New York, v.19, February 2000.

VISWANATHAN, N.; PAN, M.; CHU, C. FastPlace 2.0: an efficient analytical placer for mixed-mode designs. In: ASIA SOUTH PACIFIC DESIGN AUTOMATION, ASP-DAC, 2006, Yokohama, Japan. **Proceedings...** New York: ACM Press, 2006. p.195–200.

VISWANATHAN, N.; PAN, M.; CHU, C. C.-N. FastPlace: an analytical placer for mixed-mode designs. In: INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, ISPD, 2005, San Francisco, CA, USA. **Proceedings...** New York: ACM Press, 2005. p.221–223.

WONG, E.; LIM, S. K. 3D floorplanning with thermal vias. In: DESIGN, AUTOMATION AND TEST IN EUROPE, DATE, 2006, Munich, Germany. **Proceedings...** Leuven: European Design and Automation Association, 2006. p.878–883.

XIU, Z.; RUTENBAR, R. A. Timing-driven placement by grid-warping. In: DESIGN AUTOMATION CONFERENCE, DAC, 2005, San Diego, CA, USA. **Proceedings...** New York: ACM Press, 2005. p.585–591.

XU, J. et al. An Efficient Hierarchical Timing-Driven Steiner Tree Algorithm for Global Routing. In: ASIA SOUTH PACIFIC DESIGN AUTOMATION, ASP-DAC, 2002. **Proceedings...** Washington: IEEE Computer Society, 2002. p.473.

YAN, H. et al. Via assignment algorithm for hierarchical 3D placement. In: INTERNATIONAL CONFERENCE ON COMMUNICATIONS, CIRCUITS AND SYSTEMS, 2005. **Proceedings...** [S.l.: s.n.], 2005. p.27–30.

YAN, T. et al. How does partitioning matter for 3D floorplanning? In: GREAT LAKES SYMPOSIUM ON VLSI, GLSVLSI, 2006, Philadelphia, PA, USA. **Proceedings...** New York: ACM Press, 2006. p.73–78.

YANG, X.; CHOI, B.-K.; SARRAFZADEH, M. Timing-driven placement using design hierarchy guided constraint generation. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED DESIGN, ICCAD, 2002, San Jose, CA. **Proceedings...** New York: ACM Press, 2002. p.177–180.

YAO, B. et al. Revisiting floorplan representations. In: INTERNATIONAL SYMPOSIUM ON PHYSICAL DESIGN, ISPD, 2001, Sonoma, CA, USA. **Proceedings...** New York: ACM Press, 2001. p.138–143.

YILDIZ, M. C.; MADDEN, P. H. Improved cut sequences for partitioning based placement. In: DESIGN AUTOMATION CONFERENCE, DAC, 2001, Las Vegas, NV, USA. **Proceedings...** New York: ACM Press, 2001. p.776–779.

ZACHARIASEN, M. Rectilinear Full Steiner Tree Generation. **Networks**, [S.l.], v.33, p.125–133, 1999.

APPENDIX A ALGORITMOS PARA A REDUÇÃO DO COMPRIMENTO DOS FIOS DE CIRCUITOS VLSI CONSIDERANDO CAMINHOS CRÍTICOS

A.1 Introdução

Um dos problemas mais importantes impostos pelas tecnologias recentes está relacionado com os fios do circuito. Primeiro, considere o aumento do tamanho dos projetos enquanto que o tamanho dos componentes do circuito está se tornando drásticamente menor. Este cenário impõe redes de fios cada vez mais largas, densas e complexas. Segundo, considere que o atraso dos componentes ativos reduziu mais rapidamente que o atraso dos fios. Hoje em dia, a resistência dos fios é extremamente relevante, enquanto que isto era ignorado no passado. Por estas razões, o atraso das conexões é responsável por mais da metade do atraso do circuito como um todo. Terceiro, considere o processo de fabricação e efeitos elétricos parasitas das tecnologias mais modernas, que basicamente introduzem inúmeras regras de projeto que devem ser respeitadas para manter o circuito em funcionamento. Regularidade de conexões é cada vez mais difícil de obter devido a topologia aleatória das redes em um circuito. Finalmente, considere o consumo de potência, que é fortemente afetado pela capacitância dos nodos de um circuito. Os fios largos criam grandes capacitâncias que são carregadas e descarregadas a cada transição do sinal. Em conclusão, a qualidade de um circuito está fortemente ligada a complexidade de conexões do mesmo.

Para reduzir os problemas relacionados à fiação do circuito, há um esforço de pesquisa muito significativo para reduzir o comprimento dos fios. Fios mais curtos são mais rápidos, dissipam menos potência e melhoram a roteabilidade e manufaturalidade. Entre as técnicas propostas, o uso de algoritmos específicos é uma das maneiras mais efetivas.

Os vários estágios de um fluxo de CAD podem ser agrupados em quatro passos consecutivos: síntese de sistema, síntese de alto nível, síntese lógica e finalmente síntese física. A síntese física, que é reponsável pela transição de uma descrição do circuito em nível lógico para o leiaute é uma tarefa bastante complexa e sensível a complexidade da fiação. Ela é composta de sub-tarefas como posicionamento das células e roteamento das conexões. O leitor pode se referir a (SHERWANI, 1998) para obter mais informações básicas sobre estas etapas.

A etapa de síntese física é diretamente responsável por lidar com problemas relacionados a conexões. Por esta razão, os algoritmos que a incorporam devem necessariamente objetivar redução do tamanho das conexões. Apesar da existência de ótimas técnicas para redução de fios neste nível, a constante evolução tecnológica exige que estes algoritmos sejam atualizados regularmente. Particularmente, atraso e potência são problemas impor-

tantes nos projetos atuais que impactam significativamente as estapas de posicionamento e roteamento. Análise de atraso e potência podem identificar componentes mais críticos do circuito que deveriam receber atenção especial por algoritmos de síntese física e conseqüentemente tamanho de conexão mais curtos que a média. As metodologias conhecidas como *timing-driven* e *power-driven* são exemplos encontrados encontradas na literatura para posicionamento e roteamento dirigidos à atraso e potência.

A redução do tamanho dos componentes, que costumava ser uma importante ferramenta para melhorar o atraso e a dissipação de potência de circuitos, é hoje uma fonte de novos problemas relacionados aos fios dos circuitos. Note que hoje em dia os componentes têm dimensões próximas a atômicas, o que sugere que novas alternativas devem ser buscadas para melhorar o desempenho dos circuitos.

Recentemente, a tecnologia de circuitos 3D foi proposta. Esta tecnologia aparece como uma possível solução para a estrutura de fios de um circuito. É esperado que arranjando os elementos de um circuito em 3D possa levar à fios mais curtos. De fato, trabalhos de pesquisa recentes como (DAS, S. et al, 2004), (ABABEI; MOGAL; BAZARGAN, 2005), (DAVIS, W. et al, 2005), (BANERJEE, K. et al, 2001) e muitos outros demonstram que circuitos 3D podem de fato levar a redução do tamanho dos fios. Também é demonstrado que a melhora na fiação é proporcional ao tamanho do circuito (OBENAUS; SZYMANSKI, 1999). Os trabalhos de (3D ICS INDUSTRY SUMMARY, 2005) e (TEZZARON HOMEPAGE, 2005) especificam o existente interesse da indústria e da academia nesta tecnologia.

O fato de que projetos de microeletrônica possam ser realizados em 3D abre um imenso espaço de pesquisa na área de ferramentas de CAD, principalmente no nível de síntese física. Novos algoritmos devem ser providos para lidar com o arranjo 3D dos elementos e tirar o máximo de benefício desta tecnologia enquanto consideram limitações e restrições da mesma. Hoje em dia, a pesquisa neste campo ainda está no seu início. Considere o problema de posicionamento 3D por exemplo. A riquesa e variedade de técnicas que levaram a uma melhora expressiva dos algoritmos de posicionamento através das décadas devem propiciar uma maturidade similar para a área de posicionamento 3D.

