3-2006

Genetic algorithm based two-dimensional and three-dimensional floorplanning for VLSI ASICs

Pradeep R. Fernando
University of South Florida

Genetic Algorithm Based Two-Dimensional and Three-Dimensional Floorplanning for VLSI ASICs

by

Pradeep R. Fernando

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Computer Engineering
Department of Computer Science and Engineering
College of Engineering
University of South Florida

Major Professor: Srinivas Katkoori, Ph.D.
Soontae Kim, Ph.D.
Sanjukta Bhanja, Ph.D.

Date of Approval:
March 27, 2006

Keywords: Physical Design, 3D Integrated Circuits, Evolutionary Search, Placement, Sequence Pair

DEDICATION

To my loving parents

ACKNOWLEDGEMENTS

I would like to thank Dr. Srinivas Katkoori for giving me the opportunity to work with him. I am grateful to him for all his support and encouragement. Without his patient guidance, this work would not have been possible. I also thank Dr. Soontae Kim & Dr. Sanjukta Bhanja for the valuable time they took to review this thesis and for their helpful comments.

I am very grateful to my loving father, and mother for their unconditional support. I am indebted to them for being a perpetual source of inspiration and motivation for me. Finally, I cannot forget the help and support of all my friends and colleagues who have been with me in every step of this process. Special thanks to Vasant Manohar, Mahalingam, Vyas Krishnan, and Hari. I'm greatly thankful to Vasant, Mali, Supriya, Priya, Karthik, Preksha, and many others for helping me through my hard times.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

GENETIC ALGORITHM BASED TWO-DIMENSIONAL AND THREE-DIMENSIONAL FLOORPLANNING FOR VLSI ASICS

Pradeep R. Fernando

ABSTRACT

Dramatic improvements in circuit integration technologies have resulted in a huge increase in the complexity of circuits that can be fabricated on a single integrated circuit(IC). The significance of the performance and reliablity issues of interconnects has increased greatly demanding radically different solutions such as Three-Dimensional Integrated Circuits (3D IC) and Optical Interconnects. Three-Dimensional ICs are an elegant solution to the interconnect and device density issues in the current and future technology generations as they provide an additional dimension for packing the devices. This results in a direct reduction in the chip package area and the total wiring required to complete all the interconnections. More importantly, the number and the length of long, global wires are reduced significantly due to the availability of the third dimension for routing purposes. But to fully exploit all the advantages associated with three-dimensional ICs, a good three-dimensional packing of devices is needed. This greatly increases the importance of Floorplanning and Placement stages of the VLSI Physical Design process.

There have been many initial attempts to develop a physical design framework for three-dimensional ICs but only a few of them focus on physical design for three-dimensional macro-cell based circuit designs. This work develops a novel genetic algorithm for performing both two-dimensional and three-dimensional macro-cell floorplanning. The genetic floorplanner employs two novel crossover operators. The first crossover operator (MTOX) is an unbiased stochastic search operator, while the second crossover operator (HOOX) is

a heuristic operator that searches for floorplans with good area usage. Both the crossover operators can be applied transparently for both 2D and 3D floorplanning. Three mutation operators have been developed to work with the chosen floorplan representation scheme, namely Sequence Pairs. Despite the use of a comparatively small population size of 200, the genetic floorplanner achieves reduction in footprint area and wirelength for both 2D and 3D floorplanning as compared to some of the recent works in the literature. For 2D floorplanning, the genetic floorplanner achieves a 12% average reduction in total wirelength as compared to a Quadratic Programming based Floorplanner for a small 2% increase in area. For 3D floorplanning, the proposed floorplanner achieves a 11% average reduction in total wirelength and a 5% decrease in footprint area as compared to a Simulated Annealing based 3D floorplanner.

CHAPTER 1

INTRODUCTION

The world today is filled with numerous electronic devices meant for a wide variety of applications from the ultra-thin and compact mobile phones in the Communications field to the life-saving pace-maker devices in the field of Medicine. The development of these electronic devices has revolutionized every field of science and technology. Many scientists believe that one of the primary triggering factors for this technological revolution is the ability to fabricate such complex circuits in tiny chips called Integrated Circuits (ICs). The constant miniaturization of these electronic chips through the last 40 years has been made possible only due to the advances in Circuit Integration technology. The improvements in integration technologies can be largely attributed to the scaling of device and interconnect features so that a larger number of devices and interconnects can be accommodated in the same chip area.

1.1 Technology Scaling

Circuit Integration Technology has advanced from Small Scale Integration(SSI) that integrates about 10 devices in a chip to Very Large Scale Integration(VLSI) that packs millions of devices within a single chip. The devices in the current VLSI circuits have dimensions that are much lesser than 1 micron ($1 micron = 10^{-3} mm$). The current class of circuits that have feature sizes that are of the order of a hundredth of a micron are said to belong to the Deep Sub-Micron(DSM) class of circuits. The next generation of integration technology, Giga-Scale Integration, is expected to produce circuits with transistor counts in the order of billions and transistor sizes in the order of nanometers. The miniaturization of the devices and interconnects in the technology generations follow some scaling principles

that ensure proper functionality of the scaled devices and interconnects that are described below.

1.1.1 Scaling Principles

Scaling of the various device and interconnect dimensions will affect the behavior of the circuits being fabricated. To maintain the correct operation and the characteristics of a transistor, some critical parameters of both the devices and the interconnects also have to be scaled. The scaling principles used with VLSI circuits is discussed in the following sections.

1.1.2 Device Scaling

Scaling of devices can be done according to different principles such as Constant Field scaling, Constant Voltage scaling and Lateral scaling. Constant Field scaling is widely used as it accounts for the various second order effects of technology scaling. In Constant Field Scaling, the device dimensions L, W, and t_{ox} are reduced by a scaling factor, S (> 1). This reduces the chip area by a factor of S^2 and the gate delay by a linear factor of S as shown in Table 1.1.

1.1.3 Interconnect Scaling

Interconnect scaling can be done in two ways, namely Constant Thickness Scaling and Reduced Thickness Scaling. Reduced Thickness scaling has attained more popularity of late due to its applicability in the future technology generations without suffering from the adverse second order effects of scaling. Table 1.2 details the Reduced Thickness scaling principles and its effects on both the local and global interconnects.

1.1.4 Interconnect Issues in Future Technology Generations

From the above tables, it is clear that device scaling increases the device density on a chip by reducing the area of a chip quadratically and also produces a linear reduction in

Table 1.1 Scaling Effects on Device Characteristics (Reproduced from [32])

Parameter	Constant Field Scaling
Scaling Parameters	
Length (L)	1/S
Width (W)	1/S
Gate Oxide Thickness (t_{ox})	1/S
Supply Voltage (V_{DD})	1/S
Threshold Voltage (V_{tn}, V_{tp})	1/S
Substrate Doping (N_A)	S
Device Characteristics	
Resistance (R)	1
Gate Capacitance (C)	1/S
Gate Delay (τ)	1/S
Clock Frequency (f)	S
Dynamic Power Dissipation (per gate) (P)	$1/S^2$
Chip Area (A)	$1/S^2$
Current (I_{ds})	1/S
Power Density	1
Current Density	1

Table 1.2 Scaling Effects on Interconnect Characteristics (Reproduced from [32])

Parameter	Reduced Thickness Scaling
Scaling Parameters	
Width: w	1/S
Spacing: s	1/S
Thickness: t	1/S
Interlayer oxide height: h	1/S
Characteristics per Unit Length	
Wire Resistance per unit length: R_w	S^2
Total Wire Capacitance per unit length: C_w	1
Unrepeated RC constant per unit length: t_{wu}	S^2
Repeated wire RC delay per unit length: t_{wr}	$\sqrt{S}$
Crosstalk noise	1
Local/Scaled Interconnect Characteristics	
Length: l	1/S
Unrepeated wire RC delay	1
Repeated wire delay	$\sqrt{1/S}$
Global Interconnect Characteristics	
Length: l	D_c
Unrepeated wire RC delay	$S^2 D_c$
Repeated wire delay	$D_c\sqrt{S}$

the propagation delay through the devices. Additionally, interconnect scaling reduces the delay through the local/scaled interconnects, while the signal delay throgugh the global interconnects is not affected by interconnect scaling. Although the functionality of a circuit and its operational characteristics can be preserved by following the scaling principles, technology scaling is always accompanied by many undesirable second-order effects. As the technology generations advance, some of these undesirable effects become more pronounced and new issues arise.

The current state-of-the-art technologies fabricate two-dimensional chips with upto seven layers of metal above the device layer. The interconnections in the circuits can be broadly classified into three types based on their pitches and aspect ratios[1] :

- Local Interconnects - are the Metal1 wires that are limited to short lengths and used to connect adjacent transistors.
- Intermediate Interconnects - that are found in the lower metal layers and are also limited to short lengths.
- Global Interconnects - that have lengths comparable to the chip size and are generally found in the upper metal layers.

The line lengths of the local and intermediate interconnects tend to scale with the technology generation and hence their impact on performance is minimal. The global interconnects have much larger line widths and lengths, thus posing many challenges in the current and future technologies[1].

Due to technology scaling, the widths of the global interconnects will approach the electron mean free paths. This implies that the resistivity of the interconnects will increase with technology and hence the delay of the signals being transmitted through these wires. Since the devices are also scaled, the relative lengths of the global interconnects increases and coupled with the increased resistivity will result in a large increase in the delay through the interconnects when compared to the delay through the devices. As the device density increases with technology scaling, the device count increases to such high numbers that a

higher number of global interconnects will be needed to establish all the connections between the circuit components. Repeaters can be used to mitigate the delay in global wiring, but they consume power and chip area. Copper interconnects and low-k Dielectrics have been proposed to decrease the resistivity of the interconnects. But issues like Copper Electromigration will become significant due to aggressive pitch scaling in the future technology generations. Moreover, the increase in the operating frequency and the decrease in the supply voltage that accompanies technology scaling along with the presence of numerous long, global interconnects increases the inductive coupling effects. This greatly increases the impact of crosstalk in the nanometer era.

Despite the use of copper interconnects and low-k dielectrics, Interconnects are predicted to dictate the performance of circuits in the future technology generations. Figure 1.1 compares the delays of optimized interconnects and the gates with the progression of the technology generations.

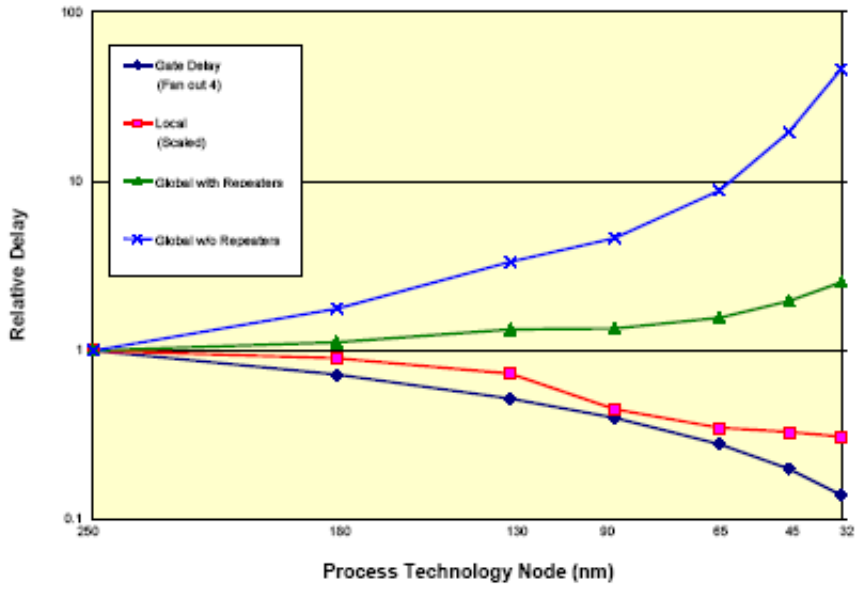

Figure 1.1 The Effects of Scaling on Gate and Interconnect Delays (Reproduced from [1])

Although research in the various stages of VLSI design cycle has helped alleviate the above-mentioned issues, radical ideas are needed to find long-term solutions for these issues and for the new problems predicted for the future technologies. One of the radical solutions proposed to handle the interconnect problems of the current and future technologies is Three Dimensional Integration[1].

1.2 Three-Dimensional Integrated Circuits

A Three-Dimensional Integrated Circuit is an integrated circuit in which active devices are found in two or more planes. A conventional Two-Dimensional circuit has exactly one (bottom) plane in which the active devices reside. All the layers above this device plane are the metal layers used for interconnections. Thus a 3D IC consists of several conventional two-dimensional integrated circuits stacked one on top of the other. Each individual two-dimensional circuit layer in the 3D IC is called a "Device Layer" or just "Wafer" and the wiring between these device layers are called "Inter-Layer Interconnects" or "Inter-Wafer Vias"(IWV). Thus a 3-Dimensional IC is a vertical stacking of multiple device layers that communicate with each other through Inter-Wafer Vias that are present between adjacent device layers.

1.2.1 Advantages of Three-Dimensional Integrated Circuits

Three-Dimensional Integrated Circuits are becoming increasingly popular in the current industry because of their numerous advantages over conventional two-dimensional ICs. Currently Three-dimensional integration technologies are being used to produce 3D memory chips[12] as the device density is much higher in a 3D IC when compared to a 2D IC with the same footprint area. Footprint Area is the product of the width and height of the package required for the 2D or 3D chip.

Three-dimensional ICs also form an elegant solution to the global interconnect problems faced in the DSM and nanometer technology generations. The intuition behind the vertical stacking of the active device layers is that the number of nearest neighbors for each transistor

or gate or module (depending on the level of the design hierarchy) is increased to six in 3-Dimensional ICs as compared to four in 2-Dimensional ICs. The availability of the third dimension for routing helps reduce the number and length of global interconnects in the circuit being designed. Rahman et al[23] characterized the wirelength distributions of interconnects in a 3D IC assuming negligible vertical interconnects and predicted reductions in the average and total wirelengths. Figure 1.2 shows the total wirelength distribution for both 2-D and 3-D ICs.

