CAD for a 3-Dimensional FPGA

by

Vikram Chandrasekhar

B.Tech., Computer Science and Engineering, Indian Institute of Technology Madras

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2007

Author ....
Department of Electrical Engineering and Computer Science
May 21, 2007

Certified by ....
Anantha P. Chandrakasan
Professor of Electrical Engineering
Thesis Supervisor

Certified by ....
Donald E. Troxel
Professor of Electrical Engineering
Thesis Supervisor

Accepted by . [illegible] ....
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

CAD for a 3-Dimensional FPGA

by

Vikram Chandrasekhar

Submitted to the Department of Electrical Engineering and Computer Science
on May 21, 2007, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

In this work, the benefits of using 3-D integration in the fabrication of Field Programmable Gate Arrays (FPGAs) are analyzed. A CAD tool has been developed to specify 3-dimensional FPGA architectures and map RTL descriptions of circuits to these 3-D FPGAs. The CAD tool was created from the widely used Versatile Place and Route (VPR) CAD tool for 2-D FPGAs. The tool performs timing-driven placement of logic blocks in the 3-dimensional grid of the FPGA using a two-stage Simulated Annealing (SA) process. The SA algorithm in the original VPR tool has been modified to focus more directly on minimizing the critical path delay of the circuit and hence maximizing the performance of the mapped circuit. After placing the logic blocks, the tool generates a Routing-Resource graph from the 3-D FPGA architecture for the VPR router. This allows the efficient Pathfinder-based VPR router to be used without any modification for the 3-D architecture.

The CAD tool that was developed for mapping circuits to the fabricated 3-D FPGA is also used for exploring the design space for the 3-D FPGA architecture. A significant contribution of this work is a dual-interconnect architecture for the 3-D FPGA which has parasitic capacitance comparable to 2-D FPGAs. The nets routed in a 3-D FPGA are divided into *intra-layer* nets and *inter-layer* nets, which are routed on separate interconnect systems. This work also proposes a technique called *I/O pipelining* which pipelines the primary inputs and outputs of the FPGA through unused registers. This 3-D architecture and I/O pipelining technique have not been found in any of the works proposed so far, in the area of 3-D FPGA design. It is shown that the Dual-Interconnect I/O pipelined 3-D FPGA on an average achieves 43% delay improvement and in the best case, up to 54% for the MCNC'91 benchmark circuits.

Thesis Supervisor: Anantha P. Chandrakasan
Title: Professor of Electrical Engineering

Thesis Supervisor: Donald E. Troxel
Title: Professor of Electrical Engineering

Acknowledgments

I would like to thank Professor Anantha Chandrakasan and Professor Donald Troxel for their support. I enjoyed the learning opportunity in Digital ICs and Systems group under their supervision and guidance. I would also like to thank other members of Anantha group whose valuable input have enhanced my insight in my research. In particular, I would like to thank Frank Honore, Payam Lajevardi and Tao Pan for their help and advice. In addition, I would like to thank the FPGA research group at the University of Toronto for the use of their CAD tool's code base. I'm grateful to Prema Srinivasan for her confidence and uncessant inspiration. Finally, I would like to thank my parents, Rajyasri Chandrasekhar and Chandrasekhar Subramanian and my sister Varsha Chandrasekhar, for their love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview of FPGAs

Field Programmable Gate Arrays (FPGAs) are semiconductor devices that contain logic components connected by a regular, hierarchical interconnect system. The distinguishing characteristic of FPGAs is their on-field programmability which allows the logic functionality of an FPGA to be reprogrammed even after the manufacturing process. The logic components in the FPGA mostly consist of memory elements such as registers or even complete blocks of memory that can be configured to hold any desired state. The hierarchical interconnect system is also programmable which allows the logic components to be connected in a variety of network configurations. Therefore the re-programmability of FPGAs is achieved by a fixed underlying architecture, which does not cater to any particular logic circuit. This lets FPGAs have a lower non-recurring cost, shorter design cycle and enables them to be re-programmed in the field to circumvent manufacturing defects. FPGAs are generally slower and have a larger die size compared to Application Specific Integrated Circuits (ASICs) that are designed for a particular logic functionality throughout their lifetime. Initially, FPGAs were used mostly for prototyping and emulation systems in the design process for ASICs. However, recently, FPGAs have become popular for a variety of mainstream products in networking, telecom, digital signal processing and in consumer electronics.

FPGAs can be classified based on the technology used to program the FPGA,

- Antifuse FPGAs - The devices are configured by burning a set of fuses. Once the chip is configured, it cannot be reprogrammed any more.
- Flash FPGAs - These devices can be reprogrammed several thousands of times and retain their configuration even when the power is switched off. However, these devices are relatively more expensive and take several seconds for reconfiguration.
- SRAM FPGAs - These devices are the most popular kind of devices as they offer unlimited re-programming using SRAM cells to hold the circuit configuration that is loaded into the FPGA. They offer very fast reconfiguration, with some devices such as the Xilinx [27] allowing even partial reconfiguration.

1.2 2-D FPGA architecture

A 2-D FPGA consists of an array of configurable logic blocks (CLBs) that are connected by a 2-dimensional grid of metal channels, as shown in Figure 1-1. Each CLB contains a small amount of digital logic in the form of Look-up Tables or LUTs which implement boolean logic functions using small random-access memories. A random-access memory of n bits can be used to implement any one of the 2^{2^n} boolean functions that have n inputs and a single output. The outputs of the LUTs are also connected to registers whose output can be chosen instead of the direct LUT output. Thus a CLB can be programmed by a small amount of memory to implement sequential logic as well as combinational logic. The grid of metal channels connecting these CLBs together, contain switch boxes at the grid-points, which can connect the intersecting metal channels to each other using programmable switches. The programmable memory in the CLBs as well as the memory controlling the switches in the switch boxes, together form the *configuration memory* of the FPGA. Any logic circuit can be implemented in the 2-D FPGA by setting the memory in the CLBs to a particular state and connecting the CLBs in a particular manner. Therefore, any logic circuit

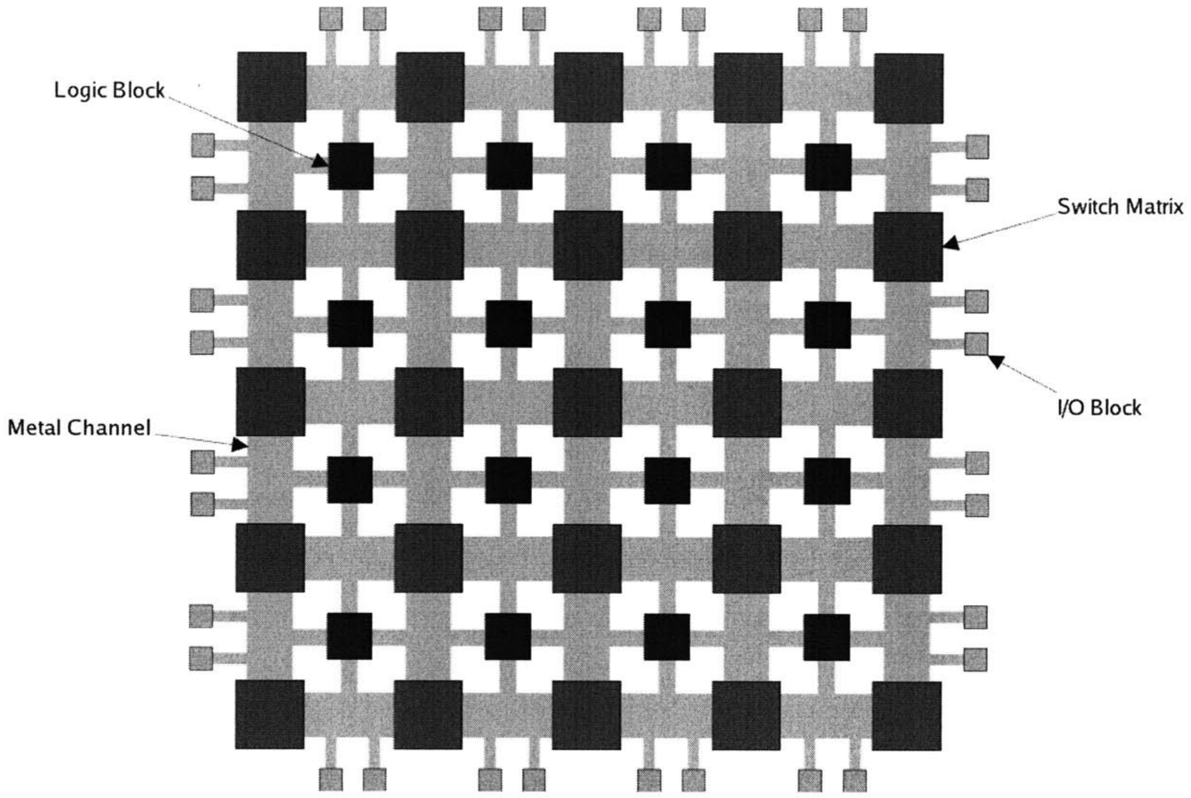


Figure 1-1: Island-style 2-D FPGA

is *functionally equivalent* to a particular state of the configuration memory of the FPGA. For this particular state of the configuration memory, the FPGA behaves exactly like the mapped circuit for all possible inputs to the circuit. The individual components of the 2-D FPGA architecture are described in detail below.

1.2.1 Configurable Logic block

The Configurable Logic Block (CLB) as shown in Figure 1-2 contains the boolean logic that are the equivalent of gates in an ASIC circuit. The logic consists of multiple Look-Up Tables (LUTs) and flip-flops. A LUT is a 16-bit SRAM accessed by a 4-to-1 MUX tree, representing a boolean logic function with four inputs and one output. The output of the LUT is also connected to a flip-flop and a 2-to-1 MUX is used to choose between the direct LUT output and the latched output. The latched output is used to implement sequential circuits in the FPGA. The CLB has four 16-bit Lookup

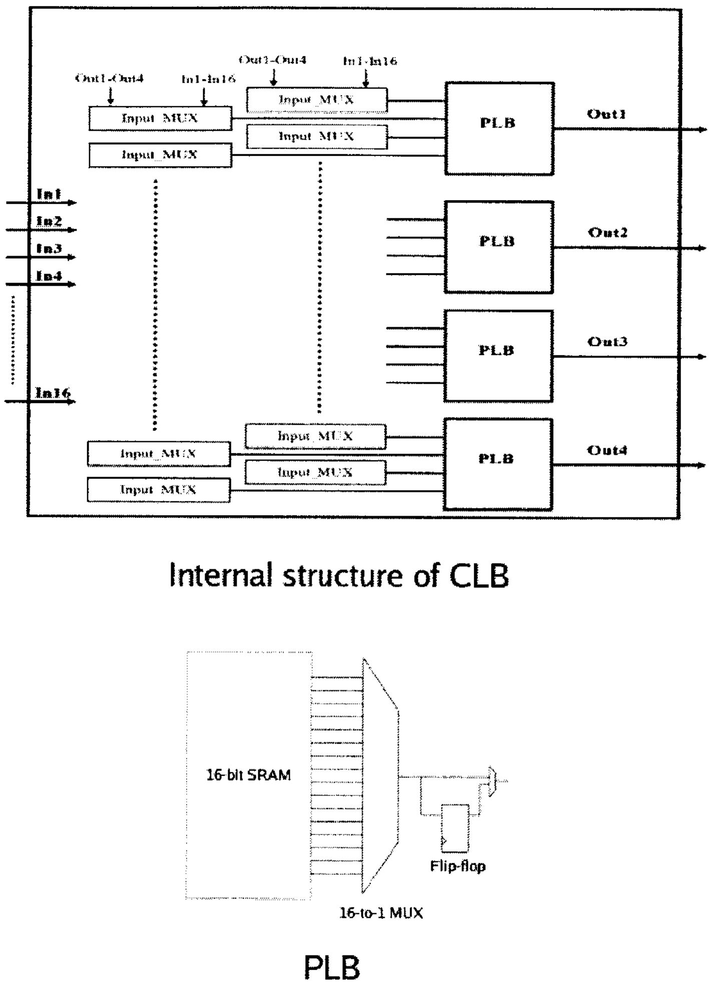


Figure 1-2: Configurable Logic Block

tables (LUTs), 16 block input pins and 4 block output pins. The 16 block input pins are internally muxed into the inputs of the LUTs. Hence the block input pins are *logically equivalent* i.e. a router can assign any one of the 16 block input pins to a particular LUT input pin. This allows the router to dynamically pick a particular block pin as the desired sink depending upon the existing congestion constraints. The 4 block output pins that can be connected to any of the 4 LUT outputs through muxes. Hence the block outputs are also logically equivalent. The logic block also has internal routing between the logic block outputs and the inputs, which allows the LUT outputs to be muxed along with the block inputs to the LUT inputs. This feature is later exploited in the *packing* stage, where a chain of LUTs in a boolean network can be packed together into a logic block. The internal routing circuitry allows such LUTs to avoid routing through switch matrices, reducing the delay between these LUTs.