Esta tese objetiva o desenvolvimento de algoritmos para redução do tamanho dos fios considerando elementos críticos. Ela considera o problema em duas perspectivas diferentes: posicionamento e roteamento. Em posicionamento a tese explora métodos para realizar posicionamento de células em circuitos 3D considerando problemas relacionados às conexões verticais (conhecidas como 3D-Vias) enquanto procura obter um tamanho reduzido de conexões aproveitando o arranjo 3D dos elementos. Um volume significativo de novos métodos e contribuições são apresentados neste texto, sendo que foram totalmente validados numa ferramenta de posicionamento chamada Z-Place. Para roteamento, um algoritmo chamado AMAZE combina trabalhos já existentes e publicados com novos métodos que são efetivos para produzir fios curtos e com baixo atraso para elementos críticos. Enquanto o algoritmo AMAZE apresenta melhoras significativas em um algoritmo largamente utilizado pela indústria de semi-condutores (Maze Routers), ele produz árvores de roteamento com flexibilidade suficiente para serem usadas em outras etapas da síntese, como estimativa de tamanho dos fios (Steiner tree), inserção de buffers e roteamento global. Devido ao fato de se aplicar em qualquer domínio modelado por um grafo, AMAZE pode ser usado em roteamento 3D necessitando apenas que haja uma modelagem adequada da 3D-Via.

O texto está organizado como segue. O capítulo A.2 resume as tecnologias de circuitos 3D explicitando como aproveitar a tecnologia para gerar circuitos com fios mais curtos,

considerando técnicas já existentes na literatura. O capítulo A.3 apresenta a ferramenta Z-Place bem como os algoritmos que a compoem, o que representa as contribuições deste tes na área de posicionamento 3D. O capítulo A.4 apresenta as contribuições desta tese na area de roteamento. Finalmente, as conclusões apresentam um resumo das contribuições desta tese para a comunidade de CAD.

A.2 Circuitos 3D Como Um Novo Paradigma de Projeto

Um circuito 3D pode ser definido como um chip VLSI com camadas ativas empilhadas chamadas de *tiers*. A figura A.1 demonstra uma visão didática de um circuito 3D composto por camadas ativas e níveis de metal. A comunicação entre duas camadas adjacentes é dada por uma via especial chamada de 3D-Via.

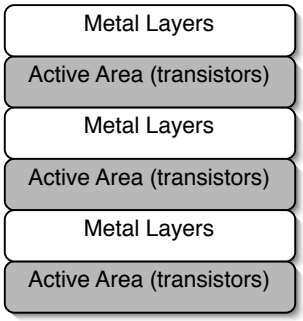


Figure A.1: Uma visão didática de um circuito 3D composto por camadas ativas e camadas de metal

A.2.1 Dados de tecnologia de circuitos 3D

Esta sessão resume algums dados tecnológicos obtidos após uma pesquisa em artigos publicados na literatura, especialmente em (DAVIS, W. et al, 2005), (PATTI, 2006a), (SUNTHARALINGAM, V. et al, 2005), (DAS, S. et al, 2004) e (GUPTA, S. et al, 2005). As 3D-Vias são caracterizadas de acordo com as seguintes características:

- A estratégia usada para integrar os *tiers* conectados por uma dada 3D-Via que pode ser *face-to-face*, *face-to-back* ou *back-to-back*.
- A distância entre dois *tiers* adjacentes (chamada também de espaçamento entre *tiers*).
- O espaçamento mínimo requirido para 3D-Vias.
- O fato de alguns tipos de 3D-Vias ocuparem área ativa e outros tipos não.

A figura A.2 exemplifica a tecnologia da empresa Tezzaron para circuitos 3D, que apresenta 3D-Vias *face-to-face* e *face-to-back*.

Uma lista de 3D-Vias e suas características é dada na tabela A.1.

Em conclusão, podemos observar que existe uma variedade de requisitos de espaçamento, enquanto que algumas 3D-Vias ocupam area ativa.

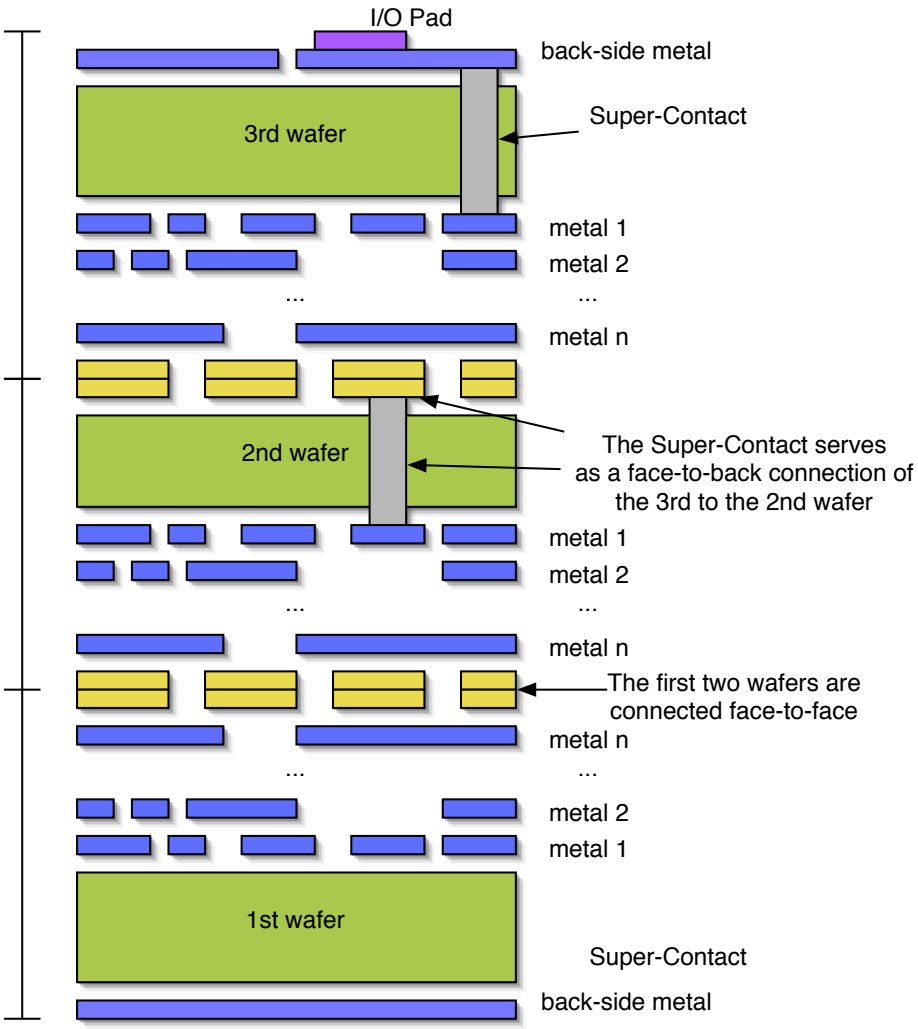


Figure A.2: Tecnologia da Tezzaron com Super-Contacts e conexão *face-to-face*.