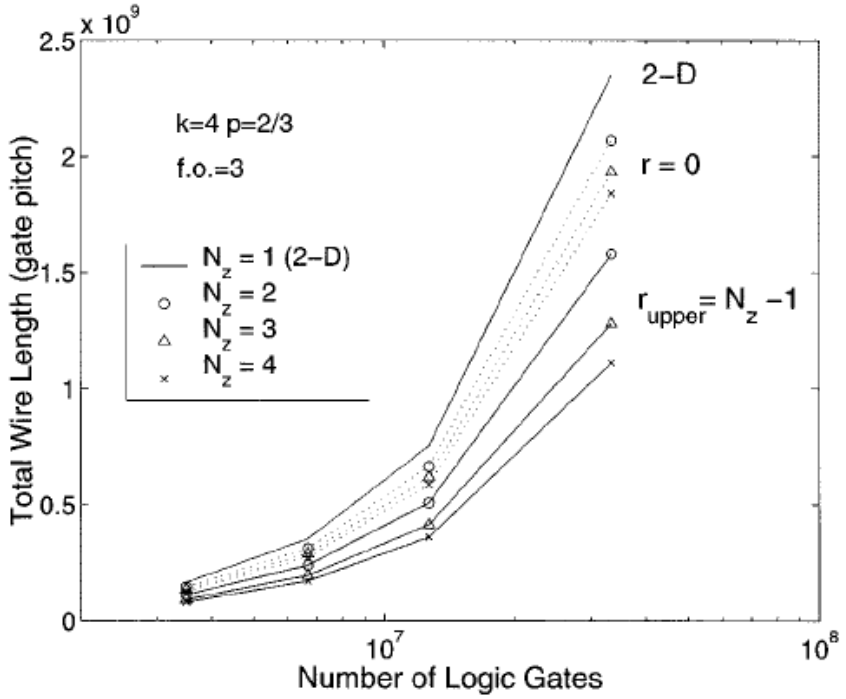

Figure 1.2 Total Wirelengths in 2D and 3D Circuits (Reproduced from [23])

Zhang et al.[34] modeled the 3D interconnects using a combination of both horizontal and vertical wires. A considerable reduction in both local and global interconnect lengths was predicted as expected but as the number of device layers exceeded six layers, the vertical interconnect capacitance produced an increase in the total interconnect capacitance. Nevertheless, it was observed that the 3-D ICs were two to three generations ahead of the 2D ICs in terms of delay and performance.

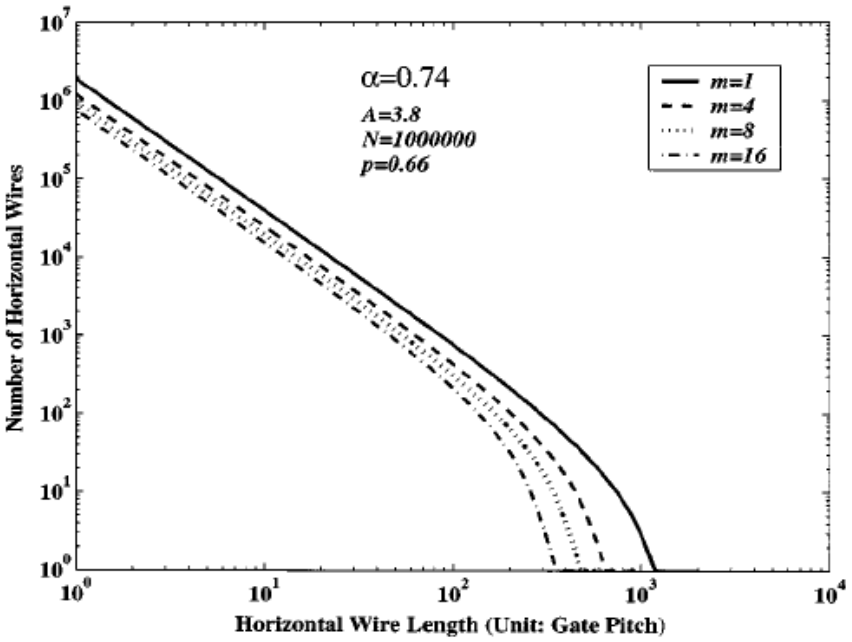

Figure 1.3 Wirelength Distribution for Horizontal Wires in 2D and 3D Circuits (Reproduced from [34])

Three-dimensional integration also promotes the fabrication of Mixed-Signal circuits in a monolithic 3D IC[28]. The digital and analog components can be fabricated on separate wafers and then bonded together using specific wafer-bonding technology. The inter-wafer bonding interface can be used to isolate the various analog and digital layers thus naturally alleviating the radiation issues in mixed-signal designs.

1.2.2 Realization of Three-Dimensional Integrated Circuits

There are numerous alternatives to implement the vertical stacking of devices in a 3-dimensional IC. True 3D ICs are produced using epitaxial methods[25] to produce devices on top of pre-fabricated substrates that have the lower level device layers. These techniques are very difficult and require non-conventional methods and costly equipment. Moreover the etching processes of the current layer must not damage the underlying devices and

interconnects. Due to the inherent difficulties of producing a true 3D IC, other alternative implementation methods have been proposed that include Wafer-Bonded 3D ICs, Through-Via 3D ICs, and Chip Stacking. All these methods can be broadly classified into the two categories:

- Packaging Techniques

  Packaging Approaches vertically stack pre-packaged two-dimensional integrated circuits. The Inter-layer interconnections are achieved using external inter-chip wires that connect the appropriate package pins of the corresponding 2-D chips. Vertical-MCMs(MCM-V) belong to this category. Package level 3-dimensional integration suffers from poor interchip bandwidth as the number of interchip vias are limited by the I/O pins available in the package selected.

- Monolithic Techniques

  Monolithic Approaches use fully-integrated technologies where unpackaged two-dimensional circuits, either dice or the entire wafers, are stacked or processed on top of each other. Packaging is done only when all the device layers have been fabricated or stacked into a single monolithic 3-dimensional die or wafer.

Monolithic Integration is achieved using either Wafer-bonding or Thin-Film fabrication technologies. Monolithic 3-dimensional ICs are divided into "True" 3D ICs and Vertically Integrated ICs or 2.5D ICs[3]. True 3D ICs use Thin-Film technologies such as Selective Epitaxial growth or Recrystallization to fabricate device layers on top of an existing substrate. 2.5D ICs are implemented by bonding two or more conventional 2-dimensional dies together using wafer-bonding or other bonding technologies. The bonding configuration will limit the scalability of the 2.5D ICs. Bonding configurations are classified into the following four types depending on whether one or both the dies being bonded are flipped:

- Face-to-Face

- Back-to-Back

- Back-to-Face
- Face-to-Back

Reif et. al[24] describe a fabrication technology used to produce scalable three-dimensional ICs using substrate thinning approaches.

1.3 Motivation

The primary motivation behind this thesis work is the increased scaling of transistors and interconnects and the various effects they have on the circuit design process. The increasing dominance of interconnects and the increase in device density has resulted in the need for floorplanners that can handle large number of devices on a single chip without compromising on both area and interconnect requirements. Thus technology scaling has resulted in a change in the objectives of the various phases of the physical design process as listed below:

- Simultaneous Optimization of Area and Wirelength

  The large increase in device density due to transistor scaling increases the gains of area savings. Thus even a small gain in chip area implies a large number of additional components that can be fit within the same chip. Thus equal importance must be given to both chip area and wirelength. But as the number of components to be packed in a chip increases, the complexity of the physical design process also increases. This increases the need for an efficient floorplanner.

- Global Interconnect Optimization

  The scaling of transistor sizes in Gigascale Integration technologies also leads to a large reduction in propagation delay through the devices. Although the interconnects are also scaled, transistor scaling leads to device sizes that are very small when compared to the lengths of the interconnects in the circuit. The signal propagation delay through the transistors becomes almost negligible when compared to the delay through the interconnects as shown in Figure 1.1. Even the use of repeaters can only

alleviate this problem for the next few technology generations. Thus the lengths of the global interconnects become the significant factor in the GSI era and dominate the performance of a circuit. Thus reduction of the length of the global interconnects is more important than reduction of total wirelength.

- Footprint Area Minimization

  The ever-increasing popularity of 3D ICs as a solution to the global interconnect problems in the future technology generations has introduced the need for a physical design framework tailored for 3D ICs. In 3D ICs, the package area is determined by the largest device layer width and largest device layer height. Thus floorplanners that balance the aspect ratios of all the device layers is needed.

1.4 Overview of Proposed Genetic Floorplanner

Although stochastic performance models[23][34] have predicted large wirelength and delay savings with the use of Three-Dimensional Integrated Circuits, efficient physical design tools are required to realize these potential savings. Designing circuits in the Giga-Scale era requires the use of hierarchical methods and IP modules to handle the highly increased complexities of the physical design process. The resultant layout designs will comprise of large sets of macrocells at each level of the design hierarchy. These factors greatly increase the importance of Macrocell Floorplanning in the GSI era.

The proposed genetic algorithm for 3-dimensional macro-cell floorplanning uses simple crossover operators to efficiently search for good floorplans. The genetic operators developed work transparently for both 2-dimensional and 3-dimensional floorplanning. The popular two-point crossover operator which has been successfully used with binary encodings has been modified to work with the sequence pair encoding for floorplans. The violations introduced by the traditional two-point crossover operator are removed by using a combination of the order crossover operator and the two-point crossover operator. The experimental results show that the proposed genetic floorplanner obtains 74.19% average reduction in Footprint Area, 61.04% average reduction in Total Wirelength, 59.45% average reduction

in the Length of the Longest Wire, and 2.29% average reduction in Total Area Usage. This is better than the stochastic predictions of around 50% savings in total wirelength reduction for 3-dimensional ICs with 4 device layers[23].

1.5 Thesis Organization

Chapter 2 presents some basic concepts in the areas of Floorplanning, and Genetic Algorithms that are needed to best understand this thesis. Chapter 3 briefly describes some of the current approaches used for two-dimensional and three-dimensional floorplanning algorithms. Chapter 4 discusses the proposed genetic approach for floorplanning in detail. Chapter 5 describes the experiments conducted and discusses the results obtained for all the experiments. Chapter 6 derives conclusions from the experimental results and discusses the direction for future work.

CHAPTER 2

FUNDAMENTAL CONCEPTS

2.1 VLSI Circuit Design

The VLSI Circuit Design Cycle is divided into a number of well-defined stages as shown below:

- System Specification – The first step of the VLSI design cycle is to obtain a comprehensive list of specifications for the desired system.
- Architectural Design – Once the system specifications are obtained a basic architecture is designed for the system.
- Behavioral or Functional Design – The behavioral design stage identifies the main functional units from the basic system architecture and the interconnections between them.
- Logic Design – In this step, the data and control flow of the various functional units in the system are derived at the Register Transfer level and tested. This description of a system is called an RTL description and is generally described using a Hardware Description Language(HDL). Boolean expressions and timing information are obtained for the various units and tested against the hihg-level specifications.
- Circuit Design – The boolean expressions obtained in the logic design stage are converted into a circuit description. The circuit description is built taking the performance and power requirements into account.

- Physical Design – The Physical Design stage converts the circuit description into a geometric description called the Circuit Layout. The layout of a circuit describes the geometries of the various devices and interconnects and the spacings between them. This geometric description is used to produce *masks* that are used by the Fabrication facilities to produce the chips. Thus the physical design stage has a direct impact on the area and the performance of the fabricated chip.

- Fabrication – The layout data from the physical design stage is converted into a set of photolithographic masks that can be used to produce each of the device and metal layers on the silicon wafer to produce the unpackaged chip, simply known as *die*.

- Packaging and Testing – The processed silicon wafer is diced into individual chips and packaged. Final testing of the chip is done before they are shipped to the market.

The final geometries of the chip heavily depend on the Physical Design stage as it is the final stage before fabrication. Additionally, physical design is the only stage that deals with the actual geometries of the devices and interconnects in the circuit being designed.

2.2 Physical Design of VLSI Circuits

The Physical Design stage in the VLSI Design Cycle is the process of determining the physical locations of the active devices and the interconnection structures between them in the chip's layout. The input to the physical design stage is a circuit description with optional timing specifications. The input circuit is processed in a series of well-defined phases as shown in Figure 2.1 to obtain an optimized circuit layout.

- Partitioning – Decomposition of the overall system into simpler sub-systems

- Floorplanning and Placement – Finding the best non-overlapping position for each module in the design for a minimal area chip layout

- Routing – Connecting the terminals of the interconnections between the placed modules using minimal wiring

- Compaction – Reduction of unused space in the chip layout
- Extraction and Verification – Extraction of the layout's functionality and testing against the desired functionality

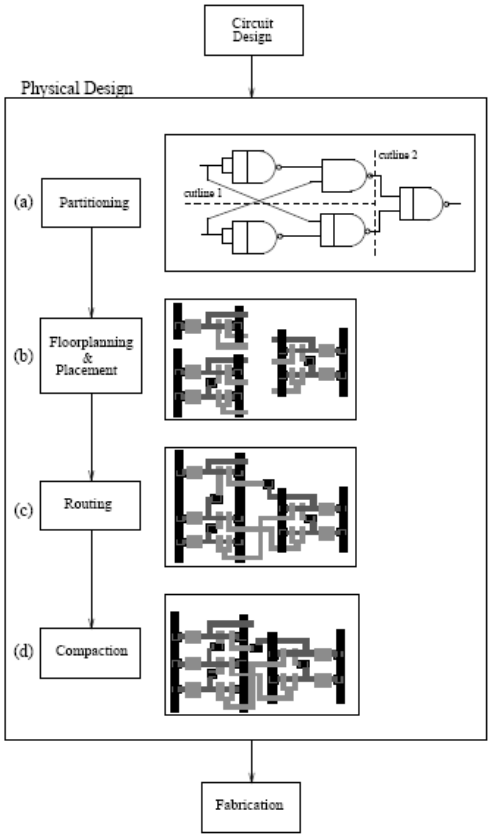

Figure 2.1 Physical Design Cycle for VLSI Circuits (Reproduced from [26])

2.3 Floorplanning and Placement

The Floorplanning phase is the second stage of VLSI Physical Design and deals with positioning all the modules obtained from the partitioning phase in a minimal bounding area. The Floorplanning and Placement phases have overlapping objectives and it is commonly considered that the Floorplanning problem is a generalized version of the Placement

problem. But there are two technical differences between the Floorplanning and Placement phases:

- The Floorplanning phase accepts two types of modules as its input, namely Fixed and Flexible modules.
  - *Fixed* or *Hard* Modules are the modules whose widths and heights are fixed.
  - *Flexible* or *Soft* Modules are the modules whose widths and heights can be changed within certain bounds that are defined by limits on the Aspect Ratio and the Area of the modules.