1.2.2 Switch Matrix

The regular grid of routing segments is connected at the grid-points by a collection of switches known as a *switch matrix.* A switch matrix is a set of programmable switches between the metal channels that can be configured to create any desired routing pattern. A switch matrix connects four metal channels, an X-channel that connects to the left of the switch matrix, an X-channel that connects from the right, a Y channel that connects from above and a Y channel that connects to the bottom of the switch matrix. Each of these channels usually has the same number of tracks W, which is known as the *channel width* of the FPGA. Therefore each side of the switch matrix has W pins that have to be connected to each other in some routing pattern. The *switch flexibility* F_s of a pin on some side of a switch matrix is the number its connections to pins on the remaining sides of the switch matrix. Figure 1-3 shows a switch matrix where the pin East_1 has $F_s = 4$, North_1 has $F_s = 3$ and West_1 and South_1 have $F_s = 2$.

A popular switch matrix architecture is the *disjoint* switch architecture which is shown in Figure 1-5. In Figure 1-5, it can be seen that when the pins on a side of the switch matrix are numbered from 1 to W, a pin numbered i on one side of the SM connects *only* to the pins numbered i on the other three sides of the switch matrix. Hence the constraint on the routing pattern is that no pin i on one side can connect to a pin j on another side for $i \neq j$. Another example of a switch architecture is the *universal* switch block where every set of nets satisfying the dimensional constraint, which is a maximum number of W tracks in each channel, is simultaneously routable through the switch matrix. The choice of the switch matrix architecture can have a significant impact on the routability of the FPGA and consequently the performance of the FPGA. The connections between the pins of the switch matrix can be made

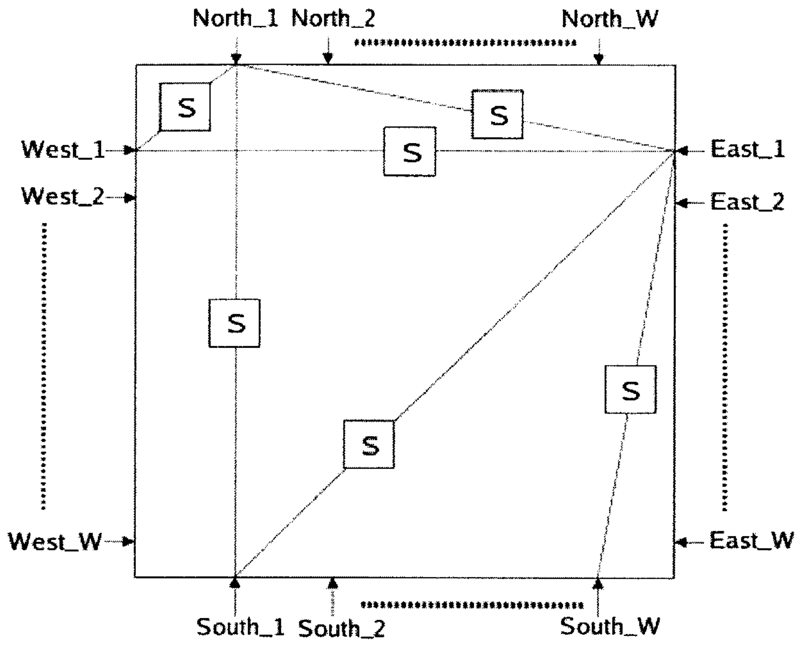


Figure 1-3: Switch block pins with different switch flexibility [14]

through a pair of tristate buffers or through a bidirectional pass transistor as shown in Figure 1-4.

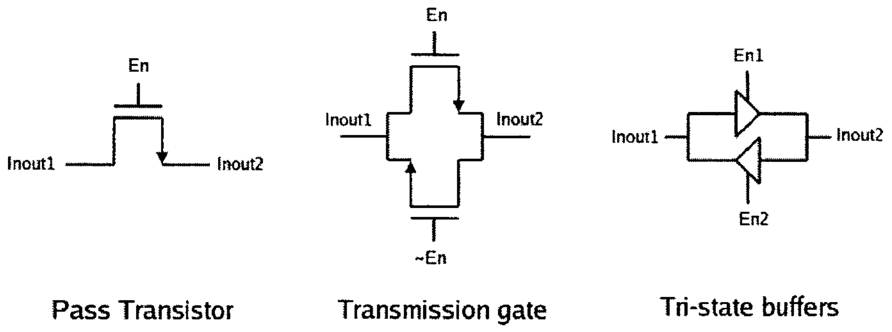


Figure 1-4: Possible options for a switch cell

Any pin in the switch matrix can be driven by at most one out of the 3 switches that connect to it. The parasitic capacitance seen by this driving switch depends on the type of the switches that connect to this pin. When the connections are made through tristate buffers, the contribution made by each bidirectional connection to the capacitance seen by the driving pin is the sum of the input capacitance C_{in} and output capacitance C_{out} of a tristate buffer. Therefore capacitance encountered by a net driver at a switch matrix pin due to the bidirectional switches alone, is given by

$$C_{pin} = 3 \times (C_{in} + C_{out})$$

This capacitance value is seen by the driving switch regardless of whether the other tristate buffer connections are open or closed. When pass transistors are used to make the connections, the state of the other connections, either open or closed, determines the capacitance seen by the driving switch. In other words, in the case of pass transistor connection, the fanout at a pin determines the parasitic capacitance seen by the switch driving the pin. The contribution of an open pass transistor switch is only the overlap capacitance value of the transistor. On the other hand, a closed transistor switch, apart from the closed transistor capacitance, exposes the entire capacitance tree behind this switch, up to the points where buffers are encountered.

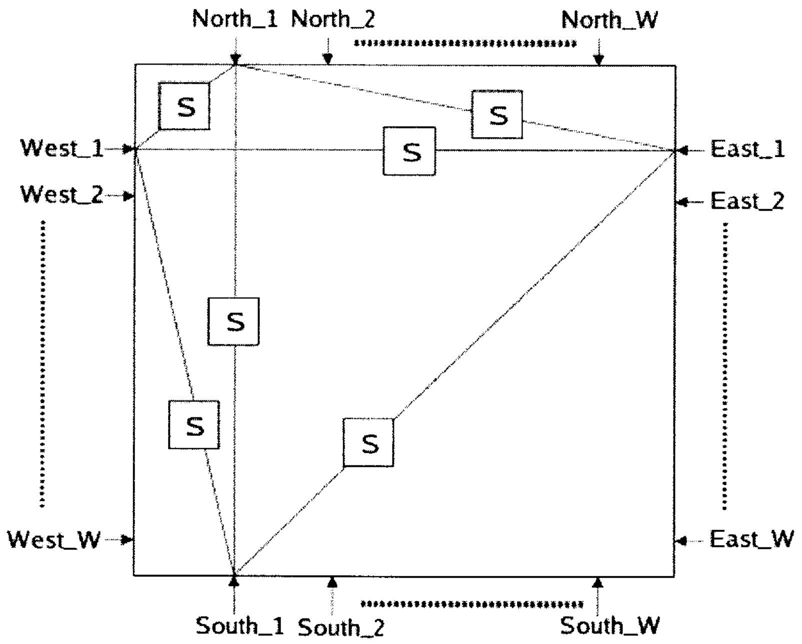


Figure 1-5: Disjoint switch block architecture

1.2.3 Channels

The metal channels that form the interconnect of the 2-D FPGA consist of two types of *channels*, namely the X and Y channels. Each channel contains wire tracks that can be individually driven by a source. The wire in each track is broken up into a multiple wire *segments* that have the same characteristics. It is found to be be more beneficial to use segments of different lengths rather than a single uniform length throughout the FPGA [26]. This allows the router connecting two CLBs in the FPGA, to choose the type of segment depending on the physical distance between the two CLBs. 'Long' wire segments that can span multiple switch matrices, bypassing intermediate switch matrices, are ideal for connecting CLBs that are far apart in the FPGA. On the other hand, to connect CLBs that are very close to each other, 'long' segments with a higher wire capacitance, result in greater delay compared to shorter segments.

A wire segment is characterized by the following properties:

- Length - The wirelength of the segment is measured in units of the distance between two adjacent switch matrices. For example, a wire segment of length 2 connects two switch matrices that are separated by a single switch matrix. A segment of length 4, connects two switch matrices in a row or column, separated by three switch matrices.

- CLB population - When a wire segment spans across N switch matrices and is connected to the corresponding CLBs in only M out of the N switch matrices, the CLB population is given by $\frac{M}{N}$.

- Switch population - For a wire segment that spans across M switch matrices and connects through switches to only M out of them, the switch population is given by $\frac{M}{N}$.

The different types of segments used in the 2-D FPGA are summarized in Table 1.1.

Thus a metal channel can be summarized by the following properties

Length	Number of spanned SMs	CLB population	Number of connected CLBs	SM population	Number of connected SMs
1	2	1.0	2	1.0	2
2	3	1.0	3	0.66	2
4	5	0.6	3	0.4	2

Table 1.1: Segment types

- Channel width, which is the number of wire tracks within the channel
- Segment types - The different types of metal wire segments that form the channel
- Segment distribution - The proportion of each of these segment types in the channel and how these segments are interleaved
- Segment staggering - The optimal orientation of the start points of each of these segment types for maximum routability

Staggering of tracks

A metal channel of an FPGA consists of W individual tracks where W is the global *channel width* of the FPGA. When laying out the tracks of a particular segment type within a channel, care must be taken not to start every track at the same grid location in the channel. To ensure greater routability, the "start points" of the segments in a channel have to be adjusted based on the length of the segment. To explain this concept further, a coordinate system must be defined for the FPGA. Let the grid point at the lower left corner in the bottom most layer of the FPGA have co-ordinates $(1, 1)$. The grid-point to its right on has coordinates $(2, 1)$ and the grid-point above it on the same layer has coordinates $(1, 2)$. An X channel has the same y-coordinate as the grid-points it passes through and a Y channel has the same x-coordinate as the grid-points it passes through. An ideal adjustment of the segments of a particular type would shift the start point of a segment of a particular track in channel $j + 1$ back by one grid-point relative to its start point in the same track in channel j.

For a X segment of length L in a channel having n tracks of this particular segment type, the segment start-point for the i^{th} track out of the n tracks, is defined as

$$segstart = (closestInteger(i * \frac{L}{n}))\%L + 1$$

where $closestInteger(x)$ is the closest integer to a given real value x. Among all the grid points spanned by this X channel, a grid point (x, y) that marks the start of such a segment has the property

$$x - (x + y - segstart + L)\%L = 0$$

Similarly, the start point of a Y segment of length L and segment start-point $segstart$ satisfies the property

$$y - (x + y - segstart + L)\%L = 0$$

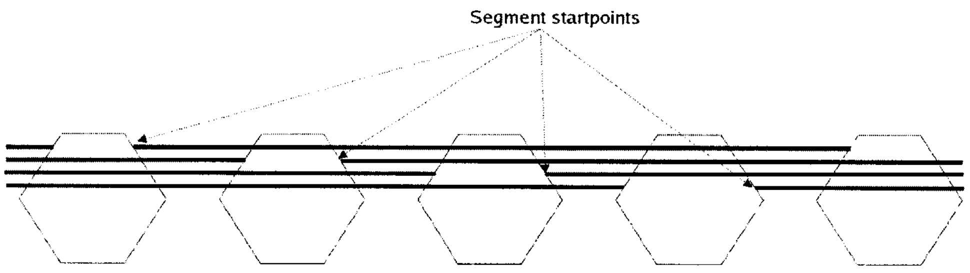


Figure 1-6: Staggering for a segment of length 4

As can be seen from the symmetry in the start point calculations, the entire interconnect grid consists of X and Y that start and end at the same locations. There are no segments in the interconnect that end at a grid-point and are unable to connect to the segments in the other dimensions. This ensure high routability that proves to be very useful when routing complex circuits.