A.2.2 Potenciais vantagens de circuitos 3D

Seja por análise analítica (BANERJEE, K. et al, 2001) (DAS; CHANDRAKASAN; REIF, 2004a) (RAHMAN; REIF, 2000) (RAHMAN; REIF, 2001) ou experimentos práticos (DAVIS, W. et al, 2005) (DAS, S. et al, 2004) (KAYA, I. et al, 2004) (ABABEI, C. et al, 2005) (ABABEI; MOGAL; BAZARGAN, 2005), é sabido que circuitos 3D podem apresentar vários benefícios, entre eles redução do tamanho dos fios mais longos, redução do tamanho médio do fio, redução da potência dinâmica (principalmente devido a redução da rede de relógio - clock), redução do atraso e tamanho do chip, entre outras. Todas estas vantagens potenciais devem ser exploradas por ferramentas de CAD apropriadas, que devem considerar limitações existentes nos circuitos 3D. Entre as principais limitações destacam-se as seguintes. Inicialmente, questões térmicas devido à difícil dissipação de calor dos *tiers* internos referenciaas em trabalhos como (GOPLEN; SAPATNEKAR, 2005) (HUA, H. et al, 2006) (RAHMAN; REIF, 2001) (ABABEI, C. et al, 2005) (DAVIS, W. et al, 2005) (DAS; CHANDRAKASAN; REIF, 2004b) que apresentam soluções e pesquisas neste tema. Secundariamente, problemas relacionados a *yield*. Alguns autores, como (DAVIS, W. et al, 2005) classificam o *yield* como um dos principais

Table A.1: Resumo dos dados tecnologicos coletados para 3D-Vias.

3D-Via	Estratégia Integração	Espaçamento Tier	Espaçamento 3D-Via	Ocupa Area Ativa
Tezzaron (Copper Pads)	face-to-face	16-20 μm	2.4 μm	no
Tezzaron (Projected)	face-to-face	16-20 μm	1.46 μm	no
Microbump	face-to-face	16-20 μm	10-100 μm	no
Contactless (Capacitive)	face-to-face	16-20 μm	50-200 μm	no
MIT (Copper/Tantalum Pads)	face-to-face	16-20 μm	5 μm	no
TSV face-to-face	face-to-face	16-20 μm	0.5 μm	no
Tezzaron Super-Via^{TM}	face-to-back	15-20 μm	6.08 μm	yes
Tezzaron Super-Contact^{TM}	face-to-back	11-15 μm	< 4 μm	yes
Microbump 3D Package	face-to-back	11-15 μm	25-50 μm	no
Contactless Inductive	face-to-back	11-15 μm	50-150 μm	yes
MITLL Through Via (SOI)	face-to-back	9-12 μm	5 μm	yes
Through Via (regular Bulk)	face-to-back	11-15 μm	50 μm	yes
Back-to-back 3D-Via	back-to-back	6-8 μm	15 μm	yes

problemas relacionados a circuitos 3D, enquanto que outros como (PATTI, 2006a) argumentam que este problema não é tão grave e pode ser contornado de diversas maneiras. Finalmente, mas não menos importante, os problemas relacionados às 3D-Vias. Muitos autores, como (DAS; CHANDRAKASAN; REIF, 2004b) e (ABABEI, C. et al, 2005) ignoram boa parte destes problemas. Curiosamente, o trabalho em (LIU, G. et al, 2005) cita inclusive que é muito complexo fazer tratamento de Vias-3D no nível de posicionamento. Por outro lado, desconsiderar que 3D-Vias podem ocupar área ativa e que também são recursos escassos e caros de roteamento, pode significar que o circuito posicionado não possa ser implementado. Então trata-se de um problema de importância significativa. Desta forma, os seguintes problemas são destacados:

- Limitação na quantidade de 3D-Vias.
- Posicionamento e legalização de 3D-Vias.
- Posicionamento de células e 3D-Vias simultaneamente.
- Modelagem das características elétricas e topológicas das 3D-Vias.
- Melhoria de comprimento das conexões com introdução de 3D-Vias.
- Particionamento das células em *tiers* considerando 3D-Vias.
- Evitando 3D-Vias em determinadas conexões

A.2.2.1 Metodologias de projeto

Esta tese estuda basicamente três metodologias de projeto para circuitos 3D com relação à granularidade da integração. Inicialmente considere a metodologia que apresenta maior granularidade, chamada de *tier level integration*. Ela integra *tiers* de natureza distinta, projetados separadamente. Após, visando obter melhora no tamanho dos fios, o

circuito pode ser integrado mais fortemente, com o uso de blocos espalhados nos diversos tiers (*ip core level integration*). Finalmente, pode-se considerar uma granularidade pequena, chamada *random logic level*. Esta última apresenta maior potencialidade para melhora de tamanho dos fios, porém é mais sucetível ao aumento de 3D-Vias, o que pode gerar os problemas ja citados na sessão anterior. Em função de ter a maior potencialidade de melhora, esta tese se baseia em integração no nível de lógica aleatória.

A.3 Z-Place: Algoritmos para posicionamento 3D

A.3.1 Introdução

Z-Place é uma ferramenta para posicionamento de circuitos 3D. Como primeiro objetivo, Z-Place procura obter o melhor comprimento de conexões sob certas restrições relacionadas às 3D-Vias. Nas próximas sessões são apresentados algoritmos e o fluxo da ferramenta.

A.3.1.1 Fluxo da ferramenta Z-Place

Z-Place executa as seguintes tarefas: tratamento dos pinos de entrada e saída (sessão A.3.1.2), posicionamento global (section A.3.1.3), posicionamento detalhado e posicionamento das 3D-Vias (sessão A.3.1.5). O posicionamento detalhado é basicamente um processo de otimização baseado no algoritmo Threshold Accept (DUECK; SCHEUER, 1990) com aplicação de diversas técnicas para melhora de tempo de CPU, como janelamento por exemplo. O planejamento da quantidade de 3D-Vias entre dois *tiers* adjacentes é executada no nível de posicionamento global. O tratamento de conexões críticas também é feito no nível de posicionamento global, evitando o uso de 3D-Vias com elementos críticos do circuito. Este método é descrito com mais detalhes na sessão A.3.1.4. A figura A.3 apresenta uma ilustração do fluxo de posicionamento.

A.3.1.2 Tratamento dos pinos de E/S

O problema de particionamento e posicionamento de pinos de entrada e saída é ilustrado na figura A.4. Para que o posicionamento global possa ter efeito é necessário que haja pinos fixos ao redor da área de posicionamento. Com isto, o tratamento de pinos de E/S deve prover pinos em todos os *tiers* de forma balanceada, como mostra a figura A.4.

Um algoritmo para particionamento e posicionamento dos pinos de E/S é apresentado em (HENTSCHKE; JOHANN; REIS, 2006). O método parte de um posicionamento 2D dos pinos ao redor da área de posicionamento e retorna uma representação 3D do circuito composta de um conjunto de *tiers* e pinos posicionados ao redor. Como objetivo secundário, o método procura servir como um bom ponto de partida para algoritmos de posicionamento gerarem soluções com um reduzido número de 3D-Vias.

O algoritmo inicia fazendo um particionamento dos pinos. Este processo é realizado em duas etapas. Primeiro são encontradas as distâncias lógicas entre cada par de pinos ao longo das redes do circuito (figura A.5.(a)). Depois, o tamanho dos pinos é utilizado de peso para a montagem de um grafo completo que contém somente os pinos de entrada e saída (figura A.5.(b)).