  In contrast, the Placement phase accepts only fixed modules in its input. A Floorplanning phase will precede the Placement phase if flexible blocks are present in the output of the Partitioning phase.
- Floorplanning calculates the *relative* positions of all the modules in the circuit and the optimal widths and heights for each of the flexible blocks in the input. Placement calculates the *actual* positions of all the (fixed) modules.

2.4 Problem Formulation

The input to the Floorplanning phase consists of

- a set of n modules or *blocks*: B_1, B_2, ... , B_n
- width (w_i) and height (h_i) of each module B_i
- Upper (AR_i^h) and Lower (AR_i^l) bounds on the Aspect Ratio for each flexible module
- List of interconnections between the modules known as the netlist of the circuit

The problem of Floorplanning deals with the determination of an optimal packing of all the modules in the input circuit such that there are no overlaps between any two modules. The optimality of the floorplan is generally measured by the area of the minimum rectangle

that bounds all the modules. With the advent of the DSM regime, wirelength metrics are added to the objective function being optimized. If flexible modules are present in the input, the floorplanning algorithm will additionally compute the width and height for each flexible module such that the constraints concerning the bounds on the aspect ratio and the area of the corresponding flexible module are satisfied.

2.5 Classification of Floorplans

Floorplans can be classified broadly into two categories based on the method used to allocate blocks to their positions in the floorplan layout. The two categories of floorplans are Slicing or Sliceable Floorplans and Non-Slicing or Non-Sliceable Floorplans.

2.5.1 Slicing Floorplans

Slicing Floorplans are those floorplans that can be obtained by recursively partitioning the entire rectangle that correponds to the current layout space being allocated to a sub-floorplan. The partitioning or cutting of the layout area is allowed only in the horizontal or vertical directions. An example of a Slicing Floorplan is shown in Figure 2.2a.

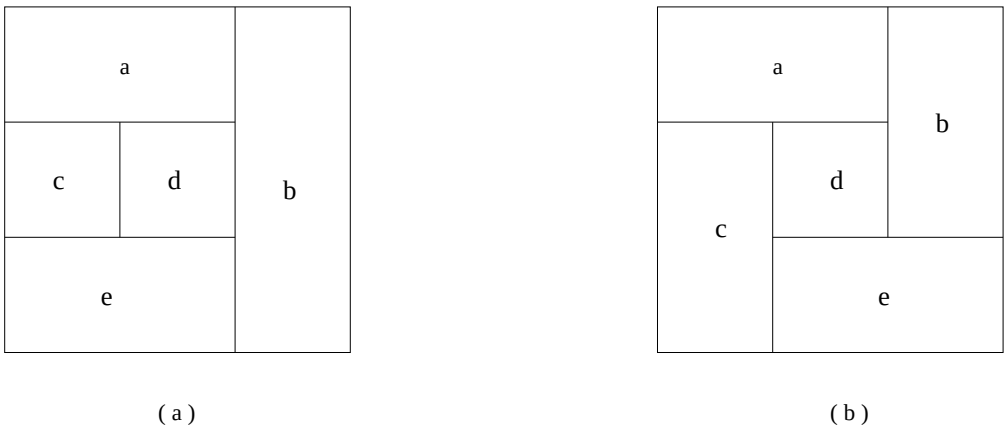

Figure 2.2 Types of Floorplans: (a) Slicing Floorplan (b) Non-Slicing Floorplan

2.5.2 Non-Slicing Floorplans

Non-Slicing Floorplans are floorplans that cannot be produced by complete recursive partitioning of the layout space. An example of a Non-Slicing Floorplan is shown in Figure 2.2b. Non-Slicing Floorplans require much more complex representation schemes such as the Bounded Slice-line Grid(BSG)[18] and the Sequence Pair(SP)[17] representations.

2.6 Floorplan Representations

Floorplan Representations play an important role as they have a direct effect on the runtimes of the floorplanning algorithms. Floorplan representations can be broadly classified into two categories based on their ability to represent both Slicing and Non-slicing floorplans.

- The first category consists of those representation schemes that can only represent Slicing Floorplans. Slicing Trees and Normalized Postfix Expressions fall under this category.

- The second category consists of floorplan representation schemes that can represent both slicing and non-slicing floorplans such as the Sequence Pair and Transitive Closure Graphs. The encoding of the floorplans into these representations and the decoding of the representations to form the floorplan are complex, time-consuming processes. But these representation schemes represent the entire set of floorplans that can be realized with the modules in the input circuit. Thus an efficient floorplanner is guaranteed to find one of possibly many optimal solutions contained in this large solution space, whereas a floorplanner using a slicing representation has to find an optimal solution from a (slicing) subset of the optimal floorplans.

2.6.1 Slicing Trees and Normalized Postfix Expressions

Slicing Floorplans can be easily represented using a Binary Tree structure as shown in Figure 2.3b. The internal nodes of the Slicing Tree represents a cut of the floorplan layout which can be either a horizontal cut(h) or a vertical cut(v). The leaf nodes represent

the modules in the floorplan. A Post-Order Traversal of the binary tree yields a Polish Expression that represents the Slicing Floorplan. The only drawback is the lack of a one-to-one correspondence between slicing floorplans and polish expressions. This drawback can be overcome by enforcing a preference between vertical cuts and horizontal cuts at the same level. The ordering of the cuts results in a Skewed Slicing Tree where no tree node and its right child have the same cut label. Post Order Traversal of the Skewed Slicing Tree results in a unique Normalized Polish Expression for every slicing floorplan. Floorplanning algorithms can use these simple Normalized Polish Expressions to represent the slicing floorplans in their optimization procedure.

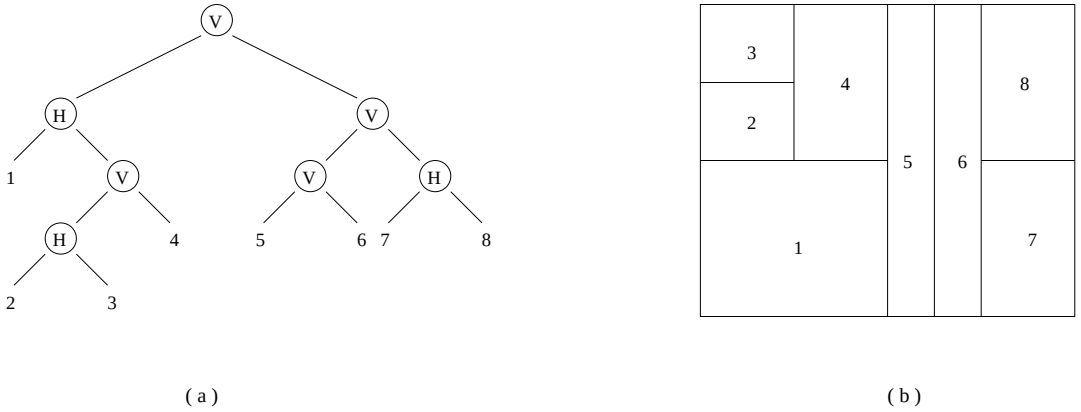

Figure 2.3 (a) Slicing Tree and (b) Floorplan for the Polish Expression – "123h4vh56v78hvv"

2.6.2 Sequence Pair Representation of Floorplans

Sequence Pairs[17] use two sequences(Γ^+, Γ^-) to represent the positional relationship between any two modules in the floorplan. Each sequence consists of a permutation of module names. The sequences are obtained using a process called Gridding. The first sequence(Γ^+), called the Positive(P) sequence or the X-sequence, gives the left-to-right order of the non-intersecting loci of module diagonals that run from the south-west corner to the north east corner of the module. The second sequence(Γ^-), called the Negative(N) sequence or the Y-sequence, gives the left-to-right order of the non-intersecting loci of the modules' diagonal that runs from north-west corner to the south-east corner of the module. The two

sequences alone are sufficient to obtain the relative positions of all the blocks in the floorplan. But a third sequence(Θ) is generally used to represent the orientations of the modules. Thus the entire floorplan layout can be drawn using the three sequences($\Gamma^+, \Gamma^-, \Theta$).

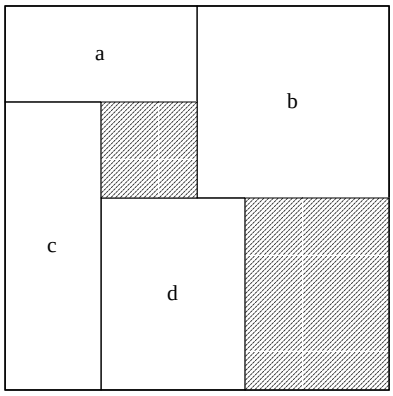

Figure 2.4 Floorplan for the Sequence Pair – "acbd, cdab"

2.6.2.1 Sequence Pair Decoding

The sequence pair representation can also be used in Placement as the absolute X- and Y- coordinates of all the modules can be computed from the sequence pair. The traditional method[17] to compute the coordinates of the modules constructs constraint graphs from the sequence pair and uses the constraint graphs to calculate the minimal bounding box area for the floorplan corresponding to the sequence pair. Faster methods that use longest common subsequence algorithms have been proposed[33].

2.6.2.2 Constraint Graphs Method

Horizontal and vertical relations between module pairs can be derived from the modules' positions in both the sequences. These relations can be used to construct horizontal (and vertical) constraint graphs. The vertices in the horizontal(vertical) constraint graph correspond to the modules in the floorplan. An edge form vertex i to vertex j in the horizontal constraint graph implies that module i is left of module j in the floorplan. The vertices are weighted with the widths(heights) of the modules. The longest path from the

source node to the sink node in this weighted horizontal(vertical) constraint graph yields the minimal width(height) of the floorplan. To find the positions of the individual blocks in the floorplan, the longest path in the constraint graphs from the source vertex to that module's vertex is computed.

2.6.2.3 Longest Common Subsequence Method

Wong et. al[33] noted that a common subsequence between the Γ^+, and Γ^- sequences corresponds to a path in the horizontal constraint graph. Thus the Longest Common Subsequence of the Γ^+, and Γ^- sequences will give the width of the floorplan represented by the sequence pair(Γ^+, Γ^-). The modules are weighted with their widths to obtain the width of the floorplan. The height of the floorplan can be computed in a similar fashion using the reversed Γ^+ sequence. Wong et al [33] devised an O(n^2) algorithm to compute the length of the longest common sub-sequence in a weighted sequence pair. By using a balanced search tree structure, the time complexity of the algorithm was reduced to O(n.logn).

2.6.3 Transitive Closure Graph Representation of Floorplans

Lin and Chang[15] proposed a novel approach of representing floorplans using two transitive closure graphs(TCG). A vertical TCG represents the vertical relations between the modules and a horizontal TCG represents the horizontal relations between the modules. The vertical(horizontal) TCG is obtained by finding the transitive closure of the vertical(horizontal) constraints graph used in the Sequence Pair approach. A placement and its corresponding TCGs are shown in Figure 2.5. The advantage of TCGs over Sequence Pairs is that the geometric relation between modules is obvious from the TCG. Fast O(n^2) methods were proposed for evaluating the cost of a floorplan represented using the TCG representation.

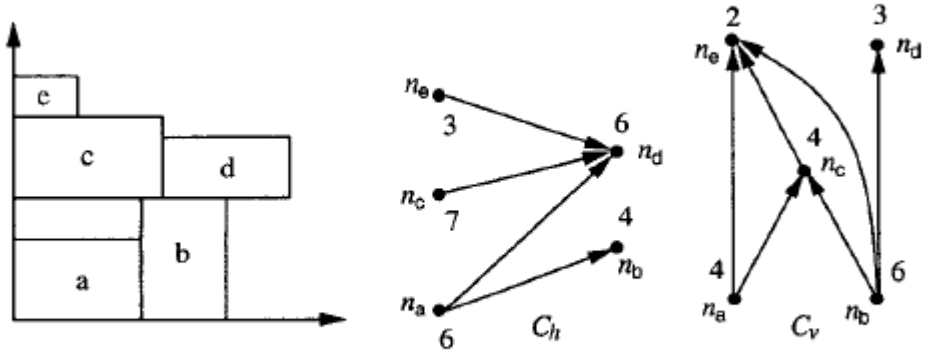

Figure 2.5 A Floorplan and its Corresponding TCG Representation

2.7 Significance of a Good Placement

Placement and Routing are the two most important phases in the Physical Design cycle as they decide the final quality of the layout in terms of the layout area and the wiring requirements. Due to the large number of devices and interconnects in a VLSI circuit, both these phases are highly complex and very difficult. The general Placement and Routing phases have been proven to belong to the NP-hard class of problems[26]. Thus automation of these two phases is required to obtain efficient layouts in a reduced amount of time.

2.8 Automation of Circuit Design

Computer Aided Design (CAD) tools are an essential part of the VLSI Design process. They can be used at any stage of circuit design to obtain a quick analysis of the current design or to explore other design alternatives. Moreover the sheer size of circuits and the exacting details required by the Physical Design phase require the elimination of human intervention as much as possible.

As the complexity of the circuits to be designed increases with technology scaling, the algorithms for automating circuit design should also change to address the new issues arising

due to the improved technologies. The traditional CAD tools primarily optimized the area of a design. As the technology approached the DSM regime, the newer CAD tools addressed issues such as power, and interconnect requirements. With the advent of the Nanometer era, the focus is shifting to global interconnects, cross-talk, and thermal issues. Due to the NP-hardness of the problems, stochastic search methods such as Simulated Annealing and Genetic Algorithms are frequently used to produce optimal circuit layouts.

2.9 Genetic Algorithms

A Genetic Algorithm(GA) is an optimization technique that uses a stochastic search strategy motivated by the process of natural evolution. Genetic Algorithms were introduced by John Holland[9] in the 1970s who borrowed some fundamental principles from the theory of genetics and evolution. Genetic Algorithms grew in popularity because of their ease of use, global perspective, robustness, and minimal problem information requirement[6]. Unlike other stochastic search strategies that operate using a single solution, Genetic Algorithms maintain an entire population of solutions or individuals from which an optimal solution is obtained. Thus Genetic Algorithms are implicitly parallel in nature.