1.3 Overview of 3-D Integrated Circuits

As mentioned before, the regularity of the FPGA interconnect is the main reason for its low performance compared to ASICS. A technique to improve the performance of the FPGA interconnect is three-dimensional integration of the transistors of the FPGA.

The motivation for 3-D circuits is that the addition of an extra dimension to the circuit can significantly decrease the *Manhattan* distance between a source and a sink. In other words, a 2-dimensional circuit can be imagined to be folded over multiple times resulting in layers of circuits that are connected by the 3-D interconnect. Due to the decreased wirelength of paths, the circuit blocks are expected to shrink towards each other resulting in higher performance of the circuit. It is shown in [13] that 31% to 56% speed improvement can be achieved with 3-D integration. Also the power dissipation of a circuit depends quadratically on the operating voltage of the circuit V_{dd} while the performance depends linearly on V_{dd}. Hence a higher circuit performance in 3-D circuits can be lowered to the performance obtained in 2-D circuits for a much lower power dissipation by lowering the V_{dd}. Several previous works in literature have argued in favor of 3-dimensional integration [33] and particularly in FPGAs, [2],[13], [15], [35], for these reasons. Several 3-D FPGA architectures have also been proposed in the literature,[3],[4], [7],[8],[14], which demonstrate higher performance and lower power dissipation than 2-D FPGAs.

1.3.1 3-D integration

In this work, the 3-dimensional layers of transistors are referred to as *active layers* and are often referred to in literature by terminology such as *device layer*, [15], *tier*, [16], and *stratum*, [17]. There are several possible methods for 3-D integration of active layers:

- Vertical Multi-Chip Module (MCM-V) - The dies are fabricated separately and bonded to a vertical Printed Circuit Board (PCB), [18] [19]. MCM-V connects the dies from their periphery to PCB and these connections have larger delay compared to the on-chip connections.

- Ultra-thin chip stacking [20] [21] and multilayer thin-film packaging, [22] - In these methods, dies are thinned and bonded. Then the inter-layer connections are made through the periphery of the dies. While these methods offer better performance than MCM-V, the delay of an inter-layer connection is still large.

- Epitaxial 3-D integration - In this method silicon seeds are used to grow more transistors on the top of the current transistors, [23]. Another similar method uses solid-phase re-crystallization where amorphous silicon is deposited on the die and laser is used for crystallization. This method is used to fabricate high-density memories [24].

- Wafer bonding - The method used in the design of the 3-D chip in [14] is wafer bonding. During the design of each active layer, 3-D wires are connected through 3-D inter-layer vias. First, during the fabrication, different dies are fabricated with conventional methods. Then, they are bonded and 3-D connections are made by etching. While this method allows high density of inter-layer connections, the delay and area overhead of inter-layer Vias are larger than on-chip Vias. Further discussion regarding the advantages and limitations of wafer bonding can be found in [14].

Chapter 2

3-D FPGAs

2.1 Architecture

This section describes in detail the architecture of the 3-D Field Programmable Gate Array. As in any FPGA, the 3-D FPGA consists of programmable logic blocks that are connected by an interconnect, which are both programmed by the configuration memory. The logic block of the 3-D FPGA is the same as a 2-D FPGA logic block but the 3-D interconnect methodology is quite different compared to the 2-D interconnect system. The key elements of the 3-D interconnect, the switch matrices and the metal channels, are described in detail in this section. The interconnect architecture described here is the first interconnect model that was conceived for the 3-D FPGA, known as the *symmetric* architecture. Later sections detail the optimizations that can be made to this architecture to achieve higher performance and overcome the problems that will be discussed at the end of this section. Since this work was primarily based on the physical parameters of the 3-D FPGA chip developed in [14], a summary of the relevant physical parameters is also presented here.

2.1.1 Global Architecture

A 3-D FPGA as shown in Figure 2-1 consists of multiple layers of logic blocks and switch matrices that are connected by vertical vias. This results in a 3-dimensional lattice of switch matrices that are connected by the metal interconnect. The 3-D interconnect consists of the horizontal X and Y channels and the vertical Z channels. Each logic block can connect to any horizontal or vertical wire segment that passes through the switch matrix at its location. The I/O pads of the FPGA are present along the periphery of the topmost layer as this layer alone is directly accessible by any off-chip source. The FPGA has an *I/O ratio* of 4 which means that four I/O pads are present in place of a single logic block in the I/O periphery.

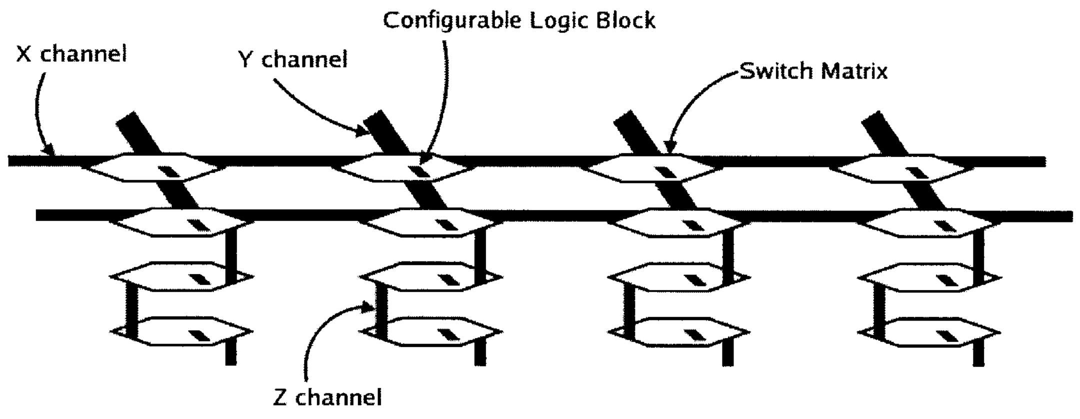


Figure 2-1: 3-D FPGA architecture

2.1.2 Symmetric interconnect architecture

The symmetric interconnect is a direct extension of the 2-D FPGA's interconnect system to 3 dimensions, with symmetric connections between the channels in all three dimensions. The vertical channel is simply an identical *third* dimension added to the 2-dimensional interconnect system. The vertical segments have the same segment properties such as number of spanned CLBs, CLB population and switch population, as the horizontal segments. However, as shown later, such a simple extension of the standard 2-D architecture to 3 dimensions has its own disadvantages. Hereafter in this work, this architecture will be referred to as the *3-D symmetric* FPGA architecture.

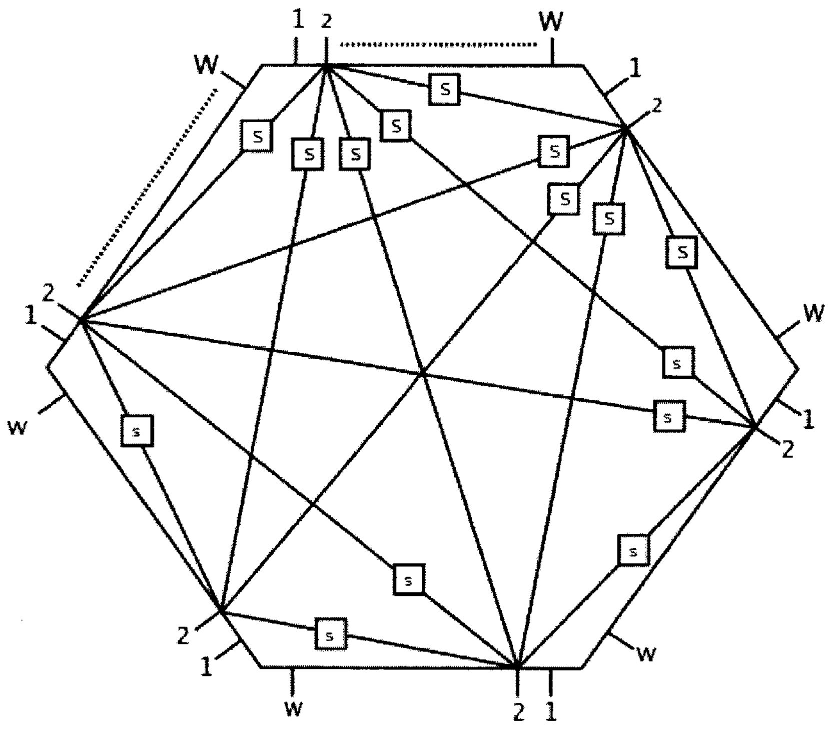


Figure 2-2: 3-D Switch Matrix [14]

Switch matrix

The switch matrix (SM) or the switch box connects the intersecting horizontal and vertical wire segments at a grid-point in the lattice. A 2-D switch matrix has four sides which connect the X and Y segments to each other. However, the 3-D switch matrix as shown in Figure 2-2 has a hexagonal geometry owing to the fact that it connects to the four horizontal segments lying in the same layer and two vertical segments that intersect at this layer. The 3-D switch matrix has W pins of each of its six sides that connect to the respective tracks of the channels, where W is the channel width. The choice of the switch matrix topology is the *disjoint* switch architecture. Thus the flexibility of this switch matrix is 5 since each pin connects to exactly 5 other pins. The connections between the pins of the switch matrix are bidirectional and are made through a pair of tristate buffers or pass transistors. Therefore, the capacitance seen at any pin in this switch matrix by a net driver is

$$C_{pin} = 5 \times (C_{in} + C_{out})$$

From the description of the 2-D FPGA's switch matrix in the previous chapter, it can be seen that the capacitance at a pin of the 3-D switch matrix is 5/3 times the capacitance seen at a pin of the 2-D switch matrix. Therefore, the additional connectivity in the 3-D switch matrix comes at the cost of increased downstream capacitance at every pin.

Interconnect channels

The segments used in the 3-D symmetric FPGA, as shown in Figure 2-3 are identical to those used by the 2-D FPGA, shown in Table 2.1. The connections between the segments and the CLBs are made through tristate buffers to isolate the capacitance due to the select logic between the switch matrix pins and the block pins.

Length	Number of spanned SMs	CLB population	Number of connected CLBs	SM population	Number of connected SMs
1	2	1.0	2	1.0	2
2	3	1.0	3	0.66	2
4	5	0.6	3	0.4	2

Table 2.1: Segment types

Staggering of tracks

The staggering of the tracks in the 3-D symmetric interconnect is also an extension of the track staggering seen in the 2-D FPGA. For a X segment of length L in a channel having n tracks of this particular segment type, the segment start-point for the i^{th} track out of the n tracks, is now defined as

$$segstart = (closestInteger(i * \frac{L}{n}))\%L + 1$$

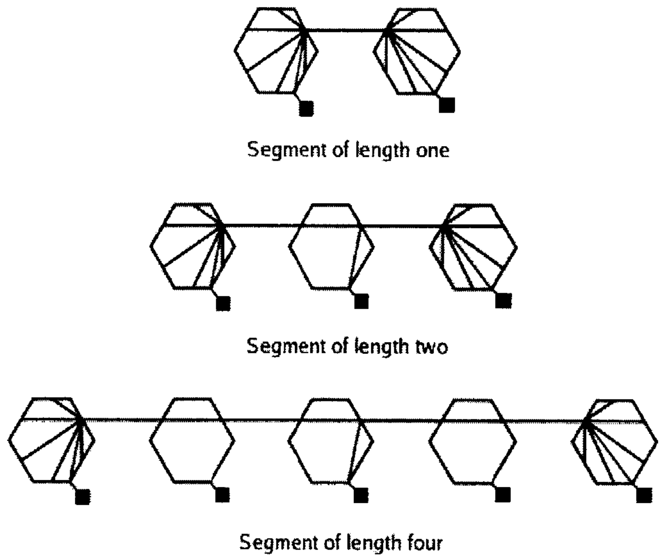


Figure 2-3: Different types of segments used in the 3-D FPGA

where $closestInteger(x)$ is the closest integer to a given real value x. The grid point (x, y, z) that marks the start of this segment has the property

$$x - (x + y + z - segstart + L)\%L = 0$$

Similarly, the start point of a Y segment of length L and segment start-point $segstart$ satisfies the property

$$y - (x + y + z - segstart + L)\%L = 0$$

while a Z segment of length L and segment startpoint $segstart$ starts at grid-points satisfying

$$z - (x + y + z - segstart + L)\%L = 0$$

Figure 2-4 shows the staggering in the 3-D interconnect for four tracks that are of the same segment type having a length of 4.

2.2 Physical parameters

The 4mm x 4mm 3-D FPGA chip with 3 active layers was designed in the Lincoln Lab 3-D FDSOI process. The hardware design of the chip and the detailed circuit design of the chip can be found in [14]. This section presents only the physical parameters and architectural features that are necessary for the design of the 3-D FPGA CAD tool and further architectural exploration.