Após obter um grafo completo de pinos, um algoritmo de particionamento divide o grafo em 2 ou mais grupos, que são assinalados para um *tier*. Finalmente, os pinos de entrada e saída são posicionados usando um mapeamento linear de sua posição original

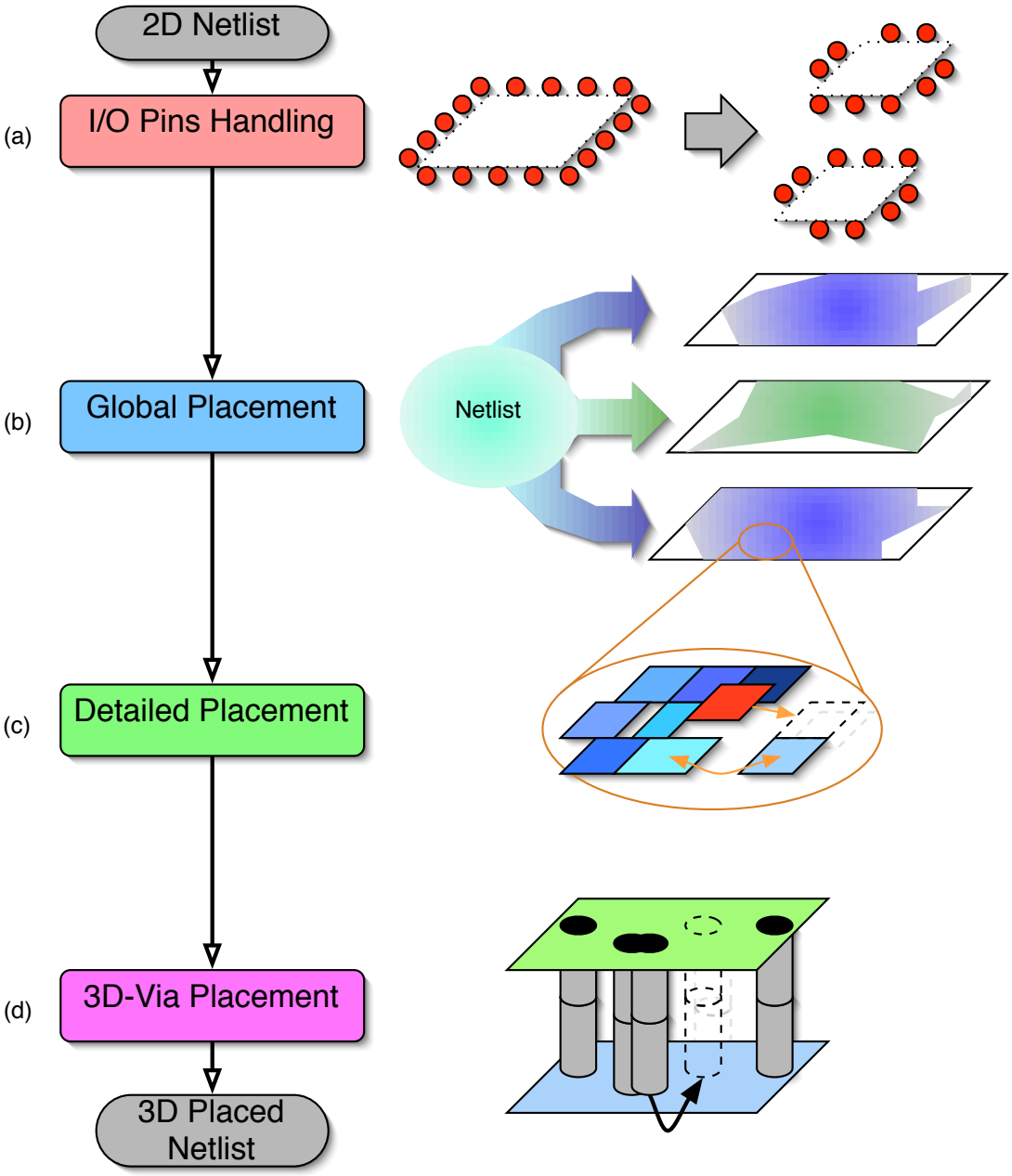


Figure A.3: Fluxo de posicionamento proposto para circuitos 3D

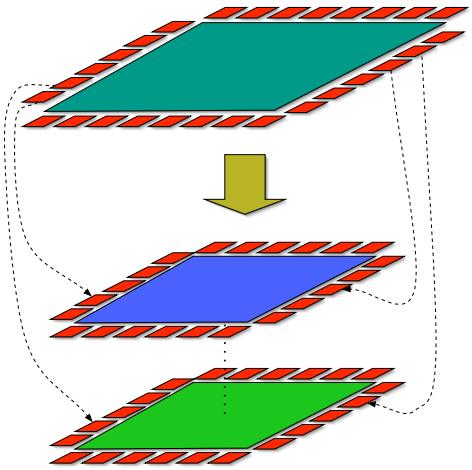


Figure A.4: Migração (de 2D para 3D) de um circuito com pinos de E/S pré-posicionados

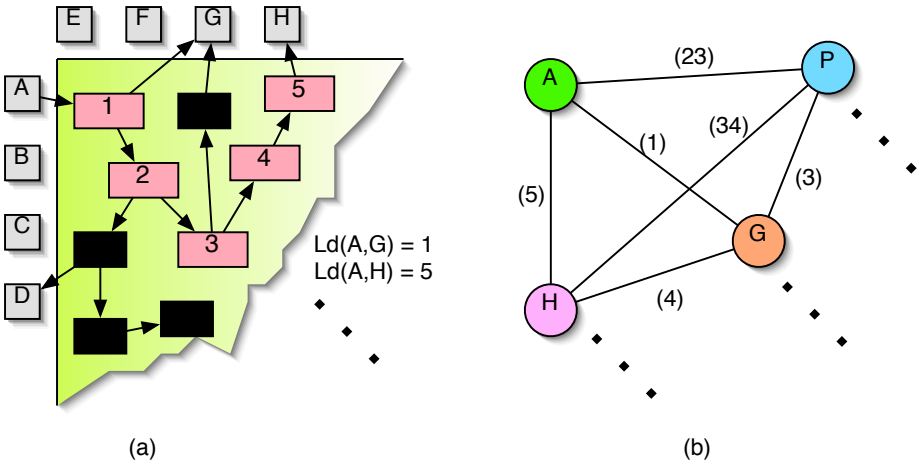


Figure A.5: Distância lógica entre pinos de entrada e saída (a) e uma parte do grafo completo de pinos correspondente (b)

para uma área reduzida, mantendo assim a ordem original dos pinos.

Foram realizados experimentos que demonstraram a efetividade do método para obter bom equilíbrio de pinos ao longo do circuito e, ao mesmo tempo, um reduzido número de 3D-Vias após realizar o particionamento das células. Observou-se que a informação da distância lógica resume a informação global do circuito de forma bastante comprimida e completa; por esta razão, o algoritmo de particionamento do grafo de pinos trata eficientemente o particionamento do *netlist* principalmente por se tratar de um grafo pequeno em relação ao hiper-grafo do circuito.

A.3.1.3 Global Placement

O espaço de roteamento global é modelado em um cubo fatiado, conforme mostra a figura A.6. Cada fatia representa um *tier* para posicionar células, enquanto que a área entre eles, destinada para níveis de metal, pode ser utilizada temporariamente por células,

que devem migrar para uma posição válida em algum tier.

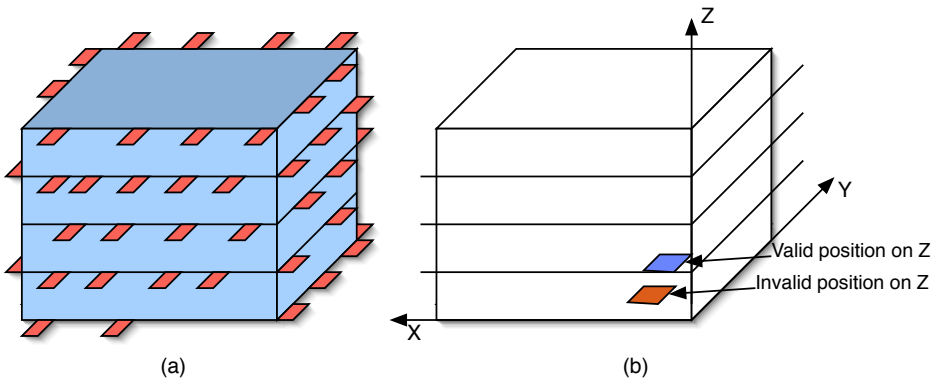


Figure A.6: Modelo de cubo fatiado com coordinadas inválidas e pinos de E/S em todos os *tiers*.

A estratégia de integração entre *tiers* adjacentes influencia o processo de posicionamento global das células. Uma das tarefas do posicionamento global é de distribuir as células ao longo do espaço de posicionamento, o que inclui o particionamento de células nos diversos *tiers*. Porém, alguns tipos de via (*face-to-back* e *back-to-back*) ocupam área ativa e por isto a distribuição das células em *tiers* deve levar em conta este fato. A figura A.7 ilustra o efeito da estratégia de integração no particionamento das células.