2.9.1 Optimization Using Genetic Algorithms

Genetic Algorithms work on a population of individuals trying to produce better individuals in each generation and finally returning the best individual encountered as the solution to the problem. GAs start off with an initial population of individuals. The initial population generally consists of a fixed number of randomly generated individuals. This ensures that many diverse individuals with varied gene combinations are present in the initial population that will help produce varying types of fit individuals in the future generations. The process of evolution in nature is simulated by the use of three types of operators that evolve the individuals in the initial population into fitter individuals in the future generations. Offspring individuals are produced based on certain operator probabilities in every generation using three major types of operators:

- Reproduction
- Crossover
- Mutation

Individuals are selected for these operations from the population based on their fitness value and the offspring produced replace individuals in the population with low fitness. Thus the principle of survival of the fittest in nature is simulated in the genetic algorithm. GAs can be broadly classified into two types based on the replacement strategy used:

- Steady-State GAs and
- Generational GAs.

Steady State GAs produce offspring fewer in number than the entire population size in each generation. Thus only a subset of the population is replaced in every generation. Generational GAs replace the entire population with offspring individuals in each generation. After a certain number of generations, the GA outputs the best individual it has encountered as the final solution. This final solution corresponds to the best solution encountered in the entire search. An abstract top-level outline of a genetic optimizer is shown in Figure 2.6.

```
Genetic Algorithm Template()
{
    Initialize Population;
    for N generations do
    {
        Select Parents(); //for Crossover Operation;
        Crossover();
        Mutate();
        Replace();
        Update_Best_Individual();
    }
    Output Best Individual;
}
```

Figure 2.6 A Steady State Genetic Optimizer Template

2.9.2 Components of a Genetic Optimizer

A simple genetic algorithm will have the following components:

- population of strings,
- encoding mechanism,
- fitness function,
- selection mechanism,
- genetic operators, and
- control parameters.

2.9.3 Individual Encoding

Genetic Algorithms do not work on the problem instances directly. Instead GAs work on an encoding of the problem instance thus minimizing the amount of problem specific information needed. Each individual in the solution space is an encoding of the problem instance. The encoding scheme should preferably map each string to a unique individual in the population. The encoding scheme should also be easy to decode so that the fitness of the individual can be computed using minimal memory and time resources.

2.9.4 Fitness of an Individual

Each individual in the population is an encoding of the solution to the problem being solved. The fitness of an individual must reflect the quality of the solution to the problem. Since genetic algorithms follow natural selection, the fitness value of an individual will decide whether or not it will survive through the generations. The fitness of an individual may also decide whether or not the individual will be chosen as a parent for mating if fitness-based parent selection schemes are used.

2.9.5 Selection Mechanism

The Selection Mechanism is the scheme used by a Genetic Algorithm to select two individuals for the reproduction or crossover operations. The purpose of these operations is to allow substrings in the fit individuals in a population to survive for many future generations. Thus the parent individuals for these operations are generally selected based on their fitness values. This will promote survival of fitter genes in the offspring and should lead to fitter individuals in the future generations. Many selection mechanisms exist including proportionate selection, roulette wheel selection, and rank-based selection. The selection mechanism is chosen based on the nature of the problem being optimized and other factors including computation time and memory requirements.

2.9.5.1 Roulette Wheel Selection

In the Roulette Wheel Selection scheme, the probability that an individual will be chosen as a parent depends on the proportion of the individual's fitness when compared to the total fitness value of the current population.

2.9.5.2 Tournament Selection

In Tournament Selection, n individuals are randomly chosen from the current population to play in a tournament against each other. The winner of the tournament is selected as the parent. A tournament involving n individuals is called an n-way tournament.

2.9.6 Genetic Operators

The two most popular types of genetic operators are the crossover and mutation operators. The crossover operator performs a probabilistic exchange of information between two individuals of the population to produce a new individual. The crossover operator selects two parent individuals from the population based on a selection scheme. It then produces an offspring individual by using certain information from the first parent and the rest of

the information from the second parent. Thus the offspring individual inherits a subset of properties from both of its parents.

The mutation operator typically picks a random individual from the population and performs an inversion or some other random operation on the individual chromosome. After a certain number of generations, the crossover operator tends to produce offspring that are very similar to the parent individuals. The mutation operator plays a critical role in restoring lost genetic material or to provide diversity in the current population at these times. Thus the mutation operator helps prevent convergence to local optimal solutions.

2.9.7 Crossover Operators

Traditional Genetic Algorithms worked on binary encodings of the problem instances. These GAs used simple crossover operators such as the Uniform Crossover and the One-point Crossover to produce the offspring individual.

2.9.7.1 Uniform Crossover

The Uniform Crossover operator gives both the parents equal probability to pass on its genes to the offspring. In a simple implementation of the uniform crossover operator, a random binary string is generated with a length equal to the length of the parents' chromosomes. If the bit value at a position i of the string is 0, then the gene at position i of the offspring chromosome is filled with the i^{th} gene from the first parent. If the value was 1, then the i^{th} gene from the second parent will be used. Uniform mutation is a very disruptive operator[29] and tends to lose the genetic information after a certain number of generations. An example of the uniform crossover operator is shown in Figure 2.7.

2.9.7.2 One-point Crossover

The One-point Crossover operator randomly generates a single cut-point on both the parent individuals' chromosomes. The first part of the first parent's chromosome is used for the first part of the offspring's chromosome. The second part of the second parent's

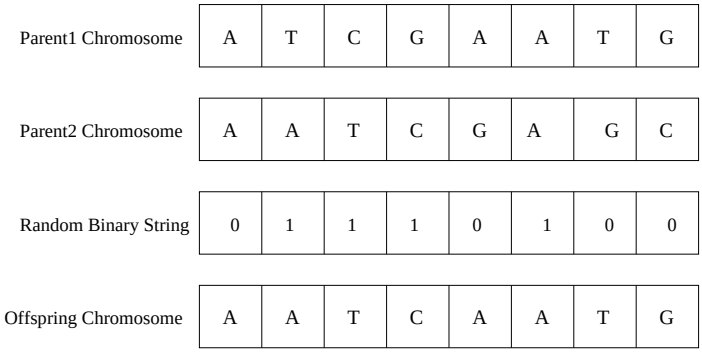

Figure 2.7 Illustration of Traditional Uniform Crossover Operator

chromosome is used for the second part of the offspring's chromosome. Figure 2.8 illustrates the one-point crossover operation.

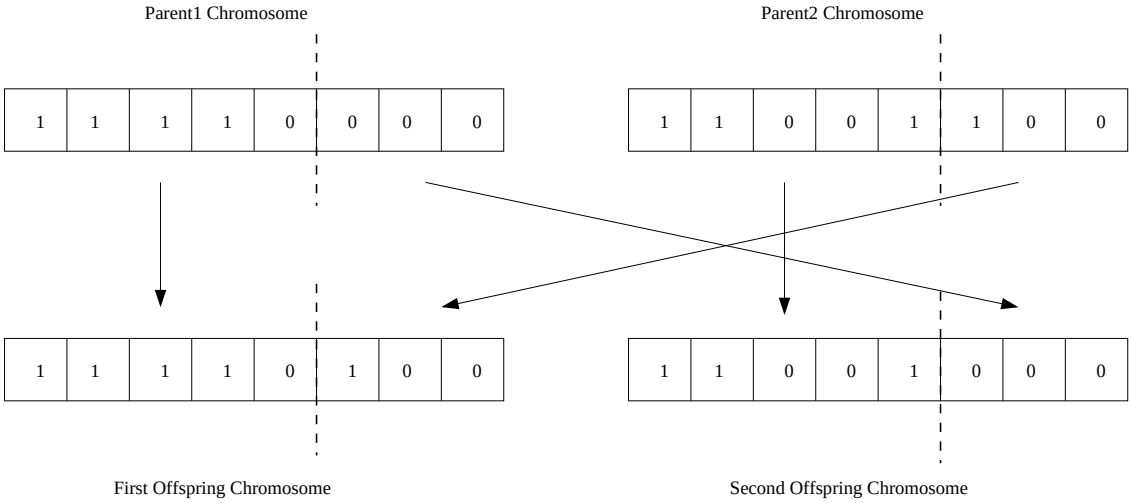

Figure 2.8 Illustration of Traditional One-point Crossover

2.9.8 Working Principle of Genetic Algorithms

The working principle of genetic algorithms is based on two theories, namely the schema theory [9] and the building block theory [6].

2.9.8.1 The Schema Theorem

A schema is a similarity template describing a subset of strings with similarities at certain positions of the string. The number of fixed positions that are to be compared for similarities is called the Order of the Schema. The distance between the outermost fixed positions is called the Defining Length of the Schema.

A schema with "k" fixed positions generates a "schema competition" among the different 2^k schemata that can be produced. Execution of a genetic algorithm generates 2^l simultaneous schema competitions. Thus the Genetic Algorithm search can be percived as a simultaneous competition among schemata to increase the number of their instances in the population.

2.9.8.2 The Building Block Hypothesis

Building blocks are schemata with high fitness values and small defining lengths. The Building block hypothesis states that strings with high fitness values can be located by sampling building blocks with high fitness values and combining the building blocks effectively.

Crossover operators tend to conserve the building block structures that present in their parents. Mutation generates radically new building blocks. And the selection mechanism provides favorable bias towards building blocks with high fitness values. By combining these operators with a good mechanism that juxtaposes the building blocks to form offspring strings, the genetic algorithm should be able to find better individuals in each generation leading towards individual with the optimal fitness.

2.10 Summary

This chapter introduced the VLSI Circuit Design Cycle and briefly discussed the various phases of the Physical Design Cycle. The Floorplanning phase of the VLSI Physical Design cycle was described in detail to emphasize the importance of the Floorplanning and Placement phase. The fundamentals behind Genetic Algorithm based optimization techniques

were described and the basic principles behind a GA-based optimizer were discussed to portray the intuition behind the working of a GA-based optimizer.

CHAPTER 3

2D AND 3D FLOORPLANNING: RELATED WORK

The general floorplanning and placement problems belong to the NP-hard class of problems. Many heuristic optimization techniques have been proposed to tackle the different variants of the floorplanning and placement problems. This chapter will discuss some of the state-of-the-art algorithms for the Macro-cell Floorplanning and Placement problems.

3.1 Review of 2D Floorplanners

Two-dimensional Floorplanning and Placement is a well-researched topic. Many optimization techniques have been applied to the 2D Floorplanning problem for optimizing various objective functions such as area and wirelength. Recently, the objective functions are changing from the traditional area metrics to reflect the interconnect issues of the future technology generations such as temperature and cross-talk. Hung et. al developed a genetic algorithm for thermal aware floorplanning[11]. Thermal aware floorplanning for three-dimensional ICs was proposed in [10]. This section presents some of the techniques used in 2D Floorplanning and Placement using convenional cost metrics.

3.1.1 Simulated Annealing Based Approaches

Simulated Annealing[14] is a randomized search based optimization technique. It belongs to the class of probabilistic, and iterative optimization algorithms. Simulated Annealing is an optimization technique that mimics the annealing process used for metals. In the annealing process used to obtain high-quality metals, the metal is first melted by heating it to a very high temperature and then cooled down slowly according to a cooling schedule until a freezing point is reached. At the end of the annealing process, all the

molecules of the metal settle down in their steady-state or low energy positions giving rise to a hard, durable metal. At any point of the annealing process, the collective energy of all the molecules in the metal define the quality of the metal and the positions of all the molecules in the metal constitute the configuration of the system. In a similar manner, the Simulated Annealing optimization algorithm examines configurations of the system at every temperature interval set by a cooling schedule and evaluates the quality of the configuration by associating a cost function with each configuration. Each configuration is a solution to the problem under consideration and the cost function is a measure of the quality of the solution. Simulated Annealing is characterized by three properties:

- Solution Space
- Move Set
- Cooling Schedule

The solution space consists of the entire set of solutions to be searched by the SA-optimization engine. The size of the solution space is dependent on the encoding used for the solution. The move set consists of the set of perturbations or moves that can be used on the current solution encoding to obtain a new encoding corresponding to a neighboring solution. The move set determines the efficiency of the SA optimizer. The cooling schedule decides the number of solutions evaluated by the SA optimizer and hence determines the trade-off between the quality of the solution and the run-time of the algorithm.

Simulated Annealing has been used successfully for solving the Floorplanning and Placement problems. It can be used in conjunction with all of the datastructures described in the previous chapter and other datastructures. The choice of the data structure defines the solution space being searched. For instance, if a slicing tree or a normalized polish expression is used, the solution space is restricted to the set of all slicing floorplans. If the Sequence Pair or TCG representation is used, the solution space is much bigger and includes the non-slicing floorplans in addition to the slicing floorplans. A different set of perturbation operations has to be defined in the move set according to the floorplan representation being

used for the search to be effective and efficient. Murata et al[17] proposed the following set of moves for their Simulated Annealing based floorplanner that uses Sequence Pairs for encoding the floorplans:

- M1 – Exchange the positions of 2 random modules in the first sequence
- M2 – Exchange the positions of 2 random modules in both the sequences
- M3 – Change the orientation of a random module

Lin and Chang[15] proposed a simulated annealing based floorplanner that uses the Transitive Closure Graph representation for floorplans. Two types of edges are identified in the TCGs, namely Reduction edge and Closure edge. The perturbation operators are defined in terms of the edges in the TCG rather than the modules. The following set of perturbation operators were proposed in the TCG-based Simulated Annealing floorplanner:

- Swapping the edges between 2 random modules in the TCG
- Reverse an edge between 2 random modules in the TCG
- Move an edge between 2 random modules in the horizontal(vertical) TCG to the vertical(horizontal) TCG
- Rotation of a module

Although the floorplan representation and the move set are different, all the SA-based floorplanners use the same annealing template as the optimization procedure. Simulated Annealing possesses the ability to climb out of local optima. Simulated Annealing accepts inferior solutions during its optimization process based on a probability that is dependent on the current temperature and the difference in the costs of the floorplans being evaluated. This is the primary reason for the ability of SA-based floorplanners to find globally optimal solutions.