The physical parameters used by the CAD tool for timing calculations can also be found in [14]. From [14], it can be seen that the length of the vertical Z channel and consequently its segment capacitance is much smaller than that of the horizontal channels. More importantly, the output resistance of a tri-state buffer or the equivalent resistance of a pass transistor is nearly 7 times the resistance of a wire segment that spans a single tile. The timing calculations for the 3-D architecture shown below reflect the effect of such high switch resistance to segment resistance ratio.

2.3 Delay analysis

While the 3-dimensional nature of the interconnect provides greater routability, it comes at a cost of increased parasitic capacitance for each segment. In a 2-D switch matrix, a driving switch of a wire segment sees only the capacitance resulting from three other connections in every switch matrix that the segment connects to. However, in switch matrix of the symmetric architecture, the driving switch of a segment sees the capacitance due to five other connections in every connected switch matrix. The

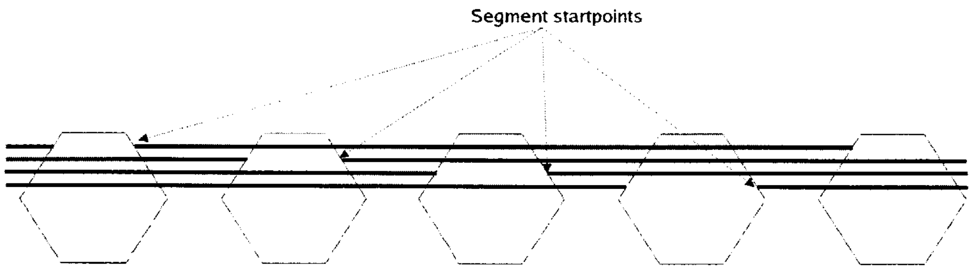


Figure 2-4: Staggering for a segment of length 4

Resistance-Capacitance model equations for each of the segments in Table 1.1 is constructed below in Table 2.2.

Table 2.2: Comparing segment delays in 2-D and 3-D

Segment Description	*Delay from RC model*	*Delay in picoseconds*
Length 1 in 2-D	$0.69 \times (R_{sw}(7C_{in} + 6C_{out} + C_{seg}) + R_{seg}(4C_{in} + 3C_{out} + 0.5C_{seg}))$	95.2
Length 1 in 3-D	$0.69 \times (R_{sw}(11C_{in} + 10C_{out} + C_{seg}) + R_{seg}(6C_{in} + 5C_{out} + 0.5C_{seg}))$	139
Length 2 in 2-D	$0.69 \times (R_{sw}(8C_{in} + 6C_{out} + 2C_{seg}) + R_{seg}(9C_{in} + 6C_{out} + 2C_{seg}))$	165.6
Length 2 in 3-D	$0.69 \times (R_{sw}(12C_{in} + 10C_{out} + 2C_{seg}) + R_{seg}(13C_{in} + 10C_{out} + 2C_{seg}))$	223.9
Length 4 in 2-D	$0.69 \times (R_{sw}(8C_{in} + 6C_{out} + 4C_{seg}) + R_{seg}(18C_{in} + 12C_{out} + 8C_{seg}))$	346.2
Length 4 in 3-D	$0.69 \times (R_{sw}(12C_{in} + 10C_{out} + 4C_{seg}) + R_{seg}(26C_{in} + 20C_{out} + 8C_{seg}))$	433.6

The Elmore delay model is used to calculate the delay for a signal to traverse a given wire segment. R_{sw} is the output resistance of the tristate buffer, C_{in} and C_{out} are the input and output capacitances of the tristate buffer, R_{seg} and C_{seg} are the resistance and capacitance of a wire segment of unit length. An example of the Elmore delay calculation is shown in Figure 2-5 for a segment of length 2 in the 3-D interconnect.

The physical values from [14] are substituted in the RC equations shown in Table 2.2, which yield the delay values for each segment type. As seen from Table 2.2, a 3-D horizontal segment of the same length, switch and logic block populations has more delay than a 2-D segment with similar characteristics. This observation indicates the need for more optimization in the interconnect architecture in order to achieve high performance in 3-D FPGAs.

As a consequence of the segments in the symmetric 3-D FPGA having more delay than their 2-D counterparts, the delay of the point-to-point connections in the FPGA also increases. Figure 2-6 shows the average delay of the point-to-point connections of both a 2-D FPGA and a 3-D FPGA with 5 layers, in the increasing order of their

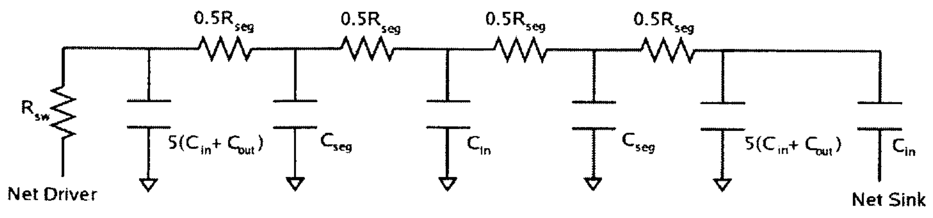


Figure 2-5: RC model of segment of length 2 in 3D

$$\begin{aligned} T_{del} &= 0.69 \times (R_{sw} \times 5 \times (C_{in} + C_{out}) + (R_{sw} + 0.5 \times R_{seg}) \times C_{seg} + \\ &\quad (R_{sw} + R_{seg}) \times C_{in} + (R_{sw} + 1.5 \times R_{seg}) \times C_{seg} + \\ &\quad (R_{sw} + 2 \times R_{seg}) \times 5 \times (C_{in} + C_{out}) + (R_{sw} + 2 \times R_{seg}) \times C_{in}) \\ &= 0.69 \times (R_{sw} \times (12 \times C_{in} + 10 \times C_{out} + 2 \times C_{seg}) + \\ &\quad R_{seg} \times (13 \times C_{in} + 10 \times C_{out} + 2 \times C_{seg})) \end{aligned}$$

wirelength. It can be seen from Figure 2-6 that the average delay of 3-D connection is considerably greater than a 2-D connection of the same wirelength. Therefore, a circuit will run faster on the symmetric 3-D FPGA only when its critical path has a much lower wirelength than the circuit's critical path in a 2-D FPGA. of the circuit in a 2-D FPGA. Though the wirelength of each connection is expected to be lower in the 3-D FPGA, the higher delay of the 3-D segments is a strong mitigating factor in the performance of the symmetric 3-D architecture.

In the 3-D FPGA, circuit blocks are expected to come much closer spatially to each other than in the 2-D FPGA, due to extra *degree of freedom* in connecting the blocks. In some sense, the 2-D circuit can be thought of as being *folded over* in a 3-D FPGA over multiple layers. Since the main advantage in the symmetric 3-D FPGA is the decrease in wirelength, it is necessary to study the relation between the connection wirelengths and the number of layers in the FPGA. Using our 3-D CAD tool which is detailed in the next chapter, the circuits from the MCNC'91 benchmark suite were mapped to 3-D symmetric FPGAs of sufficient dimensions. From the placed and routed circuit, the wirelengths of all their point-to-point connections, in units of the inter-CLB distance, were extracted. Figure 2-7 shows the individual and combined histograms of the wirelengths of the connections in the circuit *frisc* in the 2-D and

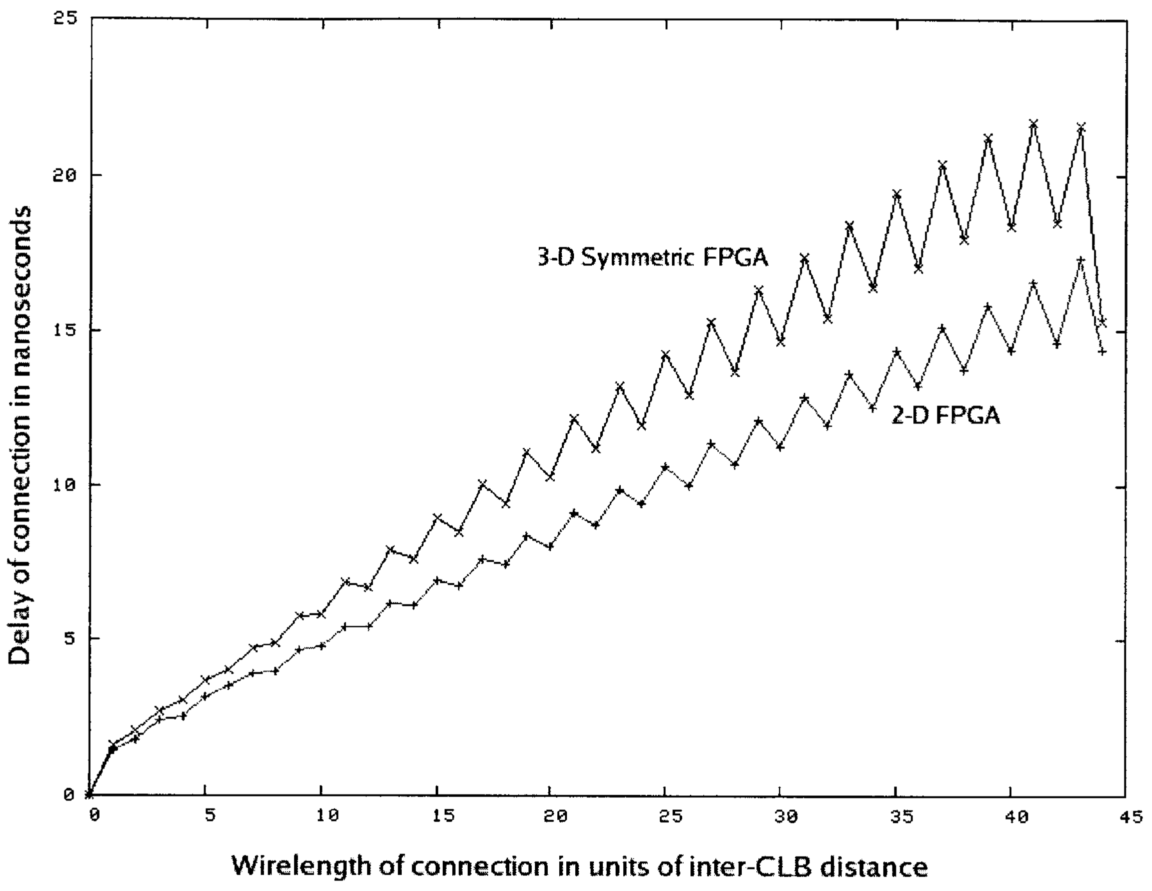
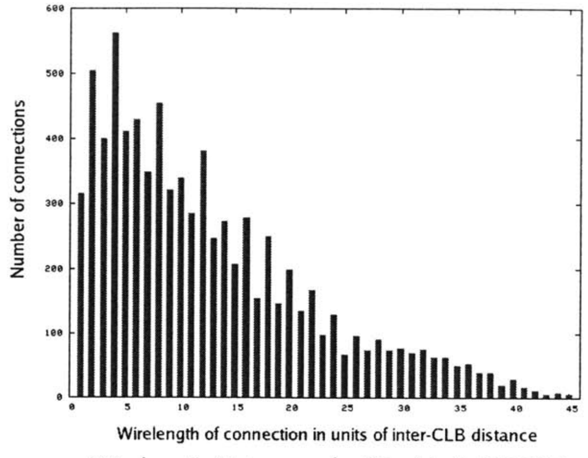


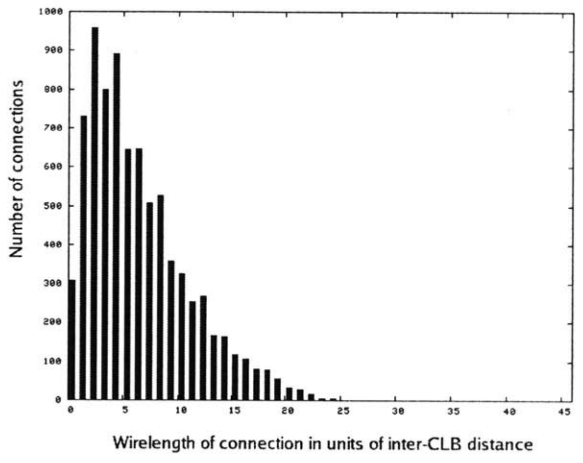
Figure 2-6: Average delay of point-to-point connections

the 3-D symmetric FPGA. As can be seen from Figure 2-7, 3-D integration certainly reduces the wirelength of the connections in the mapped circuit compared to the 2-D FPGA.