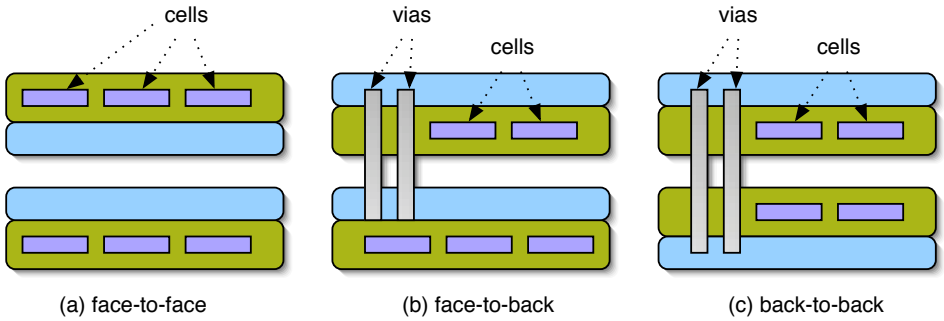


Figure A.7: Estratégias de integração 3D e como elas impactam a distribuição da área: (a) *face-to-face* and (b) *face-to-back*.

Outro fator de influência na distribuição vertical das células é o tamanho das 3D-Vias. Note que, quanto maior a 3D-Via, maior será a ocupação de área ativa por 3D-Vias. Porém, além disto, o algorítmo de posicionamento global do Z-Place controla as 3D-Vias de acordo com uma quantidade máxima de 3D-Vias pré-estipulada. Esta quantidade é calculada por uma relação entre a área disponível para Vias-3D e o tamanho de cada uma das 3D-Vias.

O algoritmo de posicionamento global proposto é baseado em otimização quadrática de comprimento dos fios. Esta algoritmo é largamente utilizado na literatura. Particularmente, o núcleo do algoritmo do Z-Place possui características similares ao trabalho (VISWANATHAN; PAN; CHU, 2005) utilizado em circuitos 2D. Em Z-Place, o algoritmo $Cell\ Shifting$ é extendido para $3D\ Cell\ Shifting$, que possui uma função especial para calcular a coordenada z das células. Esta função chama-se $Z\text{-}Cell\ Shifting$

(HENTSCHKE, R. et al, 2006) e é ilustrada na figura A.8.(a). A metodologia $Z\text{-}Cell\ Shifting$ ordena as células pela sua coordenada z e obtém pontos de corte que consideram tanto área de células quanto área ativa ocupada por 3D-Vias.

O algoritmo $3D\ Cell\ Shifting$, ilustrado na figura A.8.(b), espalha as células nas três dimensões ao mesmo tempo. Por esta razão, a otimização do comprimento dos fios é uniforme. Quando as células estiverem razoavelmente espalhadas no espaço, o algoritmo de refinamento iterativo é aplicado. Além de otimizar o comprimento dos fios e o espalhamento das células, como feito em (VISWANATHAN; PAN; CHU, 2005), o algoritmo na ferramenta Z-Place controla a quantidade de 3D-Vias inserida para que repeite o limite pré-estabelecido. Enquanto o limite é obedecido, o algoritmo permite a inserção de 3D-Vias. Porém, quando o limite é desobedecido, o algoritmo é encorajado a retirar 3D-Vias sempre que possível.

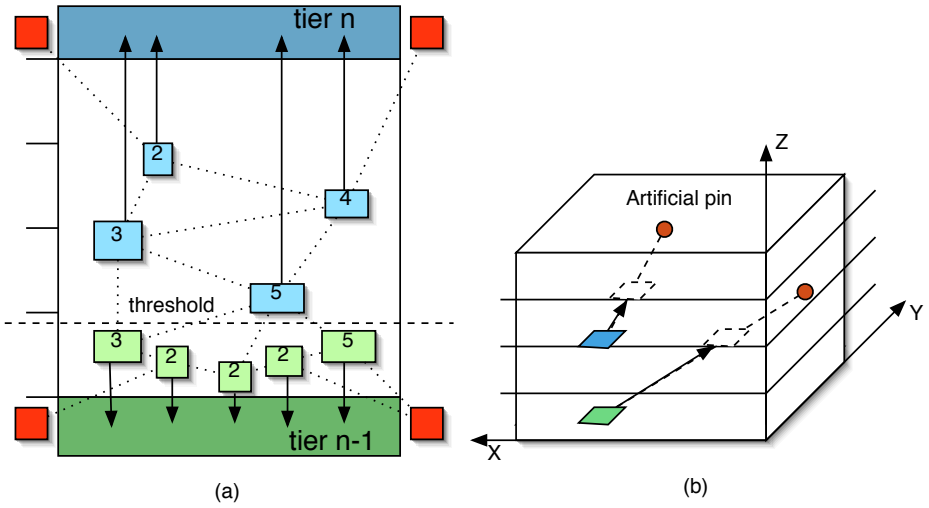


Figure A.8: A metodologia Z-Cell Shifting (a) e o algoritmo 3D Cell Shifting (b).

Os resultados experimentais demonstraram que a manipulação do limite de 3D-Vias influencia o comprimento dos fios. Note que este limite é calculado em função da área, o que torna Z-Place adaptável a diversas tecnologias, melhorando o comprimento dos fios até o limite imposto pelas mesmas. Foi estudado o efeito de adicionar mais *tiers* ao circuito 3D contra soluções providas pela ferramenta FastPlace (VISWANATHAN; PAN; CHU, 2005) e foi verificado que nossas melhores configurações produzem melhoras de 15% para 2 *tiers*, 20% para 3 *tiers* e finalmente 27% para 4 *tiers* em média.

A.3.1.4 Tratamento de caminhos críticos

3D-Vias podem ser vistas como elementos prejudiciais ao atraso e potência de uma rede por dois aspectos. Primeiro, note que uma conexão *inter-tier* demanda que haja um roteamento passando por todos os níveis de metal para que possa atingir a 3D-Via. Segundo, as características elétricas (capacitância e resistência) destas 3D-Vias variam bastante e podem ser bastante prejudiciais dependendo da tecnologia empregada. Por esta razão, deseja-se encontrar um mecanismo que evite o uso de 3D-Via em determinadas conexões escolhidas como críticas.

O mecanismo adotado é explicado na figura A.9. Para cada conjunto de células que deve se manter no mesmo *tier* é inserido um pino artificial. No caso do cálculo de atraso,

faz sentido manter próximas as células pertencentes ao mesmo caminho crítico, pois não haverá nenhuma 3D-Via ao longo do caminho. Cada pino artificial é conectado a todas as células pertencentes ao seu grupo. Esta conexão é chamada de $critical\ star$. O peso empregado nestas conexões é dado de forma que ela tenha um efeito pequeno na distribuição horizontal das células, mas tenha um forte efeito para manter elas na mesma coordenada z. Por isto, diferentemente de qualquer outra conexão do circuito, ela tem um peso forte no eixo Z e um peso bastante baixo nos outros eixos.