3.1.2 Force-directed Approaches

Force-directed floorplanning and placement algorithms simulate the classical mechanics problem of a set of bodies(masses) attached to a set of springs. The bodies attached by a spring experience an attractive force between them according to Hooke's law and try to attain equilibrium positions so that the entire system attains a low-energy state. In force-directed placement, the modules are considered to be the bodies and the nets that connect them are considered to be the springs. Conventional force-directed methods represent the modules using the module centers. Thus only the centers of the modules are considered by the placement algorithm which uses the attractive forces between these points to find the best positions for the modules in the floorplan layout. But since only module centers are considered, the initial floorplan will have overlaps between the modules. The overlaps in macro-cell floorplanning can be large as the macro-cells can be of various shapes and sizes. Thus an overlap removal phase will always follow the initial floorplanning phase in force-directed floorplanning and placement. Several solutions have been proposed to integrate the cell overlap removal phase with the floorplanning phase such as introducing repulsive forces for unconnected blocks, using additional external forces to obtain an even cell distribution over the layout area, and creation of pseudo-cells for attracting or repelling other cells.

Mo et al[16] proposed a force-directed placement algorithm for the macro-cell placement problem. A filling force is used to reduce the cell overlaps. The nets are modeled using a star model instead of the traditional clique model in this approach.

3.1.3 Genetic Approaches

Genetic Algorithms simulate the process of evolution in nature to find solutions for optimization problems as explained in the previous chapter. Genetic Algorithms belong to the class of probabilistic, iterative optimization algorithms similar to Simulated Annealing. Genetic Algorithms can also be used in conjunction with any of the previously discussed datastructures. The crossover and mutation operators have to be defined according to the data structure being used.

Valenzuela and Wang[31] proposed a Genetic Floorplanner that uses Normalized Polish Expressions to represent the floorplans. The proposed floorplanning algorithm used both fixed and flexible modules to build the floorplans. A novel shaping curve method was used to obtain the optimal implementation for each flexible module. The crossover and mutation operators were defined so that only normalized polish expressions were produced by all the operators.

Sequence Pairs were used with Genetic Algorithms in [8]. An adaptive GA was proposed that used two crossover operators namely Placement-based Partially Exchanging Crossover(PPEX) and Common Topology Preserving Crossover(CTPX), along with a set of mutation operators to find the optimal floorplan. The crossover operator to be used was adaptively chosen based on the number of elapsed generations. The PPEX operator was chosen early as it produced a highly dissimilar offspring as compared to the parent individuals. The CTPX operator was favored towards the end to produce an offspring that was similar to the parents.

3.2 Review of 3D Floorplanners

With the growing popularity of three-dimensional ICs as a viable solution to the global interconnect problems of the current and future technology generations, much research has been conducted on developing a 3D IC design flow. Ohmura[20] proposed a gain-based greedy algorithm to obtain an initial 3D placement as early as 1998. Recently, Das et. al[2] extended the Magic layout editor[22] to support the design of 3-dimensional layouts. Deng and Maly[3] developed CAD tools for automating the design of 2.5-D ICs. Most of the research in 3D CAD tools has been directed towards floorplanning and placement as Floorplanning and Placement have an increased significance in 3D physical design. These phases primarily decide the footprint area and the number of Inter-Wafer Vias of the 3-dimensional layout. The 3D Floorplanning problem has the additional task of assigning modules to the various layers available in the vertical stack of 2D layers. This is known as

Table 3.1 Brief Summary of Existing Three-Dimensional Floorplanners and Placers

Group	Type of Floorplanner	Optimization Technique
Ohmura	Initial 3D Placer	Greedy Approach
Tanprasert	3D Placer	Analytical Approach
Deng and Maly	3D Macro-cell Floorplanner	Simulated Annealing
Deng and Maly	3D Standard-cell Placer	Min-Cut Approach
Das et al	3D Standard-cell Placer	Min-Cut Approach
Obenaus and Szymanski	3D Standard-cell Placer	Force-Directed Approach
Kaya et al	3D Standard-cell Placer	Force-Directed Approach
Shiu et al	3D Macro-cell Floorplanner	Simulated Annealing
Proposed Approach	3D Macro-cell Floorplanner	Genetic Algorithm

the Layer Assignment problem. Layer assignment has a huge impact on the final quality of the layout in terms of the footprint area of the 3D IC and also the number of IW-Vias.

There have been numerous efforts to extend 2D floorplanning algorithms into 3D floorplanning algorithms as shown in Table 3.1. Considering the 3D floorplan as a stack of 2D floorplans, existing 2D data structures have been used to represent the 3D floorplans by using a set of 2D structures. For instance, 2.5D IC floorplanner uses a set of "k" BSG structures to represent a "k"-layer 3D floorplan, using one BSG structure for each 2D layer. The 3D datastructures are then integrated with the Simulated Annealing floorplanning algorithm. The SA-based floorplanner uses a few additional interlayer moves in addition to the move set used for 2D floorplanning to search for an optimal 3D floorplan.

Numerous three-dimensional "Standard-Cell" floorplanners have been proposed that use force-directed based methods[19][13], Min-cut methods[2], and Mathematical Programming methods[30]. All the three-dimensional "Macro-cell Floorplanning" algorithms proposed in the literature employ Simulated Annealing based optimization methods.

3.2.1 Greedy Initial 3D Placement

An initial 3D placement that places strongly connected modules close together was proposed by Ohmura[20]. This approach emphasizes on the layer assignment problem since it assumes that fabrication of interwafer vias is much costlier than the intra-chip via fabrica-

tion. The proposed method divides the 3D chip region into cubical slots each of which can hold a module as shown in Figure 3.1.

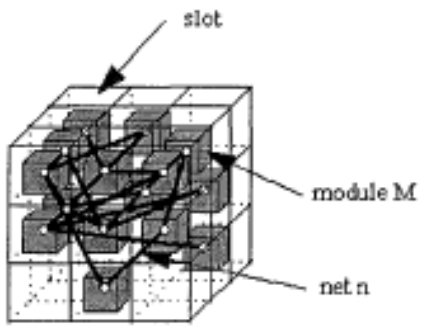

Figure 3.1 3D Chip Region Showing Cubical Slots Holding the Modules and the Nets Interconnecting the Modules (Reproduced from [20])

The modules are then moved towards their optimal positions using an iterative algorithm that moves the modules towards a slot that is nearest to the center of gravity of all the nets that it connects. The proposed method first computes the center of gravity of each net by using the centers of the modules connected by the net. Gains for moving each module towards the center of the net are computed for all the modules. The module with the maximum cumulative gain is moved. Priority is given to reduce the z-distances of nets by weighting the z-distances by a higher constant factor so that the total number of costly inter-wafer vias will be minimized.

3.2.2 Simulated Annealing Based Approaches

Deng and Maly[3] proposed a macro-cell floorplanner for $2.5D$ ICs based on Simulated Annealing. The data structure used to encode the 2.5D-floorplans was a set of BSG structures (one for each device layer in the stack). The move set for the SA-optimizer was modified to enhance floorplan optimization in all 3 dimensions. In addition to swapping

modules between two *rooms* in the same layer, swapping of modules between two rooms situated in different layers was allowed in the move set of the proposed floorplanner.

Shiu et al[27] proposed a three-dimensional macro-cell floorplanner for System-On-Package(SOP) designs. Sequence Pairs were used to represent the 3-dimensional floorplans. Each 2D floorplan corresponding to a layer in the 3D IC was represented using a sequence pair. In addition to the intra-layer moves allowed in the 2D SA-based floorplanning algorithm, two interlayer moves were added to the move set of the 3D floorplanner. The cost function used by the SA optimizer was modified to include additional terms that are specific to 3D floorplanning. The cost function used was a weighted function of the following factors:

- Total Area,
- Total Wirelength,
- Number of Inter-Wafer Vias,
- Footprint Area, and
- Dimension Deviation.

3.3 Summary

This chapter presented an overview of the existing two-dimensional and three-dimensional macro-cell floorplanners. Two-dimensional Floorplanners using optimization techniques including Simulated Annealing, Force-Directed Approaches, and Genetic Algorithms were discussed. The three-dimensional floorplanners discussed in this chapter are the only macro-cell 3D floorplanners currently available. This indicates the need for research and tool development for three-dimensional floorplanning as technology scaling is providing strong motivation for the production of three-dimensional integrated circuits in the future.

CHAPTER 4

THE PROPOSED GENETIC APPROACH

The proposed genetic algorithm is a macro-cell floorplanner that can perform both two-dimensional and three-dimensional floorplanning. Although the general macro-cell floorplanning problem involves both fixed and flexible modules, only fixed or hard modules are considered in the proposed algorithm. The following sections explain the genetic multi-dimensional floorplanning algorithm in detail.

4.1 Overall Flow of the Genetic Algorithm

The proposed genetic floorplanning algorithm uses a steady-state genetic algorithm template as shown in Figure 4.1. An initial population of independently generated random individuals is created. For a fixed number of generations, crossover and mutation are carried out using the individuals in the current population according to the crossover and mutation rates. The crossover and mutation rates are defined by the chosen values for the crossover and mutation probabilities respectively. The best individual encountered after the fixed number of generations is output as the solution for the floorplanning problem.

4.2 Components of the Genetic Floorplanner

The various components of the proposed GA-based floorplanner are listed below:

- Individual Encoding
- Individual Fitness
- Initial Population

```
Proposed Genetic Floorplanner()
{
    Generate Initial Population;
    for N_Generations do
    {
        for i in 1 to cx_Rate do
        {
            Select Parents(); //for Crossover Operation;
            Crossover();
            Replace();
        }
        for j in 1 to mut_Rate do
        {
            Mutate();
            Replace();
        }
        Find_Best_Individual();
    }
    Output best_Individual;
}
```

Figure 4.1 Overall Flow of the Proposed Genetic Flooplanning Algorithm

- Parent Selection
- Crossover Operators
- Mutation Operators

Details on all the above components will be discussed in the following sections.

4.3 Individual Representation

Each individual in the population is represented using a sequence pair representation[17]. Every sequence pair encoding corresponds to a valid two-dimensional floorplan. Thus the solution space for the genetic search engine consists of the entire set of slicing and non-slicing floorplans that can be represented using the sequence pair representation which is $n!$ without allowing any orientation changes. A three-dimensional floorplan is represented using a set of sequence pairs. A k-layer 3D floorplan is represented using a set of k sequence pairs.

The genotype of each individual chromosome that represents a 2D floorplan consists of three lists. The first list is X-sequence which is the ordered set of positive loci for the floorplan. The second list is the Y-sequence which is the ordered set of negative loci for the floorplan. The third list is the orientation of the blocks indexed by the block ordering of the positive loci list. Since the connectivity of the input circuit does not change, all individuals will refer to a common netlist for connectivity information.

4.4 Fitness of an Individual

The fitness of an individual determines whether the individual will survive through the simulated process of evolution. For two-dimensional floorplanning, the fitness of an individual is given by the floorplan's total area and the total wirelength required to route all the interconnections in the netlist. For three-dimensional floorplanning, the fitness of an individual is a weighted function of the following factors:

- Total Area - is the sum of the areas of all the layers in the 3D floorplan.
- Total Wirelength - is the sum of the half-perimeter wirelengths of all the nets in all the layers in the 3D floorplan. If the bounding boxes of the nets in adjacent layers do not overlap, then the wiring length required to connect the bounding boxes is added to the total wirelength.
- Footprint Area - is the product of the largest width and height among all the the layers in the 3D floorplan. The footprint area represents the package area of the chip.
- Total Number of Inter-Wafer Vias - is the sum of the number of inter-layer vias needed to connect all the nets.

Each of the fitness factors was normalized using a maximum value. The fitness value of each individual is a weighted function of all the normalized factors.

4.4.1 Fitness Evaluation

The fitness of an individual is measured using two types of metrics, namely area metrics and net metrics.

4.4.1.1 Floorplan Area Evaluation

Since the Sequence Pair representation is used to encode the floorplans, the Longest Common Subsequence method[33] is used for a quick evaluation of the floorplan bounding box area. The LCS method computes the X- and Y- coordinates for all the modules in the minimal area implementation of the floorplan in addition to the width and height of the floorplan. In 3D floorplanning, an intermediate sequence pair is constructed using the blocks in the current layer. The widths and heights of all the modules in the layer under consideration are grouped together and then passed to the LCS computation module along with the intermediate sequence pair. The width and height of each layer is stored so that the footprint area of the 3D floorplan can be computed.

4.4.1.2 Floorplan Wirelength Evaluation

The Total Wirelength needed to route all the interconnections in the circuit is estimated using the widely used Half-Perimeter method. The half-perimeter wirelength is computed using the centers of the modules connected by the net. In a 3D floorplan, the wiring required to route a net consists of the following components:

- Half-Perimeter wirelength in each layer that the net passes through,
- Non-overlap wirelength if the net's bounding boxes in adjacent layers that the net passes through do not overlap, and
- Inter-Layer vias, if the net spans multiple layers.

The wirelength due to the inter-layer vias is not computed. Instead the total number of inter-layer vias is calculated and used in the fitness function.

4.5 Initial Population

Genetic Algorithms start their search from an initial population. A successful GA-based optimizer needs a diverse set of individuals in the initial population. The diversity of the initial population assures a wide variety of genes that could combine in a favorable manner to obtain highly fit individuals. This is the primary reason that the initial population in the proposed approach is generated randomly. The positions of all the modules blocks in both the sequences of the sequence pair are generated independently and randomly. The orientations of each module are also randomly generated. A random initial population ensures that the genetic floorplanner will not converge quickly to a local optimum.

Although all the individuals in the initial population are randomly generated, they satisfy an area constraint. The area constraint is automatically generated using the total area of all the modules in the given benchmark circuit. This ensures that the individuals in the initial population have a set of good genes with respect to area and helps in faster convergence towards a globally optimal solution. The area constraint is set so that it is neither too restrictive so that generation of a diverse set of individuals is still possible nor too expansive so that many poorly fit individuals are avoided. For 3D floorplanning, the sum of the areas of all the layers is used for the area constraint test.

4.6 Genetic Operators

Crossover and Mutation operators are the only types of genetic operators used. The popular inversion operator is used as a mutation operator. Novel Crossover and Mutation operators for two-dimensional and three-dimensional macro-cell floorplanning using sequence pairs were developed. The operators were designed to optimize both area and wirelength simultaneously. The genetic operators were designed to build optimal top-level floorplans using good sub-floorplans. The genetic operators are described in detail in the following sections.