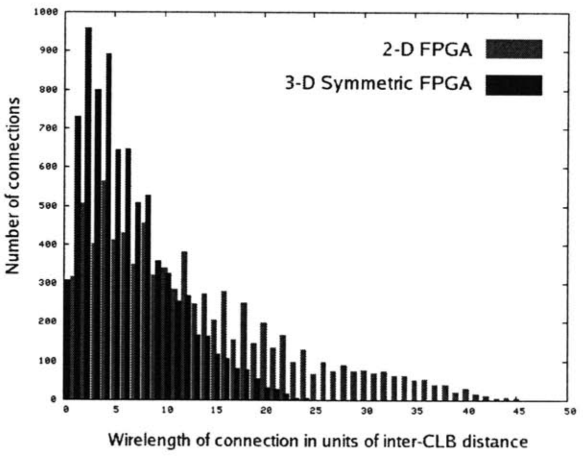
However, a 3-D FPGA must be able to achieve the same reduction in the delay histogram as well in order to guarantee higher performance than the 2-D FPGA. Figure 2-8 shows the individual and combined delay histograms for the circuit *des* mapped to both a 2-D and a 3-D symmetric FPGA. The delay intervals in the delay histograms are of 0.5 nanoseconds. It can be seen from Figure 2-8 that the delays of the point-to-point connections are also reduced in the 3-D symmetric architecture, but not as significantly as the wirelength distribution.



Wirelength histogram for 'frisc' in 2-D FPGA



Wirelength histogram for 'frisc' in 3-D symmetric FPGA



Combined wirelength histogram for 'frisc'

Figure 2-7: Wirelength histograms for circuit *frisc*

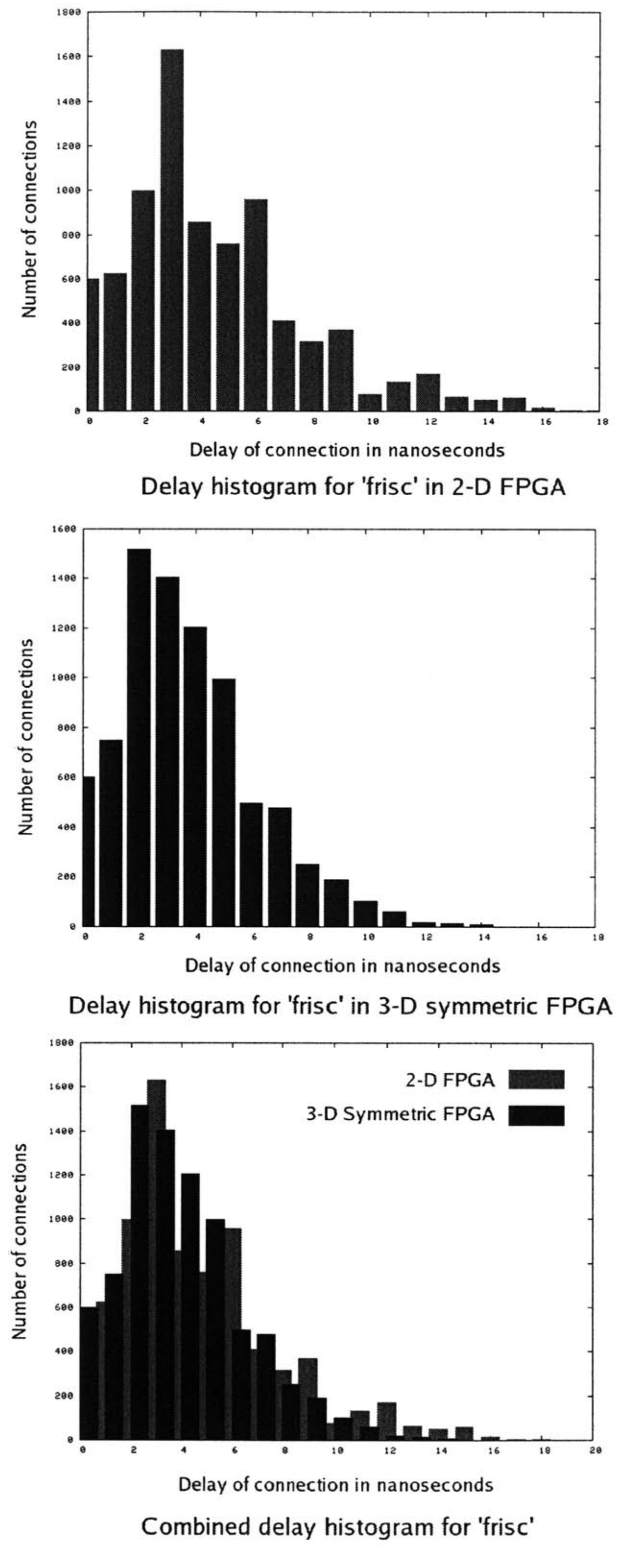


Figure 2-8: Delay histograms for circuit *frisc*

Chapter 3

CAD for 2-D and 3-D FPGAs

A CAD tool for an FPGA accepts an RTL description of a circuit and an architectural description of the FPGA for generating a configuration bitstream that can be loaded onto the FPGA. This complex problem of mapping a circuit to an FPGA is broken down into a sequence of subproblems or stages as shown in Figure 3-1. Each of these problems can be reduced to some classical theoretical problems that have been extensively researched. Each stage has its own optimization goal which is related to the ultimate goal of satisfying space and performance requirements for the circuit when mapped to the FPGA.

Several CAD tools have been proposed earlier for both 2-D FPGAs [26] and more recently for 3-D FPGAs, [5],[6], [13]. All these CAD tools have a common tool flow which will be detailed in this chapter. This work also proposes a complete CAD tool for 3-dimensional FPGAs by extending a popular 2-D FPGA CAD tool known as the Versatile Place and Route (VPR) CAD tool [26]. The objective of this CAD tool is to facilitate the exploration of the 3-D FPGA architecture which is a relatively new architecture compared to the 2-D FPGA.

The first part of this chapter provides an introduction to the standard 2-D FPGA CAD tool flow and the second part of this chapter describes the contributions of this work in the design of a 3-D FPGA CAD tool. Many of the stages in the FPGA tool flow are independent of the architecture and are therefore identical in both the 2-D and the 3-D FPGA CAD tool. Only one important stage of the VPR tool, known as the *placement* stage and a graphical representation of the circuit used in the tool, known as the *Routing-Resource* graph, are modified to convert it into a 3-D FPGA CAD tool. The CAD stages preceding the placement stage and the routing stage that follows the placement, are completely retained from the original tool.

The following section describes the standard FPGA CAD tool flow that is applicable to both 2-D and 3-D FPGAs. The later section describes in detail, the modifications proposed by this work to the placement stage.

3.1 Generic FPGA CAD tool flow

3.1.1 Logic synthesis

The first stage known as *synthesis* converts the high-level circuit description into a gate-level netlist. A logic optimization process is used to remove the redundant logic from the netlist and simplify the logic whenever possible. The gate-level netlist is then converted into a network of Look-Up Tables (LUTs) with the objective of minimizing the number of LUTs or the expected performance of the circuit. This problem of converting a gate-level netlist to an LUT-level netlist is called technology mapping and has been extensively studied in literature, [30] [31]. This work uses the Berkeley MVSIS tool for the purpose of synthesizing a network of 4-input LUTs from an RTL level circuit specified in the Berkeley Logic Interchange Format (BLIF).

3.1.2 Packing

Every CLB in the 3-D FPGA contains 4 LUTs as described in the previous section. The LUTs in the boolean network obtained from the logic synthesis stage need to be clustered into groups of 4 LUTs, to be mapped to the CLBs of the FPGA. The optimization goal in this stage is to pack lookup tables sharing common signals such as inputs and outputs, into a single logic block. When a logic block containing LUTs with shared signals is mapped to the architecture, fewer nets need to be routed to the CLB thereby reducing congestion.This problem known as *packing* is a clustering problem or

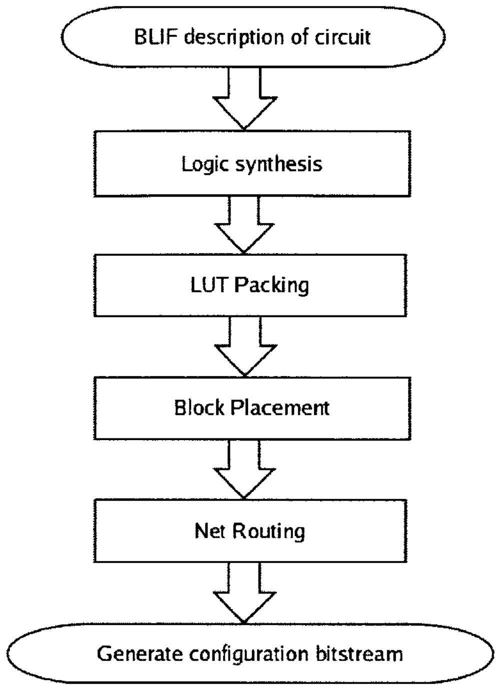


Figure 3-1: Standard CAD tool flow for FPGAs [26]

equivalently a graph partitioning problem with constraints on the maximum partition size. The packing problem also has several good solutions in the literature, [30], [31]. The VPR tool itself contains a timing-driven packing tool known as TV-Pack that packs an LUT netlist into logic blocks of a specified size. The TV-Pack tool assumes no knowledge of the underlying architecture except the capacity of the logic block and is therefore used without any modification for the 3-D FPGA.

3.1.3 Placement

The third stage of the design flow, known as the placement problem, is the first stage in the tool flow that is dependent on the architecture of the FPGA. Since this stage alone will be modified for 3-D FPGAs, this stage is described in more detail. The placement problem requires the mapping of each logic block produced by the packing stage to a specific CLB location in the 3-dimensional grid. The goal in this stage is to find a mapping of the logic blocks to CLBs that minimizes the delay of the *critical path* in the mapped circuit.

In the case of a sequential circuit being mapped to the FPGA, the critical path is the path with the maximum delay between any two flip-flops in the FPGA. Such a path may traverse through several switch matrices and through several combinational logic blocks before reaching its destination. In this case, the critical path delay can be further divided into the *net delay* and the *logic delay.* The *net delay* is the sum of the delays of all the CLB-to-CLB connections on the path, while the *logic delay* is the sum of the delays incurred in passing through the internal logic of all the intermediate CLBs. In the case of a combinational circuit being mapped to the FPGA, the *critical path* is the path with the maximum delay between an input and output pad of the FPGA. The *critical path delay* is the minimum period of the signal used to clock the circuit and is therefore a direct measure of the performance of the circuit.

The placement of logic blocks in the FPGA can be considered as an arrangement of logic blocks as well as empty locations in the 3-dimensional grid of CLBs. It can be seen that finding the exact permutation of logic blocks in n CLB locations to yield optimum performance is a problem of complexity $O(n!)$. Hence, heuristics are employed in order to find a sub-optimal solution in a reasonable duration of time.

There are three primary approaches to solving this problem:

- Partitioning-based solutions or min-cut solutions

  Placement by recursive partitioning repeatedly divides a given circuit into subgraphs in order to minimize the objective function. With each partitioning of the circuit, the CLB grid is also partitioned. Each subgraph is then assigned to a partition of the CLB grid. This process proceeds recursively till each subgraph is assigned to a unique CLB in the grid. The total number of cuts between the subgraphs is an estimate of the total wirelength of the circuit whose minimization is an objective of the placement process. Most partitioning based techniques employ some form of the prototype iterative heuristic of Kernighan and Lin (KL) [32] which uses a pair-swap move structure. During each pass, every block is moved exactly once between two partitions. At the beginning of the pass, all blocks are free to be swapped. Iteratively, the pair of the unswapped blocks with highest gain is swapped and the cost of the new partitioning is updated. After all the blocks have been swapped, the lowest-cost partition encountered throughout the pass is restored and a new pass begins.

- Analytic solutions followed by local iterative improvements

  The placement problem can be formulated as a sequence of quadratic programming problems derived from the entire connectivity information of the circuit. The problem can then be solved through a combination of global optimization and rectangle dissection, which is based on the partitioning techniques.