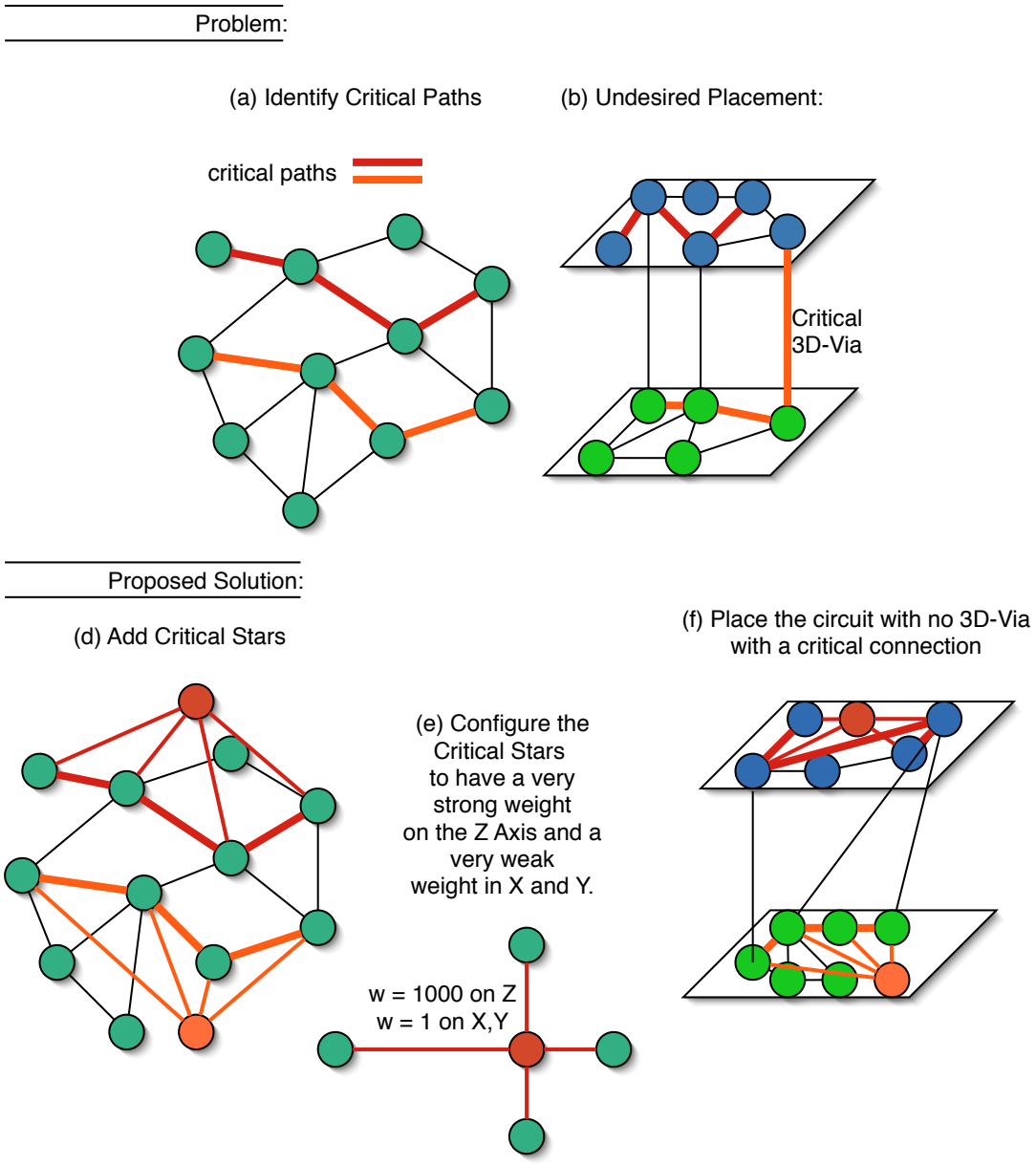


Figure A.9: Uma ilustração do método proposto para evitar 3D-Vias em conexões críticas.

Foram realizados experimentos com um conjunto de *benchmarks* sintetizado pela ferramenta Leonardo, destacando um grupo de 100 caminhos críticos. O conjunto forma um total de 12 circuitos, mapeados para 2, 3 e 4 tiers, formando 36 combinações. Foi medido a quantidade de 3D-Vias nos caminhos criticos, o comprimento total das conexões e o

número total de 3D-Vias. Após aplicar a técnica descrita acima, todas as combinações (com a exceção de 2) cairam de várias centenas (entre 88 e 13351) para 0. O efeito no comprimento total das conexões foi mínimo; em média apenas 0.03%. Este experimento levou a conclusão de que o algoritmo aproveitou um grau de liberdade, existente no posicionamento, de que várias soluções com mesmo tamanho de conexões possuem diferentes configurações de Vias-3D. O efeito na quantidade total de 3D-Vias também foi muito baixo: apenas 2% em média.

A.3.1.5 Posicionamento de 3D-Vias

Entre cada *tier* do circuito existe uma camada de 3D-Vias. A área ocupado por cada 3D-Via depende do tipo e da tecnologia para este nível específico de 3D-Vias. A etapa de posicionamento global já garante que as 3D-Vias podem ser posicionadas no seu nível sem que hajam sobreposições. Resta então encontrar estas posições de forma que o tamanho das conexões seja minimamente afetado. Observe que uma 3D-Via pertence a uma rede de células já posicionadas; qualquer posição dada para esta 3D-Via dentro da área delimitada por células da rede não acrescenta nenhuma penalidade ao comprimento desta conexão. Assim, o problema de posicionamento de 3D-Vias é definido para tentar manter todas as 3D-Vias dentro de regiões retangulares definidas pela rede que as pertence; caso não seja possível, que a distância entre as 3D-Vias e a região destinada seja minimizada.

O algoritmo proposto, detalhado em (HENTSCHKE; REIS, 2007), baseia-se em um algoritmo de legalização de células conhecido como Tetris (KHATKHATE, A. et al, 2004). O algoritmo é descrito graficamente na figura A.10. O primeiro passo é assinalar uma banda para cada 3D-Via; depois, as células são movidas uma a uma (ordenada pela sua posição x inicial) para o fim de uma banda; a banda é escolhida de forma que o custo da solução seja minimizado. No caso deste algoritmo, o custo é calculado baseado na posição de destino estar ou não estar dentro da região destinada para a 3D-Via.

Os resultados experimentais demonstram que o método consegue acomodar todas as 3D-Vias com uma degradação muito pequena no tamanho das conexões (sempre menos de 5% mas na maior parte dos casos difíceis a degradação foi menos de 0.1%). Comparado com um método publicado (YAN, H. et al, 2005), o método proposto obtem resultados ligeiramente melhores com uma vantagem de ordens de magnitude em tempo de CPU.

A.4 Roteamento de Steiner com o algoritmo AMAZE

O problema de roteamento pode ser definido informalmente como a tarefa de conectar pinos fixos através de fios de metal. Como já foi revisado anteriormente, estes fios hoje em dia têm impacto decisivo na performance e custo de um circuito. Assumindo que as etapas anteriores tenham feito um bom trabalho para manter os elementos do circuito próximos o suficiente, o roteamento é responsável pela conexão propriamente dita.

Para que etapas anteriores possam efetivamente produzir soluções boas para o roteamento, elas precisam estimar o tamanho das conexões, mesmo não possuindo ainda as informações necessárias para realizar o roteamento propriamente dito. Para fazer isto, os pinos recebem posicionamentos temporários e é realizada uma árvore no formato de Steiner, que se caracteriza por possuir junções de fios em qualquer parte da árvore. O ponto exato onde ocorre a junção é chamado de Steiner point. Existem diversos algoritmos que geram arvores de Steiner em tempo rápido de forma que possa ser utilizado durante o processo de síntese do circuito.