4.6.1 Crossover

The proposed Genetic approach uses two crossover operators to search for an optimal floorplan. Each crossover first selects 2 parent individuals whose genes will be passed on to the offspring.

4.6.1.1 Parent Selection

Parent Selection is generally performed based on the fitness of the individuals. Two parent selection strategies are employed by the proposed genetic floorplanner.

In the early stages, the randomly generated initial population has a large number of individuals that have low fitness values. If a simple genetic algorithm is used in conjunction with a tournament based selection strategy, the individuals with the low-fitness values will be replaced within the first few generations. These individuals are replaced before they can contribute any good genes towards the optimization process. Hence the proposed genetic floorplanner uses a "Worse-Than-Average fitness" parent selection scheme with high probability in the first few generations of the genetic optimization process. This gives a chance for even the low-fitness individuals to contribute whatever good genes they have to the optimization process through their offspring. After this initial phase, the population will comprise of a set of individuals with fitness values in the medium to high range.

The tournament selection strategy is the second parent selection strategy used by the proposed genetic floorplanner. In the tournament selection strategy used, a random sub-population of 8 individuals is gathered from the population to decide the two parents to be involved in the crossover operation. The size of the tournament is chosen based on the population size used in the algorithm as prescribed in [7]. For a population size of 100–200, [7] prescribes the tournament size to be set at 8 to guarantee continuous supply of good building blocks. A tournament is conducted within this sub-population based on their fitness values. The two finalists are selected as the parents for the crossover operation.

Initially, the probability of using either of the two parent selection schemes is the same. As the genetic floorplanner reaches a phase where all of the individuals in the population

have a medium to high fitness, the probability of using the tournament selection scheme is increased so as to dominate the "Worse-Than-Average" selection scheme.

The above parent selection strategies are used by the genetic floorplanner for both two-dimensional and three-dimensional floorplanning.

4.6.1.2 Modified Two-point Order Crossover Operator

The first crossover operator used by the proposed genetic floorplanner is based on two classical crossover operators used in several genetic algorithms, namely the Two-Point crossover operator and the Order crossover operator. The traditional 2-point crossover operator randomly chooses two cut-points that splits both the parents' chromosomes into three parts each. Two offspring are produced from the two parents using the method below. The first offspring is formed by concatenating the first part of the first parent's chromosome, the second part of the second parent's chromosome, and the third part of the first parent's chromosome together. The second offspring is formed by concatenating the first part of the second parent, the second parent of the first parent, and the third part of the second parent. Figure 4.2 illustrates the two-point crossover operator.

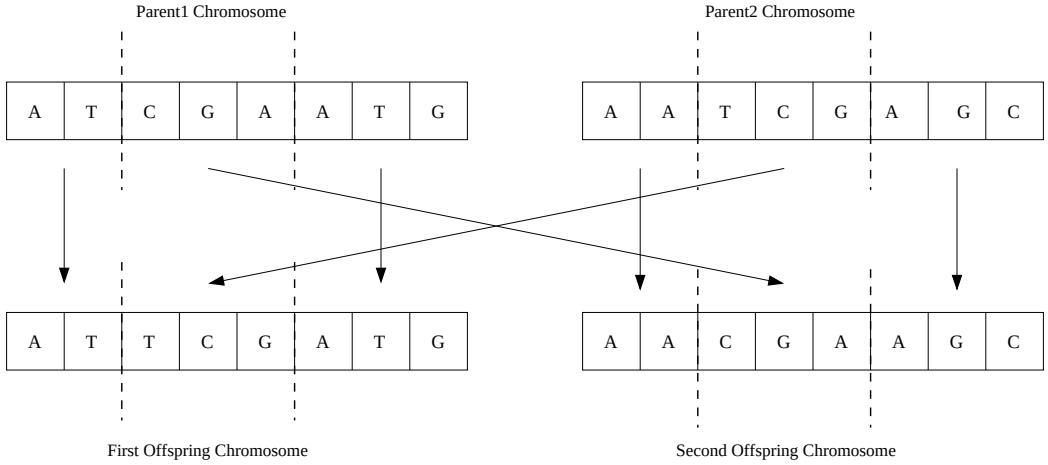

Figure 4.2 Traditional Two-point Crossover Operator

The original 2-point crossover operator was proposed for use with chromosomes that were encoded as binary strings that can accomodate duplicate genes in the individual's

chromosomes. If this traditional method is used with Sequence Pairs it will result in invalid Γ^+ or Γ^- sequences either due to duplication and/or deletion of module names. These violations must be removed to obtain a sequence pair that represents a valid floorplan. To eliminate the occurrence of such violations in the Offspring's Sequence Pair, the traditional 2-point crossover operator is combined with the order crossover operator to yield the Modified Two-point Order Crossover(MTOX) as shown in Figure 4.3.

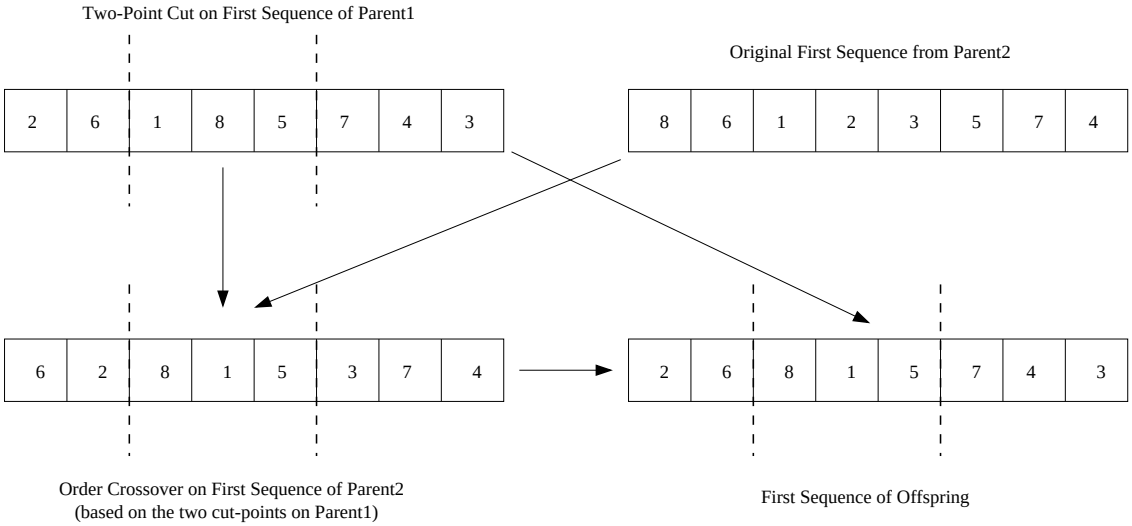

Figure 4.3 Modified Two-point Order Crossover Operator (MTOX)

4.6.1.3 Heuristic One-point Order Crossover Operator

The proposed genetic floorplanning algorithm uses a heuristic crossover operator in addition to the MTOX operator. While the MTOX operator is an unbiased operator that attempts to search the entire solution space for a good solution by randomly combining chromosome segments from the parent individuals in the offspring indiviudal, the Heuristic One-point Order Crossover operator(HOOX) tries to bias the search towards promising regions of the solution space by promoting the transfer of good subfloorplans from the parent individuals to the offspring individuals. The HOOX operator uses the traditional

One-point Crossover operator to partition the parent individuals into 2 subfloorplans as shown in Figure 4.4.

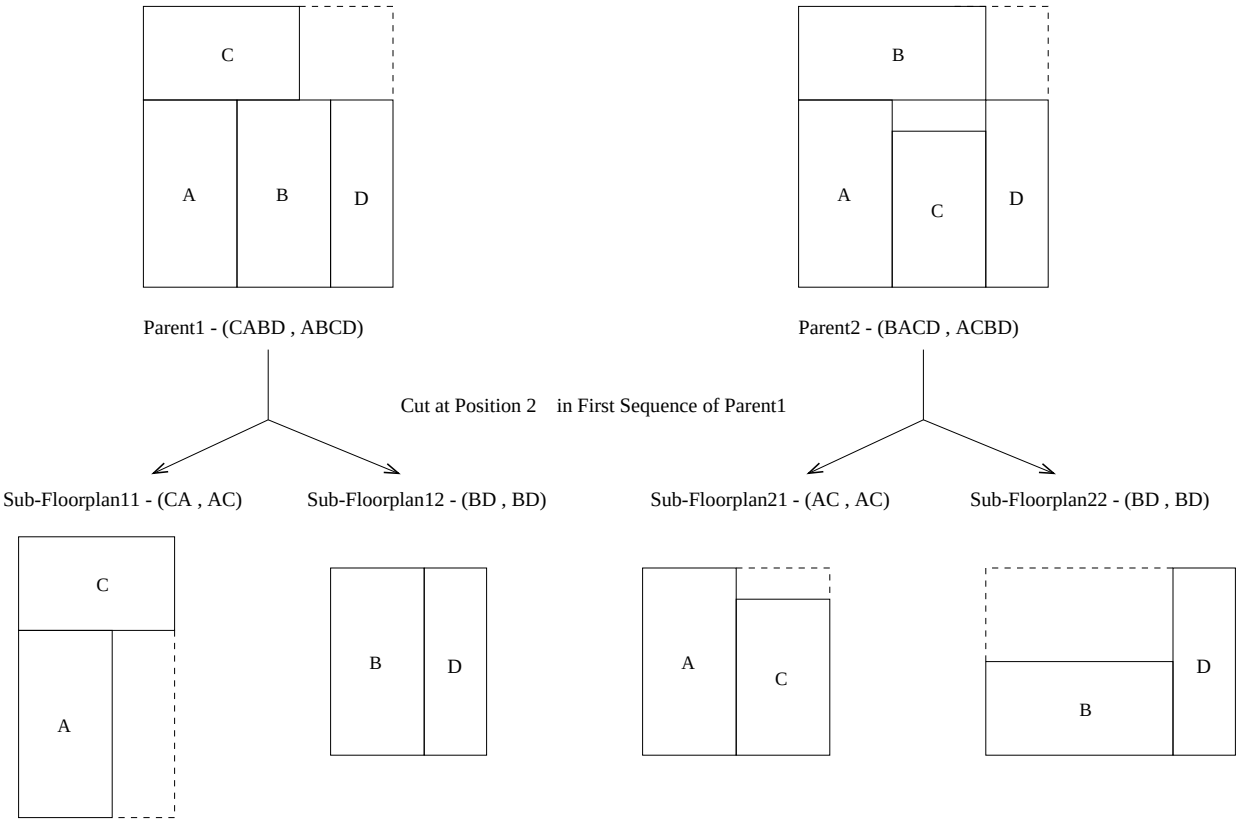

Figure 4.4 Sub-Floorplan Generation in Heuristic One-point Order Crossover Operator

The order crossover operator is used to eliminate any violations in the Sequence Pairs for the 2 sub-floorplans. Corresponding sub-floorplans from each parent individual are compared to see which parent's subfloorplan yields the better area usage. The two subfloorplans with the better area usage are then copied into the offspring individual's chromosome. The subfloorplans can be combined in 4 different configurations as shown in Figure 4.5. The configuration with the best bounding box area for the top-level floorplan is chosen for the offspring individual.

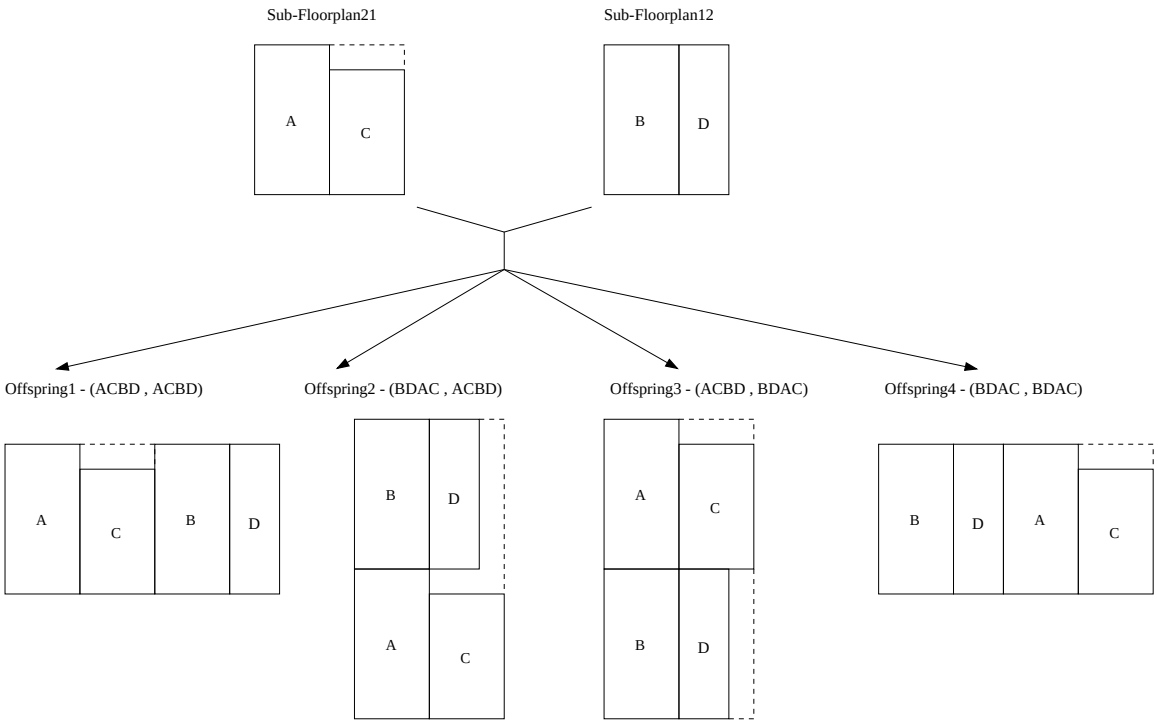

Figure 4.5 Four Offspring Configurations Generated from Good Sub-Floorplans in Heuristic One-point Order Crossover Operator

4.6.2 Mutation Operators

The mutation operator is generally used by genetic algorithms to produce diversity in the population. The mutation operator helps in avoiding quick convergence to local optima. In the proposed genetic floorplanning algorithm, mutation is performed on an individual that is randomly chosen from the current population. Instead of mutating the individual genes of a chromosome using a mutation probability, a mutation rate is defined using the mutation probability and the population size. Random individuals are then selected according to the mutation rate from the population and one of the mutation operators is randomly applied to the selected individual.