- Simulated Annealing

  Simulated Annealing (SA) is a stochastic minimization technique that is based on the metallurgical annealing process used to cool molten metal very gradually to produce a metal crystal lattice with lower internal energy. The VPR CAD tool contains an SA placement algorithm for 2-D FPGAs which has been extended to the case of 3-D FPGAs. This SA placement algorithm is an application of the generic SA algorithm shown in Algorithm 1, to the problem of CLB placement in the FPGA.

The SA placement process explores the placement solution space by minimizing a chosen objective function F of the placement, through a large number of local changes. The SA process has a parameter known as the *temperature*, T, which dictates the likelihood of accepting a local change to the current placement. The SA process starts with an initial random placement of the circuit's blocks in the FPGA and an initial value of T. The SA process then makes a certain number of random swaps among the CLBs of the FPGA at the current *temperature* T, with each swap involving at least one CLB which is used in the current placement. Each swap causes a change in the chosen objective function value, which is immediately recalculated. The change in objective function value is $\Delta C = F(Placement_{New}) - F(Placement_{Old})$. The swap

Algorithm 1 Generic SA algorithm

```
1: Start with random initial state
2: $T \Leftarrow 1$
3: while $T \geq T_{cutoff}$ do
4:   for $i = 1$ to $N$ do
5:     Move to an adjacent state in the solution curve
6:     Calculate new value of objective function, $C$
7:     $\Delta C = C_{New} - C_{old}$
8:     if $\Delta C \leq 0$ then
9:       Accept the new state
10:    else
11:      Generate random value $r$, $0 \leq r \leq 1$
12:      if $e^{-\frac{\Delta C}{T}} > r$ then
13:        Accept the new state
14:      else
15:        Reject the new state
16:      end if
17:    end if
18:  end for
19:  Update $T$ based on annealing schedule
20: end while
```

is then accepted with a probability $e^{-\frac{\Delta C}{T}}$. At higher temperatures, this probability is very high which allows the SA process to accept inferior solutions and thereby explore the solution space. The temperature is then either scaled down by a pre-determined factor (*fixed annealing schedule*) or by a schedule based the current progress of the placement process (*adaptive annealing schedule*). As the temperature decreases, the probability of a swap that results in an inferior value of F becomes very low. This allows the SA process to quickly converge to a sub-optimal placement solution that is very close to the optimal placement solution. The objective function F chosen by the VPR placement process is a sum of the delays of all the connections in the circuit, weighted by the *criticality* of each connection. The criticality of a connection P between a net source and a sink in a circuit with critical path delay D, is defined as

$$Criticality(P) = \frac{Slack(P)}{D}$$

The slack of a connection is the difference between the time of arrival of a signal across the connection and the required time of the arrival of the signal at the sink, in order to propagate to the next connection in the circuit. For every connection in the critical path of the circuit, the slack would be 0 since the critical path has the greatest delay among all the paths in the circuit. Therefore the objective function F known as the timing cost function, for n connections, $C_1 \ldots Cn$ is given by

$$F = \sum_{k=1}^{n} Criticality(C_k) \times Delay(C_k)$$

An important implication of this timing cost function is that if a CLB swap changes the critical path of the circuit, then the timing cost needs to be recalculated. The new critical path delay value would have to be calculated and the criticality values of all the connections in the circuit would have to be updated. But this would considerably slow down the computation in the placement process. Hence the tool assumes that the critical path delay value is not greatly affected by a block of swaps which can all be made without updating the criticality values.

3.1.4 Routing

The last computational stage of the CAD flow is the routing stage in which all the nets of the FPGA are routed between their sources and sinks. After the logic blocks and I/O pads of the circuit have been placed in the FPGA's CLBs, the router determines which of the FPGA's switches should be turned on to make all the required pin-to-pin connections. Once again, the optimization goal is to minimize the delay of the critical path in the circuit. The VPR router uses a optimized version of the PathFinder algorithm, [26], to quickly route the connections in the circuit. As later described, this stage is retained without modifications in the 3-D FPGA tool by simply altering the abstraction of the FPGA architecture that is used by this stage. This abstraction, known as the *Routing Resource* graph, is described below.

Routing-Resource graph

The VPR router, based on the Pathfinder algorithm, operates on a graphical abstraction of the FPGA architecture known as the Routing-Resource (RR) graph. Every wire segment and I/O pin is represented by a node in this graph and directed edges between these nodes explicitly model the connectivity information of the FPGA. The bi-directional switches are represented by pairs of directed edges between the nodes. As seen in Figure 1-2, the internal routing of the CLB makes it possible for any logic block input pin to connect to any input pin of any of the 4 LUTs in the CLB. Thus, the input pins of the CLB are *logically equivalent.* Similarly, any of the LUT output pins can connect to any of the output pins of the CLB which are also logically equivalent. To expose this property of the logic block's pins to the router, a *source* and *sink* node are created in the RR graph for every CLB. The input pins of a CLB have directed edges to the sink node while directed edges exist from the source node to the output pins of the CLB. Every RR node stores additional information about its physical counterpart, such as resistance, capacitance, track or pin number and so on. Timing-driven routing can then be performed directly on the RR graph using the physical information in the nodes.

3.2 Contributions to 3-D FPGA CAD flow

3.2.1 Routing-Resource graph for 3-D FPGAs

This work extends the VPR CAD tool to 3-D FPGAs by modifying only the placement stage of the tool and by retaining all the remaining stages of the tool. The stages preceding the placement stage can be retained with no effort, but the routing stage that follows the placement does depend on the underlying FPGA architecture. However, the routing tool abstracts the FPGA architecture into a *Routing Resource* graph (RR graph) which is then used by the VPR implementation of the PathFinder algorithm. Therefore to use the VPR router without any modifications, the RR graph corresponding to the 3-D FPGA is generated. The generation of the RR graph of the 3-D FPGA is a painstaking process of converting every pin and wire segment in the FPGA into a RR node and connecting all the RR nodes to reflect the 3-D connectivity. The CAD tool proposed by this work for 3-D FPGAs can take the architectural description of any 3-D FPGA and automatically generate the corresponding RR graph. This allows the VPR router to integrate seamlessly with the modified placement stage. Figure 3-2 shows a slice of the 3-D FPGA architecture and the corresponding subgraph of the RR graph generated for the 3-D FPGA architecture. It can be seen that the additional connectivity of the 3-D interconnect is exposed directly to the router through the RR graph.

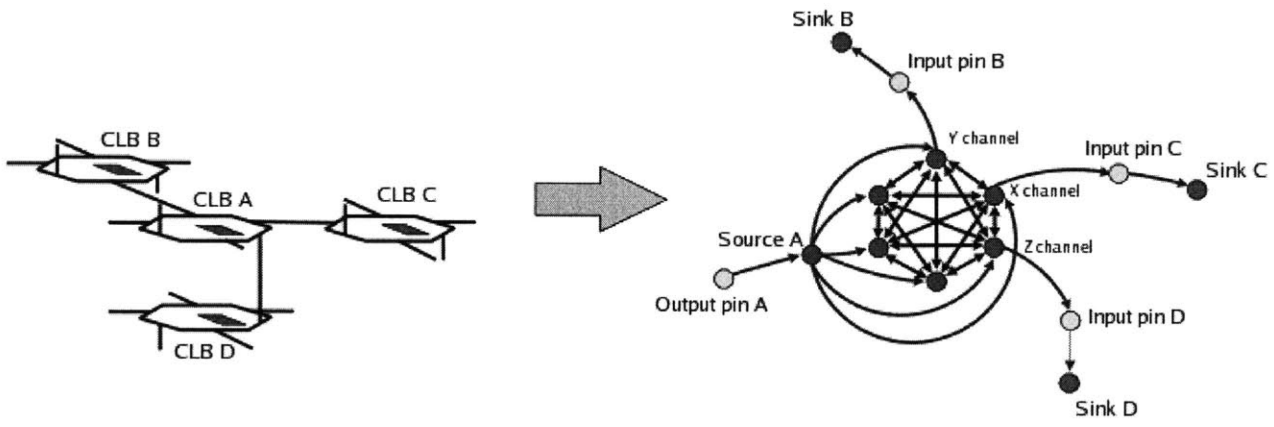


Figure 3-2: Translation of architecture to Routing-Resource graph

3.2.2 Modifications to VPR placement

In order to extend the VPR placement to the case of 3-D FPGAs, several internal data structures of the VPR code base have been modified to reflect the 3-D architecture. These structures include the CLB matrices which now represents a 3-dimensional grid of CLBs, switch matrix representations, I/O periphery information and CLB to track connectivity, among several others. Apart from these modifications which are necessary to extend the VPR placement to 3-D FPGAs, some changes have also been introduced in the SA algorithm used in the placement, which are described below.

Non-adaptive schedule

The actual implementation of the SA placement process is highly dependent on the problem space and the choice of parameters can make a significant change in the runtime of the process. These parameters include the annealing schedule, the number of swaps attempted at each temperature and the exit criterion for the process. The VPR placement for 2-D FPGAs uses an adaptive annealing schedule that adjusts itself and the initial parameters, based on the ratio of the number of successful swaps to the total number of attempted swaps at a particular temperature. The VPR placement algorithm and its parameters are designed to keep this success ratio nearly constant and equal to 0.44, [26]. However, when the 2-D placement algorithm is extended to the case of the 3-D placement, the parameters specified for the 2-D placement algorithm are not suited to maintain the success ratio is a constant. In fact, finding an ideal set of parameters to maintain a constant success ratio for the 3-D placement becomes quite a tedious process. Hence a small compromise is made in the performance of the placement algorithm implementation by choosing a simple non-adaptive SA schedule that makes a definite number of temperature steps. This allows all the parameters of the SA algorithm to be scaled linearly depending on the current temperature, initial temperature and the exit temperature.

Two-stage Simulated Annealing

The timing cost function specified in the VPR placement algorithm is intended to reduce wirelength and congestion in the final routing, since the evaluation of the critical path after every swap of CLBs is impractical. However, the placement algorithm must still maintain the global objective of reducing the critical path delay of the final circuit. In order to deal with the dual objectives of minimizing the timing cost function and the critical path delay, a two-stage Simulated Annealing process as shown in Algorithm 2 is proposed. The two-stage SA process consists of :

- A local SA process that performs small local improvements to the circuit by minimizing the timing cost function

Algorithm 2 Minimize critical path delay of a placed circuit

```
1: Start with random placement of circuit's blocks
2: $T \Leftarrow 1$
3: while $T \geq T_{cutoff}$ do
4:   for $i = 1$ to $N_{outer}$ do
5:     $oldDelay \Leftarrow$ current estimate of critical path delay
6:     $P \Leftarrow$ Snapshot of current placement
7:     for $j = 1$ to $N_{inner}$ do
8:       Select random blocks of same type and swap them
9:       Calculate new timing cost
10:      $\Delta C = New\ Timing\ Cost - Old\ Timing\ Cost$
11:      if $\Delta C \leq 0$ then
12:        Accept the swap
13:      else {Increase in timing cost}
14:        Generate random value $r$, $0 \leq r \leq 1$
15:        if $e^{-\frac{\Delta C}{T}} > r$ then
16:          Accept the swap
17:        else
18:          Reject the swap, restore blocks
19:        end if
20:      end if
21:    end for
22:    Update timing structures with state of blocks
23:    $newDelay \Leftarrow$ Critical path delay of new placement
24:    if $newDelay > oldDelay$ then
25:      Restore block state to $P$
26:    end if
27:  end for
28:  $T = T/(annealingfactor)$
29: end while
```

- A global SA process that minimizes the critical path delay by accepting or rejecting a block of modifications made by the local SA process

At a temperature T, the local SA accepts any swap of CLBs that change the timing cost by a negative value. A swap that results in a positive change of $\Delta_{timingcost}$, is accepted with a probability of $e^{-\frac{\Delta_{timingcost}}{T}}$. The global SA takes snapshots of the current placement of blocks in the FPGA, a small number of times, N_{outer} at every temperature. The number of swaps attempted between every snapshot is N_{outer}. At a temperature T, a placement snapshot x with critical path delay $d(x)$ is accepted by the global SA with a probability of $e^{-\frac{d(x)-d_{prev}}{Td_{prev}}}$ where d_{prev} is the critical path delay of the last accepted snapshot. If the snapshot is rejected, then the two-stage SA process rolls back to the previously accepted placement snapshot. The rollback consists of updating the block structures and recalculation of the delay structures as well as the timing cost function. Since the critical path delay is updated only a small number of times at every temperature, the two-stage SA process requires only a reasonable amount of computational effort. The two-stage SA ensures that at lower temperatures, CLB swaps that decrease the timing cost function but increase the critical path delay value are accepted with far lower probability.