Esta tese propõe um novo algoritmo de roteamento com característica de algoritmos

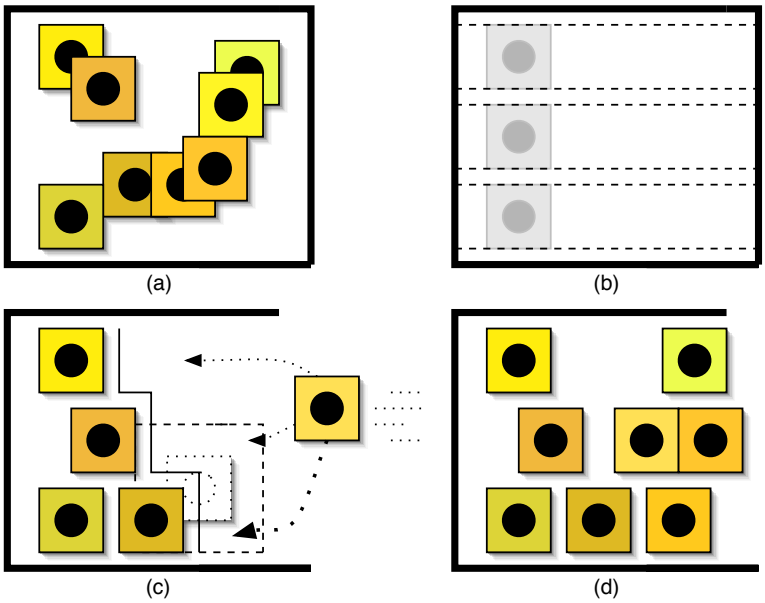


Figure A.10: Um algoritmo estilo Tetris para legalizar as 3D-Vias; (a) representa uma solução inicial; (b) demonstra o passo 2 do algoritmo que fatia a área em bandas horizontais; (c) ilustra os passos 4-10 do algoritmo que assinalam uma banda para as células; (d) demonstra a solução final.

de geração de árvores de Steiner chamado AMAZE. Entre as motivações deste trabalho está o fato de promover melhoras em um algoritmo de roteamento padrão da indústria chamado *Maze Router*. Uma segunda motivação seria promover um algoritmo de roteamento cujo tempo de CPU seja compatível aos algoritmos de árvores de Steiner, de forma que o mesmo algoritmo possa ser usado tanto na estimativa da conexão quanto no roteamento propriamente dito. Os *Maze Routers* são algoritmos extremamente flexíveis, o que favorece a adaptação dos mesmos para diferentes aplicações, como por exemplo roteamento de circuitos 3D.

O algoritmo AMAZE promove métodos para endereçar o comprimento total das árvores geradas, o atraso para elementos críticos e finalmente o tempo de CPU consumido pelo mesmo.

A.4.1 Melhorando o tamanho das árvores com a técnica de biasing

A técnica de biasing pode ser entendida como uma técnica de desempate de conexões de mesmo tamanho com um critério que tenta favorecer conexões futuras. Considere o exemplo da figura A.11; a conexão do nodo à esquerda para o nodo b pode tomar duas rotas, conforme visto nas opções da figura A.11.(a) e A.11.(b). A opção na figura A.11.(a) favorece o comprimento total da árvore, já que o nodo c pode ser unido a árvore com um fio menor.

A situação da figura de exemplo é bastante simples de capturar, mas situações mais complexas, onde há diversos nodos para conexão e não se sabe quais poderão tirar proveito da conexão são bastante comuns. Por esta razão desenvolveu-se um critério para definir quais nodos vizinhos podem participar na decisão da rota e quais nodos não devem participar por estarem numa posição desfavorável. Este critério é explicado na

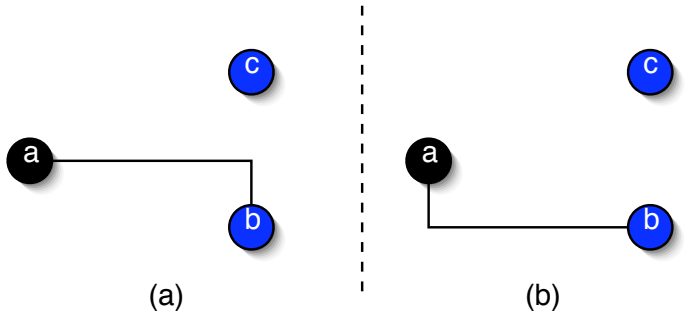


Figure A.11: Uma situação de roteamento favorável para a redução da fiação através do compartilhamento (a) e uma situação favorável para o isolamento dos caminhos

figura A.12. Após computados os nodos afetados, o caminho a ser escolhido é tomado baseado na posição média de todos os nodos afetados.

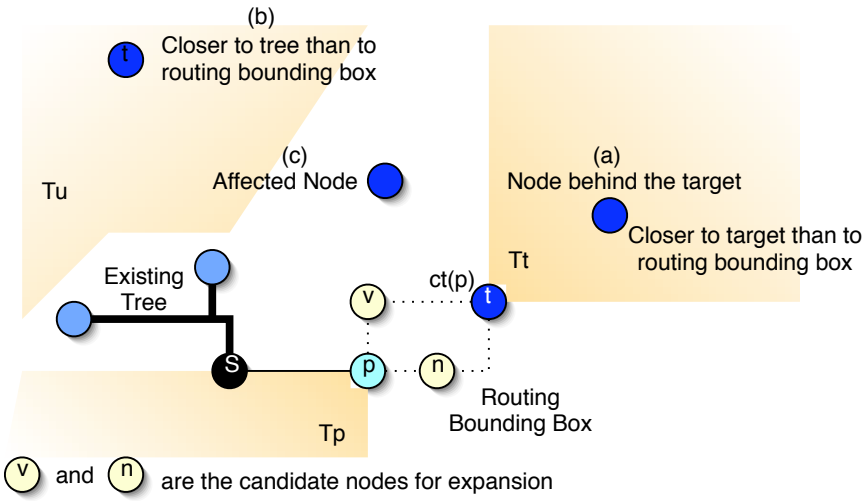


Figure A.12: Uma ilustração da técnica de biasing e os nodos afetados. (a) mostra um nodo que não é afetado porque está atrás do nodo de destino. (b) mostra um nodo que está mais próximo da árvore existente do que a caixa até o destino. (c) mostra um exemplo de um nodo afetado. Nesta situação descrita, o caminho dará preferência a tomar a rota passando pelo nodo v em função do nodo afetado.

Resultados experimentais mostram que a técnica de biasing propicia uma melhora média de 2% no comprimento das conexões, sendo que uma grande maioria delas não é influenciada pela técnica. Porém, nas redes em que a decisão de qual rota tomar influencia o comprimento da conexão, este impacto é bem maior, na ordem de 15%, como visto na figura A.13.

Por fim, o algoritmo de roteamento com a técnica de biasing foi comparado com um algoritmo que obtem a topologia com o menor comprimento de fio possível; verificou-se que o algoritmo AMAZE produz árvores com degradação de apenas 3% em relação à

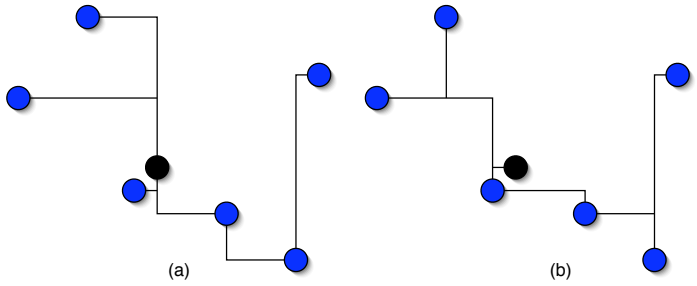


Figure A.13: Árvores sem *biasing* (a) e com *biasing* (b).

solução ótima.

A.4.2 Melhorando o atraso para conexões críticas

A topologia de uma árvore têm uma influência importante no atraso da mesma. Vários tipos de topologias são demonstrados na figura A.14. Na figura A.14.(a) apresenta-se o problema de conectar vários pinos pertencentes à mesma rede a um elemento chamado driver, que alimenta a rede com um sinal que deve se propagar até todos os outros nós. A figura A.14.(b) mostra uma topologia de comprimento mínimo, na qual elementos distantes do driver podem ter um atraso muito grande. A figura A.14.(c) apresenta uma topologia chamada de arborecência, que conecta todos os nós com distância mínima até o driver. Note que o comprimento da conexão nesta topologia é bastante grande. A figura A.14.(d) demonstra uma topologia intermediária entre (b) e (c). A topologia em (e) é chamda de estrela, e possui uma característica interessante de não compartilhar fios, característica que melhora o atraso, além de que todos os nodos são conectados a árvore com um fio de comprimento mínimo. Finalmente a topologia em (f) indentifica o elemento crítico da árvore e o conecta em estrela até o driver; os demais componentes são conectados com a topologia de comprimento mínimo.