4.6.2.1 Mutation Operator - MUTOP1

The first mutation operator for the floorplanning algorithm picks 2 random modules from the individual and exchanges their positions in both the Γ^+ and the Γ^- sequences of the individual. This results in a swap of the locations of the two modules in the floorplans as shown in Figure 4.6.

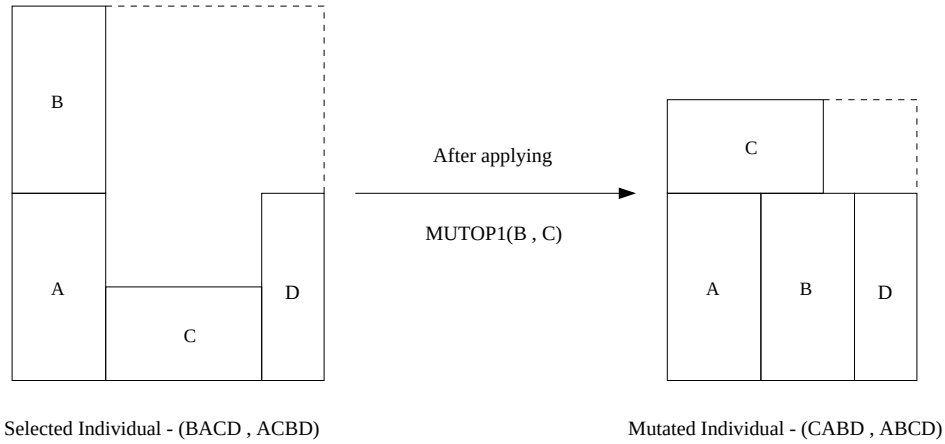

Figure 4.6 Mutation Operator - MUTOP1

4.6.2.2 Mutation Operator - MUTOP2

The second mutation operator for the floorplanning algorithm picks a random module from the individual and changes its orientation as shown in Figure 4.7.

4.6.2.3 Mutation Operator - MUTOP3

The third mutation operator is based on Inversion. A random subset of modules is chosen and their positions in the X- and Y-sequences of the sequence pair are reversed. Since inversion is typically used to promote linkage between module groupings, the positions of the modules in both the sequences are reversed so that the relative positions of selected subset of modules is still maintained.

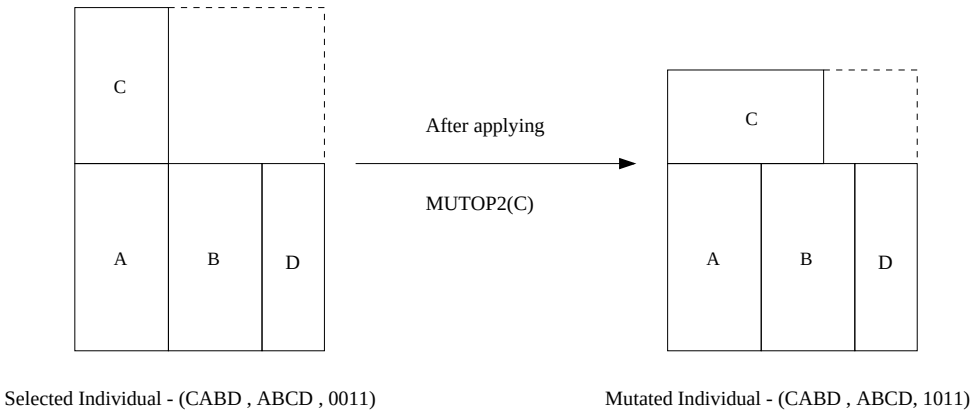

Figure 4.7 Mutation Operator - MUTOP2

4.6.3 Population Update

The offspring produced by the crossover operations replace individuals in the current generation's population to form the next generation's population. The individual(s) from the current generation that have very low fitness values are selected to be replaced in every generation. This ensures survival of individuals with good fitness values. For every offspring, the worst individual in the current population is identified. If the fitness of the offspring individual is better than the selected individual's fitness, then the offspring replaces that individual in the population. This process is repeated for every offspring individual produced in any generation.

4.7 Summary

In this chapter, a genetic algorithm based macro-cell floorplanner was presented that has the ability to produce both two-dimensional and three-dimensional floorplans. Novel crossover operators were described that work transparently on the multi-layer floorplans to produce optimal multi-layer floorplans.

CHAPTER 5

EXPERIMENTS AND RESULTS

The proposed genetic floorplanning algorithm was tested on two publicly available macro-cell benchmark suites. The first set of benchmark circuits are provided by the Microelectronics Center of North Carolina (MCNC)[5]. The second set of benchmark circuits are provided by the Gigascale Systems Research Center (GSRC)[21]. The functionalities of all the circuits in both the sets of benchmarks are unknown.

5.1 Description of the Benchmark Circuits

The MCNC benchmark suite comprises of five benchmark circuits. Relevant data about the benchmark circuits characteristics is provided in Table 5.1. The MCNC benchmark circuits are available in different formats including the old YAL, and VPNR formats. The new "blocks/nets" format files for the MCNC benchmark circuits were obtained from the GSRC Bookshelf Floorplanning Slot[21]. The new format for these benchmarks have three files for each circuit. The "blocks" file includes information on the modules such as number of modules, module names and vertex co-ordinates, and information on the pads such as number of pads and pad names. The "nets" file includes information on the interconnections in the circuit. The third file has information on the positions of the pads and any pre-placed blocks.

The GSRC benchmark suite comprises of sixteen circuits. The circuit characteristics for these benchmarks are provided in Table 5.2. The GSRC benchmark circuits are available only in the "blocks/nets" format. None of the GSRC benchmark circuits have any information on the positions of the pads.

Table 5.1 Characteristics of the MCNC Benchmark Circuits

Circuit Name	Number of Blocks	Number of Nets	Number of Pads
apte	9	97	73
xerox	10	203	2
hp	11	83	45
ami33	33	123	42
ami49	49	408	22

Table 5.2 Characteristics of the GSRC Benchmark Circuits

Circuit Name	Number of Blocks	Number of Nets	Number of Pads
n10	10	118	69
n10b	10	133	86
n10c	10	119	68
n30	30	343	212
n30b	30	350	227
n30c	30	390	271
n50	50	485	209
n50b	50	511	269
n50c	50	515	243
n100	100	885	334
n100b	100	806	374
n100c	100	855	323
n200	200	1585	564
n200b	200	1714	624
n200c	200	1532	533
n300	300	1893	569

5.2 Implementation Details

The genetic floorplanner was coded in C++ using Standard Template Libraries. Input Parsers were written in LEX and YACC to read in the input files and integrated with the floorplanning tool. Fitness computation for the individuals required software modules that calculated the Bounding Box Area for a sequence pair using the Longest Common Subsequence method described in [33] and the Half-Perimeter Wirelengths of all the nets in the Floorplan. The faster O($n$log$n$) algorithm detailed in [33] that uses Red-Black trees for LCS computation was coded using C++/STL. In the floorplanner, the modules are allowed only two degrees of orientation - flipped(rotated by 90 degrees) or not flipped.

Half-Perimeter Wirelength estimation was used for computing the wiring requirements of a floorplan. The centers of the bounding modules of the net were used to compute the half-perimeter wiring. For three-dimensional floorplanning, the total wirelength included any additional routing needed if the bounding modules of a net in adjacent layers did not overlap.

5.3 Experimental Setup

The proposed genetic floorplanner can handle both 2D and 3D floorplanning. Experiments were run to generate both 2D and 3D floorplans for all the circuits in both the benchmark suites using the same set of parameter values. Table 5.3 lists all the important parameters and their values used in the proposed GA-based floorplanner.

Table 5.3 Values for the Parameters Used in the Genetic Floorplanner

Parameter	Value
Population Size	200
Number of Generations	1000
Crossover Probability	0.8
Mutation Probability	0.05

5.4 Experimental Results

The genetic floorplanner accepts as an argument the desired number of layers in the floorplan. The default number of layers in a floorplan is set to 1. Thus the genetic floorplanner will produce two-dimensional floorplans by default. By changing the number of layers, three-dimensional floorplans can be produced without any re-compilation. Since the floorplanner is GA-based, the floorplanner was run 11 times for each benchmark file. Table 5.4 shows the best results for bounding box area, total wirelength, and run-time obtained on the MCNC benchmarks for two-dimensional floorplanning.

Table 5.4 Two-Dimensional Floorplanning Results on the MCNC Benchmark Circuits

Circuit Name	Area (mm^2)	Total Wirelength (mm)	Runtime (sec)
apte	46.925	211.772	295
xerox	19.831	276.157	380
hp	9.159	70.889	572
ami33	1.200	31.339	802
ami49	37.812	677.943	1821

Table 5.5 Two-Dimensional Floorplanning Results on the GSRC Benchmark Circuits

Circuit Name	Area	Total Wirelength	Runtime (sec)
n10	233352	13572	51
n10b	232162	13320	38
n10c	237965	16129	35
n30	224910	42164	180
n30b	210090	37789	181
n30c	235304	45005	195
n50	217000	96320	324
n50b	217765	86633	327
n50c	217434	93928	328
n100	210166	165819	831
n100b	186095	151134	795
n100c	201601	162156	810
n200	231876	399787	1631
n200b	234515	354058	1756
n200c	228435	387374	1801
n300	396268	663019	2781

Table 5.5 lists the best results obtained for area, total wirelength, and runtime for two-dimensional floorplanning for all the GSRC benchmark circuits. Table 5.6 shows the best results for the GSRC benchmark circuits for Three-Dimensional Floorplanning with the number of layers set to four. In addition to the best results for Total Area, Total Wirelength, and run-time, the table also shows the best results obtained for Footprint Area, and the number of Inter-Wafer Vias.

Table 5.6 Three-Dimensional Floorplanning Results on the GSRC Benchmark Circuits

Circuit	Footprint Area	Total Area	Total WL	No. of Vias	Runtime (*sec*)
n10	65780	228171	4671	57	72
n10b	63440	228548	4031	45	58
n10c	70596	238883	5179	57	56
n30	58912	223529	16806	137	278
n30b	56115	210503	14930	136	297
n30c	62944	241042	17384	148	261
n50	56840	220596	39040	330	469
n50b	57330	220237	35886	300	493
n50c	57479	222024	38411	311	485
n100	53410	207512	68858	556	1176
n100b	46842	179354	64065	533	1038
n100c	51027	200548	68288	560	1189
n200	55200	218007	162583	1247	2623
n200b	55626	216450	138766	1078	2428
n200c	53410	210388	155908	1234	2412
n300	91964	356788	262643	1585	3942

5.5 Comparisons

The results obtained for both the two-dimensional and three-dimensional floorplanning experiments were compared against existing floorplanners. A discussion of the various comparisons is given in the following sections.

5.5.1 Three-Dimensional Versus Two-Dimensional Floorplanning

The best results obtained from 11 runs of the three-dimensional floorplanning experiments were compared against the best results obtained from the two-dimensional floorplanning experiments respectively. The comparisons were made in terms of both area and wirelength metrics. Table 5.7 shows the percentage improvements obtained by using 3D floorplans over 2D floorplans. Rahman et. al[23] predicted total wirelength savings of about 50% when using 3-D ICs with 4 device layers, neglecting the vertical wirelengths. The total wirelength results obtained from the experiments do not account for the vertical wires. We only report the total number of Inter-Wafer Vias needed for completing all the

interconnections in the 3-D layout. The experimental results for total wirelength savings show an average reduction of 61.04% which is better than the predicted savings. Moreover the length of the longest wire in each of the circuits reduces by an avverage of 59.45% as shown in Table 5.7. All of these wirelength savings are achieved without any increase in deadspace area. This is illustrated by the average savings of 2.29% reported in the total area used.

Table 5.7 Comparison of 3D Floorplanning and 2D Floorplanning Results for the Proposed Genetic Floorplanner on the GSRC Benchmark Circuits

Circuit	Percentage Savings in			
	Footprint Area	Total WL	Total Area	Longest WL
n10	71.81	65.58	2.22	63.81
n10b	72.67	69.74	1.56	61.42
n10c	70.33	67.89	-0.39	66.28
n30	73.81	60.14	0.61	58.01
n30b	73.29	60.49	-0.20	59.41
n30c	73.79	61.37	-0.35	61.83
n50	73.81	59.47	-1.66	56.61
n50b	73.95	58.58	-0.07	54.84
n50c	73.56	59.11	-2.11	57.33
n100	74.59	58.47	1.26	62.39
n100b	74.83	57.61	3.62	58.10
n100c	74.69	57.89	0.52	58.58
n200	76.19	59.33	5.98	57.01
n200b	76.28	60.81	7.70	59.32
n200c	76.62	59.75	7.90	55.39
n300	76.79	60.39	9.96	60.84
Avg Savings	74.19	61.04	2.29	59.45

5.5.2 Three-Dimensional Floorplanning

The results obtained from the three-dimensional floorplanning experiments were compared against the results of the Simulated Annealing based 3-D floorplanner[27]. The Footprint Area, the Total Wirelength, and the number of Inter-Wafer Vias results for the GSRC benchmark circuits alone were reported in [27]. Comparisons were made in terms of both footprint area and total wirelength. Table 5.8 shows the results of both the comparisons

and also reports the percentage savings obtained in area and wirelength by the proposed aprroach. The proposed genetic floorplanner obtains an average reduction of 4.97% in footprint area and an average reduction of 11.09% in total wirelength.