3.3 Comparison of original and modified placement processes

The changes proposed above were to reduce the computational time of the SA process in order to obtain a placement solution that is reasonably close to the optimal solution. Figure 3-3 shows the solution curves of both the original SA placement process and the two-stage SA placement process for the circuit *ex1010* mapped to an FPGA of 5 layers. It can be seen from Figure 3-3 that the two-stage SA process converges more quickly to the proximity of the optimal solution and attains a better solution than the original SA process.

In spite of all the proposed modifications to the VPR placement algorithm, the essential nature of the Simulated Annealing technique is still maintained which allows the 3-D placement algorithm to converge to a fairly good solution. Figure 3-4 shows the visual progress of the SA placement for the circuit *bigkey* mapped to an 3D-FPGA with four layers. The circuit's blocks are indicated by the filled squares while the empty squares represent the empty CLBs of the FPGA. The algorithm starts with an initial random placement of the circuit's blocks and converges to a solution where all the circuit's blocks are compacted together in an almost symmetric shape across the four layers of the FPGA.

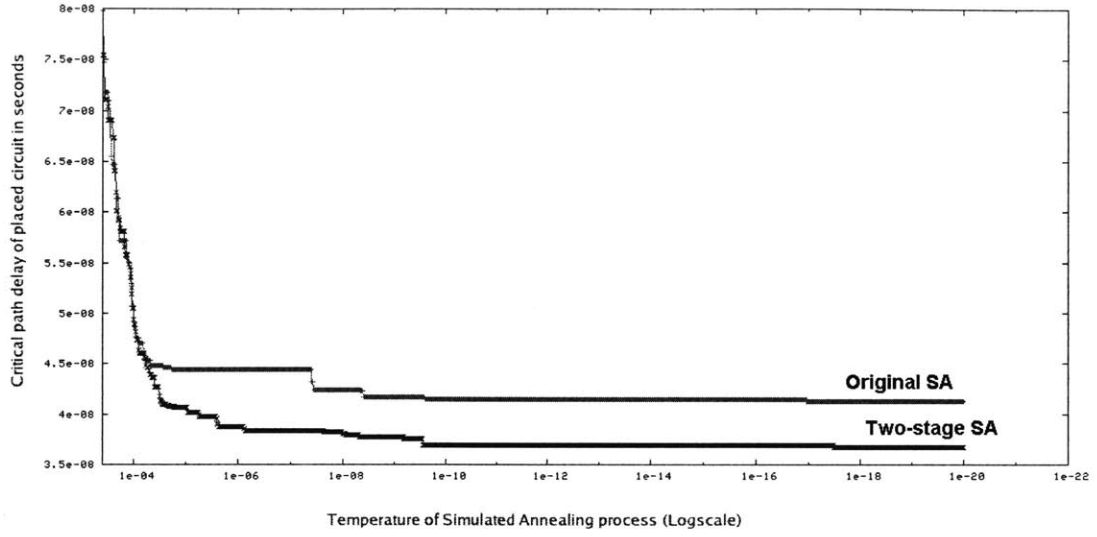


Figure 3-3: Comparison of original SA and two-stage SA processes

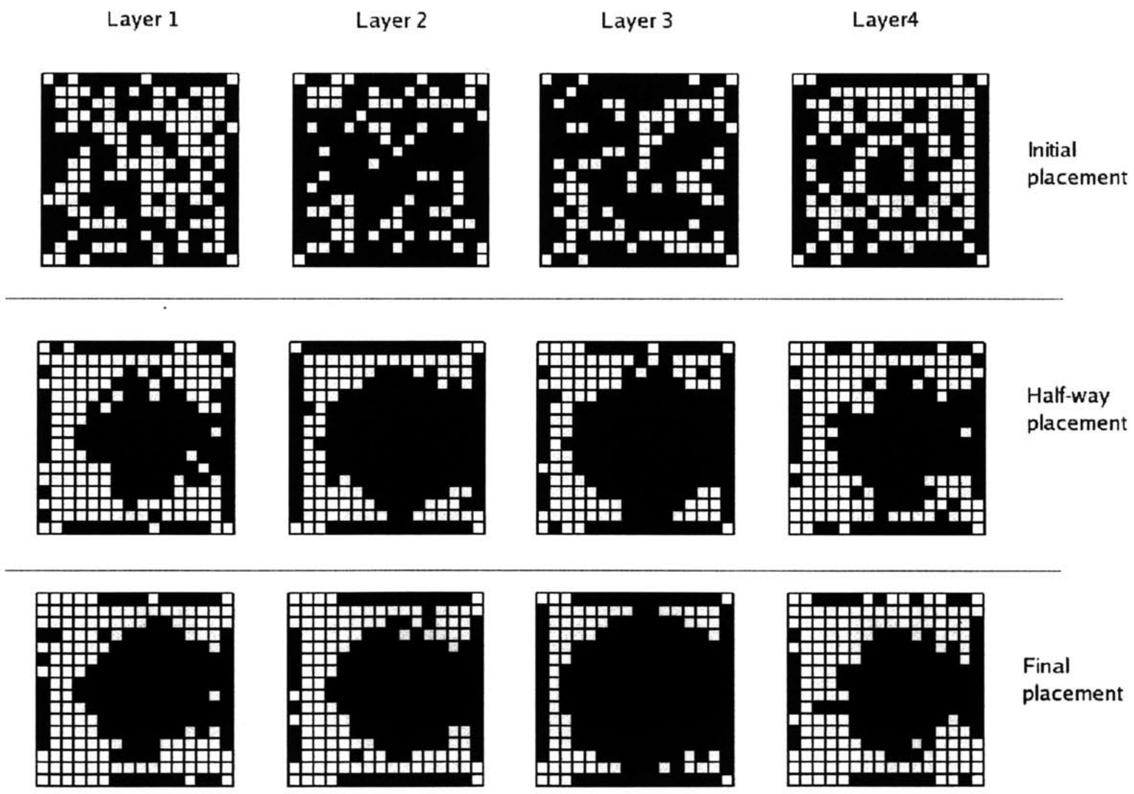


Figure 3-4: Progress of SA placement

Chapter 4

Architectural Exploration

4.1 Need for alternative 3-D interconnect

In the symmetric 3-D interconnect system described so far, every wire segment is connected in each of its switch boxes to five other wire segments of the same track. Though such an interconnect system has a high degree of connectivity, every net that is routed in the FPGA does not require such high connectivity. A simple example of such a net is one that is entirely routed on a single layer of a 3-D FPGA, with its source and all its sinks present in the same layer, as shown in Figure 4-1.

This net is a *2-dimensional* net that will never require any of the vertical channels in the FPGA. However, the horizontal segments that are available to the router, are all connected to vertical segments. The same net routed in a 2-D FPGA is shown in Figure 4-1. It can be seen that despite the same wirelength, the net will have greater delay when routed in a 3-D FPGA as compared to a 2-D FPGA. The remaining set of nets are those that span across multiple layers of the FPGA. Since these nets also use segments that are symmetrically connected, these nets will have greater delay than nets of the same wirelength in a 2-D FPGA. On the whole, nets of the same wirelength will have greater delay in the symmetric-interconnect 3-D FPGA compared to a 2-D FPGA. A chief contribution of this work is a 3-D FPGA architecture with an efficient interconnect system that has the same connectivity as the symmetric 3-D architecture.

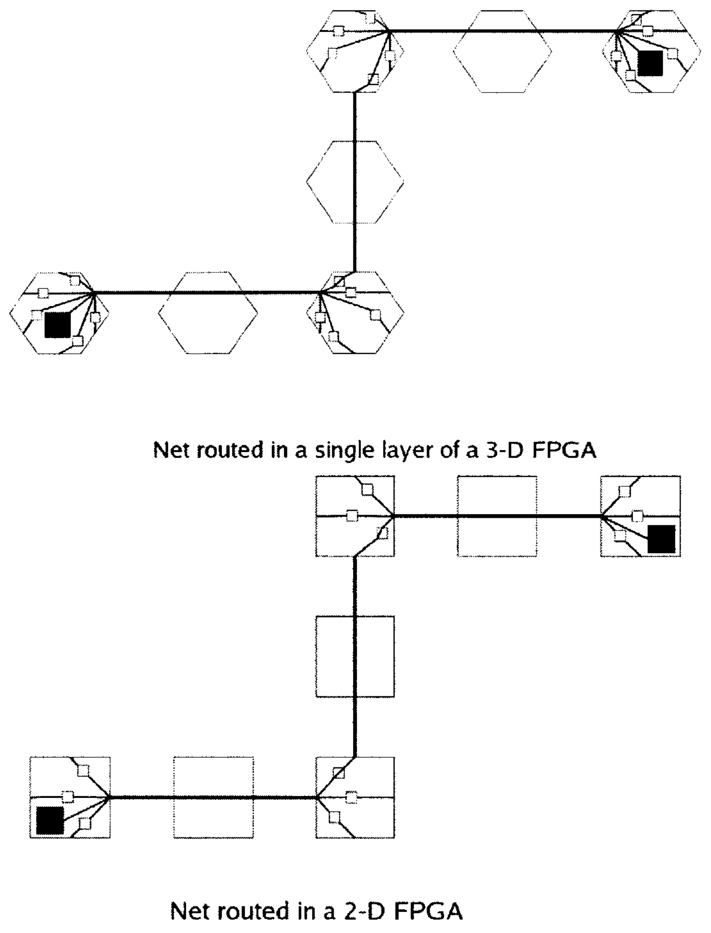


Figure 4-1: Net routed in a 2-D FPGA and in a single layer of a 3-D FPGA

4.2 Separating intra-layer and inter-layer connectivity

The nets in a circuit mapped to a 3-D FPGA can divided into two categories based on the location of their sources and sinks in the FPGA.

- Intra-layer nets - These nets, as shown in Figure 4-2 are completely routed within a single layer of the FPGA. They do not require connections to any of the vertical channels.

- Inter-layer nets - These nets are routed across multiple layers of the FPGA, as shown in Figure 4-3. These nets require vertical channel connections to traverse across the different layers of the FPGA.

In the symmetric architecture, intra-layer nets are forced to use horizontal wire segments, which have additional capacitance compared to the 2-D wire segments. However, if the interconnect contains horizontal segments that have no connections to vertical channels, a timing-driven router would utilize these segments for routing intra-layer nets. With such horizontal segments, with no connections to vertical channels, are available to the router, it can be deduced that the routing of an intra-layer net in a 3-D FPGA would be identical to the routing of the same net in a 2-D FPGA. Therefore, in every horizontal channel in the 3-D FPGA, a fraction of the wire segments should be left unconnected to the corresponding vertical segments. This allows intra-layer nets to be routed in the 3-D FPGA with no additional delay compared to the 2-D FPGA.

The remaining fraction of wire segments in a horizontal channel must have 3-D connectivity, to be used by the router to route inter-layer nets. However, these wire segments are still symmetrically connected to five other wire segments in every switch matrix that they connect to. A careful inspection of the routing of an inter-layer net shows that even an inter-layer net does not require such high connectivity. Among the various possible routes for a net in a 3-D FPGA, a *dimension-wise routing* is the only kind of routing which demands least connectivity from the interconnect system. In a *dimension-wise routing* of a net, the net is routed entirely in one of the three available dimensions to all of its sinks before proceeding to route in the next dimension. Since the distance between a source and a sink consists of three separate

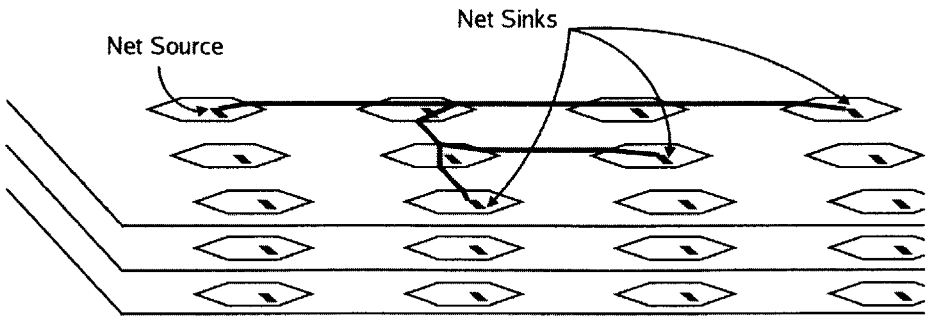


Figure 4-2: Intra-layer net

displacements in the X, Y and Z dimensions, these distances can be completely traversed one at a time. For example, an inter-layer net can be routed first only through the X segments, then only through the Z segments and finally only through the Y segments in a particular track. The inter-layer net shown in Figure 4-3 is re-routed in this manner and is shown in Figure 4-4. Such a *dimension-wise routing* requires no additional wirelength compared to any other possible routing of the same net. Therefore, if every net in the FPGA was routed in a dimension-wise manner, the resulting critical path will not have greater delay due to this manner of routing. Thus in an interconnect system, where *dimension-wise* routing is the only possible method to route inter-layer nets, inter-layer nets would have no additional delay compared to any other interconnect system. In the *dimension-wise* route of the net shown in Figure 4-4, it can be seen that the X and Y segments in a particular track only need to be connected to the segments in the Z direction in the same track and not to each other at all. Therefore an inter-layer net can be routed using X and Y segments that are connected to the corresponding Z segments, but *not to each other.* Such a X or Y segment now connects to only 3 corresponding wire segments in the same track, as compared to 5 segments in the symmetric interconnect system. The result is that the capacitance of such an X or a Y segment is the same as a segment of the same wirelength and switch population in a 2-D FPGA.