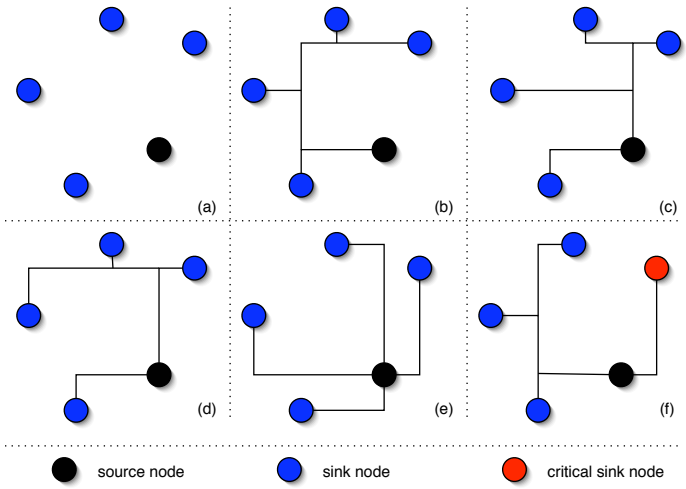


Figure A.14: Topologias para melhora do atraso; (a) rede; (b) árvore mínima; (c) arborecência mínima; (d) topologia intermediária entre (b) e (c); (e) estrela; (f) conexão direta do nodo crítico.

O algoritmo AMAZE procura obter topologias semelhantes a topologia (f). Para isto é proposto um fator de compartilhamento, que permite controle sobre o compartilhamento do fio que conecta o driver até a conexão crítica. Verificou-se experimentalmente que deve-se preferencialmente manter este fator em 0, ou seja, não perimitir nenhum compartilhamento. Porém, a opção de usar outros valores pode ajudar em algumas topologias. A figura A.15 ilustra o fator de compartilhamento.

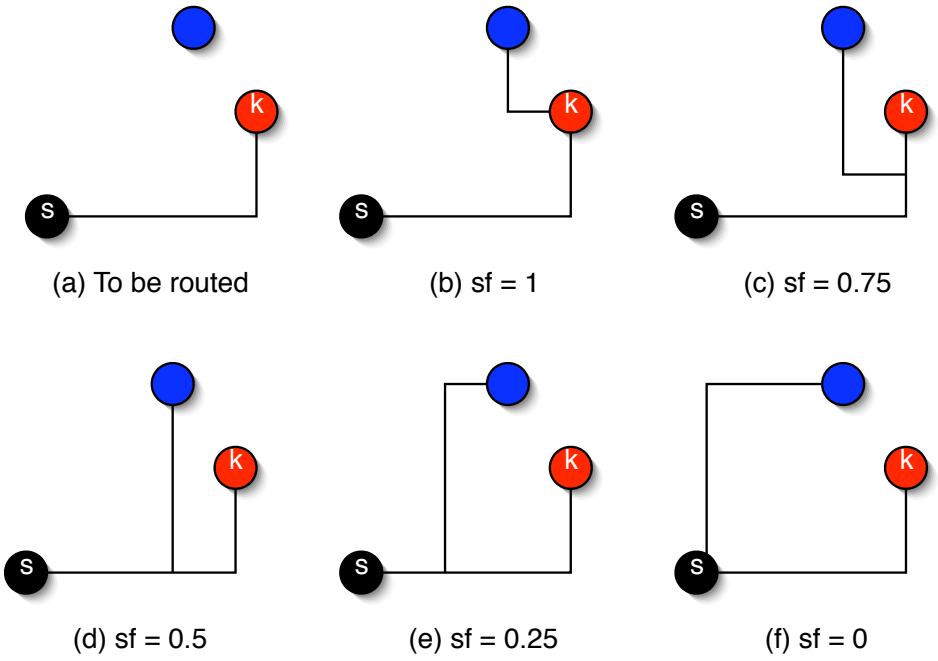


Figure A.15: Efeito do fator de compartilhamento.

Além do controle do compartilhamento, AMAZE também possui um fator para o controle sobre o tamanho da caminho do driver até o pino crítico. No caso de existir somente um pino crítico, o caminho mínimo é obtido simplesmente dando prioridade para fazer o roteamento deste pino antes dos demais. Porém, na presença de dois ou mais pinos críticos, pode haver compartilhamento e neste caso uma conexão crítica pode ter seu comprimento aumentado. A figura A.16 explica o uso do fator com um exemplo. Mais detalhes sobre ambos os fatores podem ser encontrados em (HENTSCHKE, R. et al, 2007b).

Resultados experimentais mostram que AMAZE é mais efetivo para melhorar o atraso para elementos críticos que heurísticas do estado da arte para árvores de Stiener, como as P-Trees (de 1% a 30%).

A.4.3 Melhorando o tempo de CPU

Finalmente o algoritmo AMAZE apresenta métodos para melhorar o tempo de CPU de um Maze Router. Inicialmente, faz-se uso de pesquisa heurística com o algoritmo A* (HART; NILSSON; RAPHAEL, 1968). O algoritmo AMAZE propõe uma maneira de calcular a função heurística do A* de maneira eficiente, pois evita que se compute a distância para todos os nodos de destino. Esta otimização também esta ligada com a proposta de uma estrutura de dados eficiente para o armazenamento dos nodos abertos.

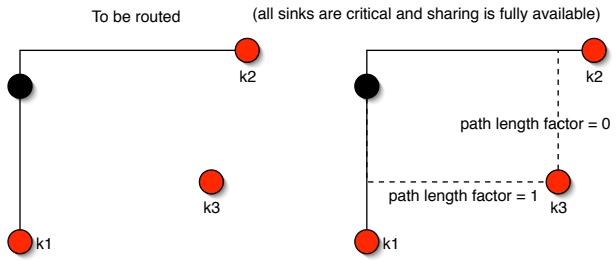


Figure A.16: Efeito do fator de tamanho do caminho.

Esta estrutura está divida em duas partes: uma lista de nodos abertos e uma lista de espera. A lista de nodos abertos em si é aprimorada em relação a estrutura clássica de fila de prioridades por possuir tempo de inserção constante. Somando todas as otimizações, o algoritmo pode atingir um desempenho bastante interessante, comparável a geração heurística de árvores de Steiner. AMAZE é capaz de gerar 415 árvores de 7 pinos em menos de 1 segundo.

A.5 Conclusões

É bastante sabido que conexões são um fator limitante para o desempenho de um circuito. Este trabalho propôs diversos algoritmos que endereçam o problema de obter conexões mais curtas em um circuito.

Na parte de posicionamento foi proposto um fluxo completo de ferramentas com diversos algoritmos. Foram apresentados algoritmos para particionamento e posicionamento de pinos de entrada e saída, posicionamento de células em 3D, posicionamento de 3D-Vias e otimização de caminho crítico. Os algoritmos propostos consideram as 3D-Vias como recursos limitados e com restrições de espaço. Os algoritmos propostos também se adaptam à tecnologia empregada. Em geral verificou-se que é possível reduzir o tamanho dos fios em 10% para cada *tier* adicionado.

Na parte de roteamento, foi proposto um algoritmo de roteamento que combina diversos métodos para empregar boas topologias de árvore em um tempo de CPU bastante rápido. Verificou-se, por exemplo, que controlando a quantidade de compartilhamento de um fio pode-se encontrar várias otimizações no atraso de uma árvore. Comparado com algoritmos da literatura, o algoritmo AMAZE produz árvores até 30% melhores. É interessante destacar o fato de AMAZE ser um conjunto de melhorias em um algoritmo padrão da indústria.