Table 5.8 Footprint Area and Total Wirelength Comparisons With the SA-based 3D Floorplanner

Circuit	GA Floorplanner		SA Floorplanner[27]		% Savings	
	Area	Total WL	Area	Total WL	Area	Total WL
n10	65780	4671	66740	8395	1.44	44.36
n10b	63440	4031	74469	10198	14.81	60.47
n10c	70596	5179	71760	11640	1.62	55.51
n30	58912	16806	68420	38106	13.90	55.90
n30b	56115	14930	60984	34850	7.98	57.16
n30c	62944	17384	69153	33353	8.98	47.88
n50	56840	39040	60973	44749	6.78	12.76
n50b	57330	35886	61650	34019	7.01	-5.49
n50c	57479	38411	60960	46619	5.71	17.61
n100	53410	68858	56166	54347	4.91	-26.70
n100b	46842	64065	50400	58270	7.06	-9.95
n100c	51027	68288	53760	56278	5.08	-21.34
n200	55200	162583	56977	123765	3.12	-31.36
n200b	55626	138766	56635	119849	1.78	-15.78
n200c	53410	155908	53345	148634	-0.12	-4.89
n300	91964	262643	83232	165433	-10.49	-58.76
	Average Savings				+4.97	+11.09

The reductions in total wirelength are found mainly in the benchmark circuits that have lesser than 100 modules. The genetic floorplanner performs worse than the SA-based floorplanner for the benchmark circuits with 100, 200 and 300 modules. This is primarily due to the fact that the population size chosen for the experiments is 200. When the input circuits have more than 50 modules, the initial population does not contain sufficient number of diverse individuals. The genetic floorplanner has to work with a limited set of gene combinations and hence does not perform better than the Simulated Annealing approach. For bigger populations, the Genetic Algorithm will perform better but the convergence to a good solution will be slower. Figure 5.1 shows the normalized fitness convergence plots for the "n300" benchmark circuit for population sizes of 200, 400, 600, 800, and 1000. It is seen

that there is an improvement in the fitness with increasing population size as predicted. Figure 5.5.2 shows an improvement in the wirelengths of the large benchmark circuits with increasing population size. All the fitness values in the figures were obtained using the same values for all other parameters. But the optimal parameter values for different population sizes will be different. Hence the improvement obtained in the fitness values can be increased further by computing the optimal parameter values and then running the experiments. The computation of the optimal parameter values is not trivial and will be explored as part of future work. It should be noted that a reduction in fitness in the plot indicates an improvement as the fitness is the weighted sum of the normalized area and wirelength metrics.

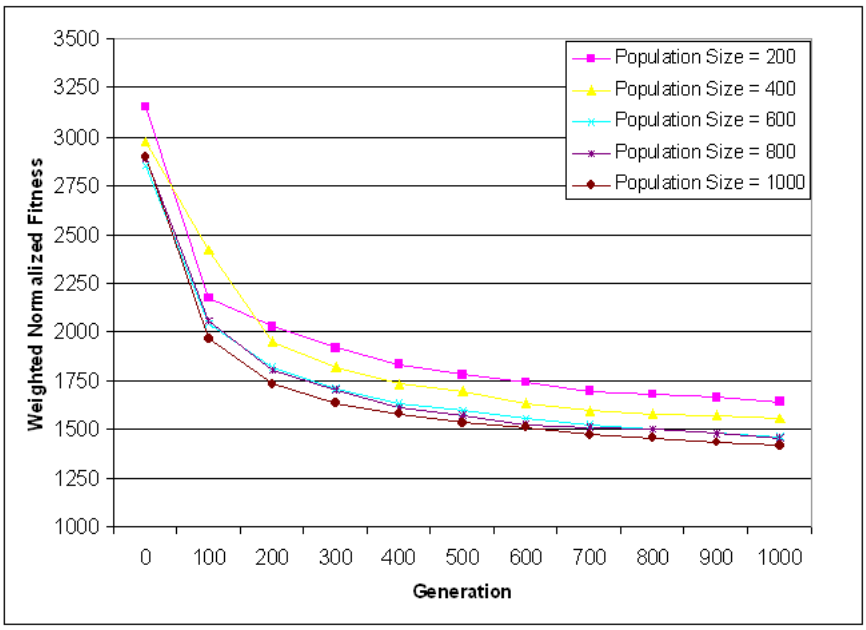

Figure 5.1 Normalized Fitness Convergence Plot for $n300$ Benchmark Circuit

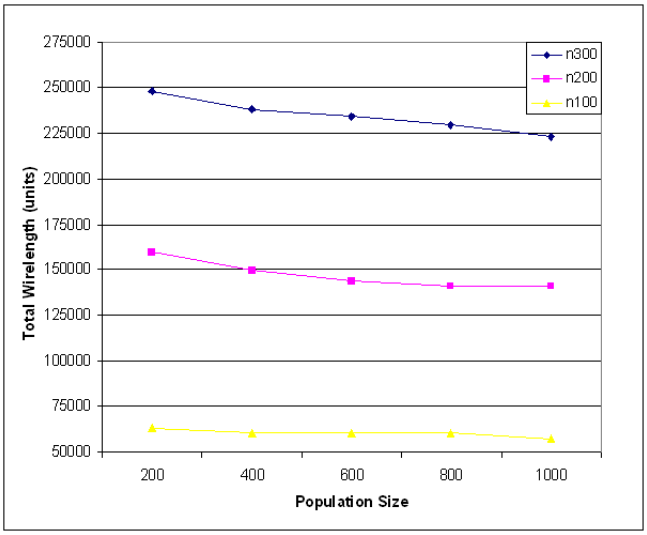

Figure 5.2 Graph Illustrating Improvements in Total Wirelength With Increase in Population Size

5.5.3 Two-Dimensional Floorplanning

The results obtained from the two-dimensional floorplanning experiments were compared against the best and the average results of the Quadratic Programming based Floorplanner using Less Flexibility First principles[4]. The QP based Floorplanner reports best area, average area, and average wirelength results for two large macro-cell MCNC benchmarks, namely ami33 and ami49. The area and wirelength comparisons are shown in Table 5.9. On an average, the proposed genetic floorplanner obtains 12.06% savings in total wirelength for a small 2.11% increase in area.

Table 5.9 Area and Wirelength Comparisons With the LFF-based Quadratic Programming 2D Floorplanner

Circuit	Proposed Floorplanner			QP Floorplanner[4]			% Savings		
	Area		WL	Area		WL	Area		WL
	Best	Avg	Avg	Best	Avg	Avg	Best	Avg	Avg
ami33	1.200	1.259	39.398	1.177	1.200	45.3	-2.66	-4.95	+13.03
ami49	37.812	38.898	782.282	36.6	39.18	879.9	-3.31	+0.72	+11.09
	Average Savings							-2.11	+12.06

The results obtained from the two-dimensional floorplanning experiments were also compared against the two-dimensional floorplanning results of the SA-based Floorplanner[27]. The SA based Floorplanner reports the best results for area, and total wirelength for the GSRC macro-cell benchmark circuits. The area and wirelength comparisons are shown in Table 5.10. On an average, the proposed genetic floorplanner obtains 10.63% savings in total wirelength along with an area savings of 1.30%.

Table 5.10 Area and Wirelength Comparisons With the SA-based 2D Floorplanner

Circuit	GA Floorplanner		SA Floorplanner[27]		% Savings	
	Area	Total WL	Area	Total WL	Area	Total WL
n10	233352	13572	258152	18164	9.61	25.28
n10b	232162	13320	251178	15128	7.57	11.95
n10c	237965	16129	268865	19880	11.49	18.87
n30	224910	42164	245115	54586	8.24	22.76
n30b	210090	37789	234574	45931	10.44	17.73
n30c	235304	45005	233867	55979	-0.61	19.60
n50	217000	96320	231431	104395	6.24	7.74
n50b	217765	86633	237266	94790	8.22	8.61
n50c	217434	93928	234567	106562	7.30	11.86
n100	210166	165819	210378	180413	0.10	8.09
n100b	186095	151134	185868	169767	-0.12	10.98
n100c	201601	162156	208616	185215	3.36	12.45
n200	231876	399787	214349	393644	-8.18	-1.56
n200b	234515	354058	208960	336236	-12.23	-5.30
n200c	228435	387374	206954	394358	-10.38	1.77
n300	396268	663019	329589	658162	-20.23	-0.74
	Average Savings				+1.30	+10.63

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

A Genetic Algorithm based macro-cell floorplanner that can produce both two-dimensional and three-dimensional floorplans has been developed. Novel cross-over operators were defined that build floorplans by combining good sub-floorplans. A heuristic crossover operator that explicitly optimizes the floorplan area has been developed. Mutation operators that work on the Sequence Pair encoding of two-dimensional and three-dimensional floorplans have been developed.

The developed floorplanner needs many parameter values to be set such as the population size, crossover and mutation probabilities, and the number of generations. These values determine the efficiency of the floorplanner and must be set based on some analysis using techniques from Design of Experiments. Explicit methods that optimize wirelengths are to be developed in the future. Adaptive probabilities for the crossover operators are to be derived so that the best crossover operator is chosen automatically by the genetic optimizer. With these enhancements, the proposed genetic floorplanner will become a robust and highly efficient floorplanning tool.

REFERENCES

[1] Semiconductor Industry Association. International Technology Roadmap for Semiconductors. 2005 Edition.

[2] S. Das, A. Chandrakasan, and R. Reif. Design Tools for 3-D Integrated Circuits. pages 53–56. IEEE Asia and South Pacific Design Automation Conference, 2003.

[3] Y. Deng and W.P. Maly. Interconnect Characteristics of 2.5-D System Integration Scheme. 2001.

[4] S. Dong, Z. Yang, X. Hong, and Y. Wu. Module Placement based on Quadratic Programming and Rectangle Packing using Less Flexibility First Principle. pages 61–64. IEEE Int'l Symposium on Circuits and Systems, 2004.

[5] MCNC Electronic and Information Technologies [online]. www.mcnc.org .

[6] D.E. Goldberg. Genetic Algorithms in Search, Optimization, and Machine Learning. 1989.

[7] D.E. Goldberg. The Design of Innovation: Lessons from and for Competent Genetic Algorithms. 2002.

[8] K. Hatta, S. Wakabayashi, and T. Koide. Solving the Rectangular Packing problem by an Adaptive GA based on Sequence Pair. pages 181–185. Asia and South Pacific Design Automation Conference, 1999.

[9] John H. Holland. Adaptation in Natural and Artificial Systems. 1992.

[10] W.-L. Hung, G. Link, Y. Xie, N. Vijaykrishnan, and M.J. Irwin. Interconnect and Thermal-aware floorplanning for 3D Microprocessors. International Symposium on Quality of Electronic Design, 2006.

[11] W.-L. Hung, Y. Xie, N. Vijaykrishnan, C. Addo-Quaye, T. Theocharides, and M.J. Irwin. Thermal-aware floorplanning using genetic algorithms. pages 634–639. International Symposium on Quality of Electronic Design, March 2005.

[12] Matrix Semiconductor Inc. www.matrixsemi.com/technology/index.html.

[13] I. Kaya, M. Olbrich, and E. Barke. 3-D Placement considering Vertical Interconnects. pages 257–258. IEEE Int'l SOC Conference, 2003.

[14] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by Simulated Annealing. 1983.

[15] J.M. Ling and Y.W. Chan. TCG: A Transitive Closure Graph-based Representation for Non-Slicing Floorplans. 2001.

[16] R.K. Mo, A. Tabbara, and R.K. Brayton. A force-directed macro-cell placer. pages 177–180. Proceedings of IEEE Intl. Conf. on CAD, 2000.

[17] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani. VLSI module placement based on rectangle-packing by the sequence pair. pages 1518–1524. IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, 1996.

[18] S. Nakatake, K. Fujiyoshi, H. Murata, and Y. Kajitani. Module Placement on BSG-Structure and IC Layout Applications. 1996.

[19] S.T. Obenaus and T.H. Szymanski. Gravity: Fast Placement for 3-D VLSI. pages 298–315. ACM Transactions on Design Automation of Electronic Systems, July 2003.

[20] M. Ohmura. An Initial Placement for 3-D VLSI. pages 195–198. IEEE Symposium on Circuits And Systems, 1998.

[21] GSRC Bookshelf2 Floorplanning Slot [online]. www.cse.ucsc.edu/research/surf/GSRC.

[22] J.K. Ousterhout, G. Hamachi, R.N. Mayo, W.S. Scott, and G.S. Taylor. The Magic VLSI Layout System. June 1984.

[23] A. Rahman and R. Reif. System Level Performance Evaluation of Three-Dimensional Integrated Circuits. pages 671–678. IEEE Transactions on VLSI Systems, December 2000.

[24] R. Reif, A. Fan, K.-N. Chen, and S. Das. Fabrication technologies for three-dimensional integrated circuits. ISQED, 2002.

[25] G. Roos, B. Hoefflinger, M. Schubert, and R. Zingg. Manufacturability of 3D epitaxial-lateral-overgrowth CMOS circuits with three stacked channels. Microelectronic Engineering, 1991.

[26] Naveed Sherwani. Algorithms for VLSI Physical Design Automation. Springer-Verlag Publishing Company, 1999.

[27] P.H. Shiu, R. Ravichandran, S. Easwar, and S.K. Lim. Multi-layer Floorplanning for Reliable System-on-Package. pages 69–72. IEEE Symposium on Circuits And Systems, 2004.

[28] Shukri J. Souri, Kaustav Banerjee, Amit Mehrotra, and Krishna Saraswat. 3-D Heterogeneous ICs: A technology for the Next Decade and Beyond. 5th IEEE Workshop on Signal Propagation on Interconnects, 2001.

[29] M. Sriniνas and L.M. Patnaik. Genetic Algorithms: A Survey. pages 17–26. IEEE Computer, June, 1994.

[30] T. Tanprasert. An Analytical 3-D Placement that reserves routing space. pages 69–72. IEEE Symposium on Circuits And Systems, 2000.

[31] Christine Valenzuela and P.Y. Wang. VLSI Placement and Area Optimization using a genetic algorithm to breed Normalized Polish Expressions. pages 390–401. IEEE Transactions on Evolutionary Computation, August 2002.

[32] Neil H.E. Weste and David Harris. CMOS VLSI Design: A Circuits and Systems Perspective, 3rd edition. Addison-Wesley and Pearson Education Inc., 2005.

[33] D.F. Wong, X. Tang, and Tian R. Fast Evaluation of Sequence Pair in Block Placement by Longest Common Subsequence Computation. pages 106–111. Design And Test in Europe, 2000.

[34] R. Zhang, K. Roy, C.K. Koh, and D.B. Janes. Stochastic Interconnect Modeling, Power Trends, and Performance Characterization of 3-D Circuits. pages 638–652. IEEE Transactions on Electron Devices, April 2001.