The order of routing can also be in the Y direction first, followed by the Z direction and finally by the X direction. However, the Z segment in a track still connects to

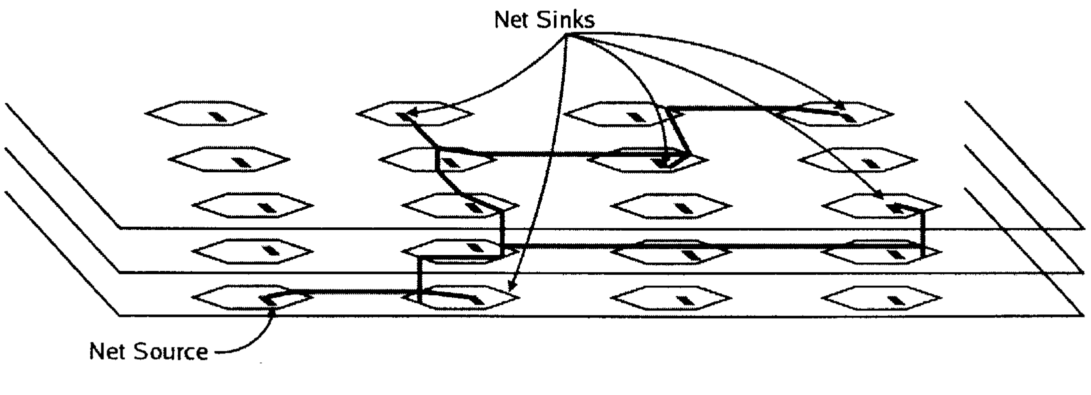


Figure 4-3: Inter-layer net

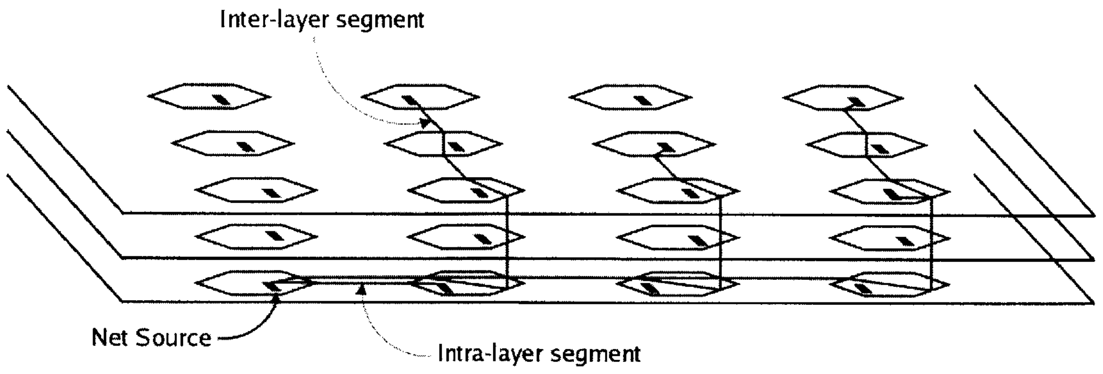


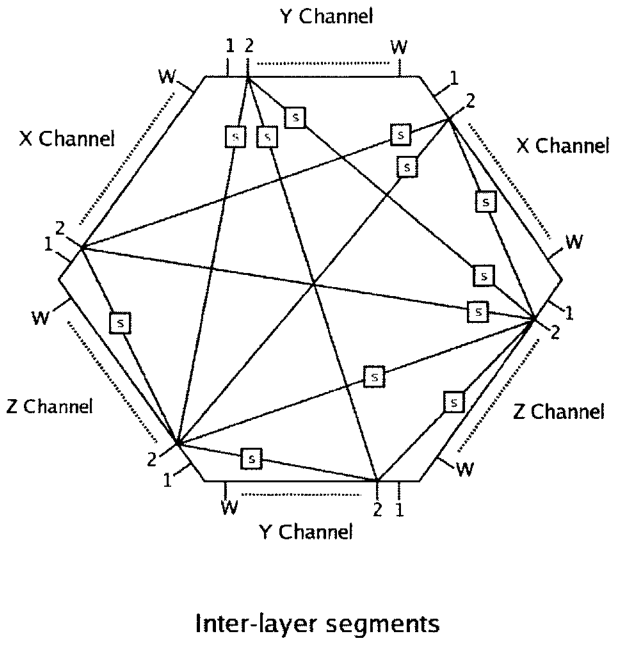
Figure 4-4: Re-routed Inter-layer net

two other segments in the same track and hence its capacitance remains the same as in the symmetric 3-D architecture. The *dimension-wise* routing can connect two 3-dimensional points in any one of the 6 possible orders of the three dimensions. However, there is a benefit in using the X-Z-Y or the Y-Z-X order compared to the other four orders of *dimension-wise* routing. The goal of the new 3-D interconnect system is to outperform a 2-D FPGA, using a small number of the layers. Therefore the wirelength of an inter-layer net in the Z dimension will be much smaller compared to the wirelength of the net in the X or Y dimensions. Therefore, it is more beneficial to reduce the capacitance of the horizontal X or Y segments rather than reducing the capacitance of the Z segment at the expense of one of the horizontal segments.

4.3 Dual interconnect 3-D FPGA architecture

In order to separate the inter-layer and intra-layer nets into two separate interconnect systems, a new 3-D FPGA architecture known as the *Dual interconnect* 3-D FPGA architecture. The interconnect consists of two sets of connected channels,

- Intra-layer interconnect - X and Y segments that are connected only to X and Y segments of the same track and not to Z segments in the same track. The Z segments in such tracks connect only to the adjacent Z segments and to CLBs.
- Inter-layer interconnect - X segments are connected only to adjacent X segments and Z segments in the same track, and not to the Y segments in the



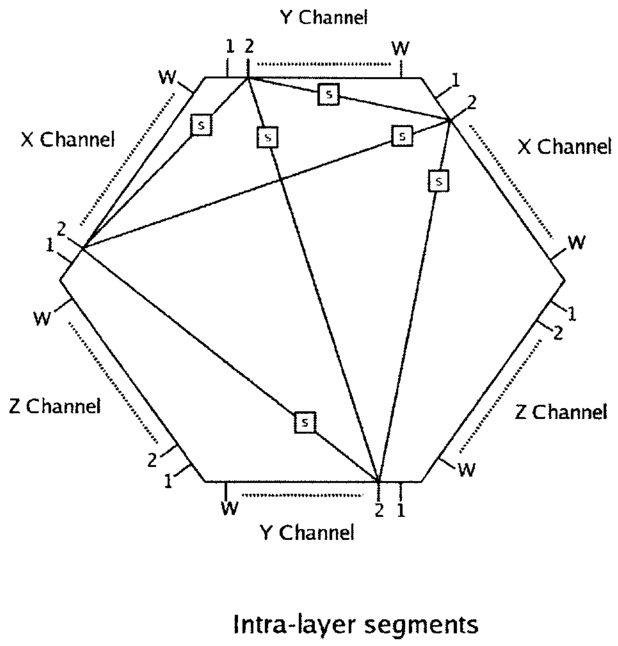


Figure 4-5: 3-D Switch Matrix connections for the two types of segments

same track. Similarly, Y segments are connected only to adjacent Y segments and Z segments in the same track, and not to the X segments in the same track. This form of connectivity forces the router to use *dimension-wise* routing to route inter-layer nets, which is shown to cause no additional delay.

Thus the switch matrix no longer has symmetric connections between its pins like the symmetric architecture in Figure 2-2. The connections in the new switch matrix for both intra-layer and inter-layer segments are shown in Figure 4-5.

This architecture does not have increased parasitics due to the additional 3-D connectivity, unlike the symmetric 3-D architecture. In fact, every point-to-point connection will have lesser delay in the dual-interconnect architecture compared to the symmetric architecture.

The delay values for all the possible connections in a $16 \times 16 \times 5$ 3-D FPGA architecture were calculated using both the symmetric 3-D interconnect architecture and the new dual-interconnect architecture. The set of 3-dimensional separations and the corresponding delays for the symmetric architecture were sorted in the increasing order of delays to produce a monotonic curve of delay values. The delays for the dual-interconnect architecture were then ordered in the same order of 3-dimensional separations as the sorted order of the symmetric architecture. This allows the easy comparison of the delay value of every point-to-point connection in the FPGA. The two delay curves are shown in Figure 4-6. From Figure 4-6, it can be seen that for any possible separation in the 3-D FPGA, the dual-interconnect system yields a path of lesser delay compared to the symmetric interconnect architecture.

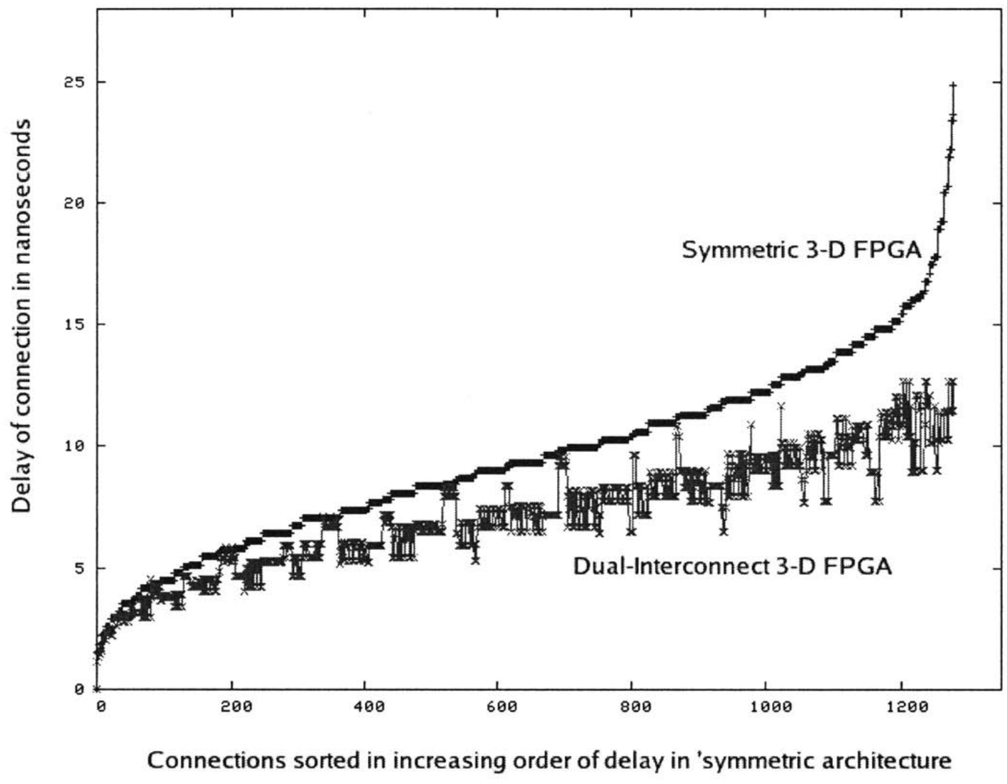


Figure 4-6: Delays of all possible point-to-point connections in a 16x16x5 FPGA

Figure 4-7 compares the average delay of a point-to-point connection in all the three architectures - the 2-D interconnect, the 3-D symmetric interconnect and the 3-D dual-system interconnect. The dual-interconnect system is seen to exhibit the lowest average delay for a connection of a certain wirelength, among all the three architectures.

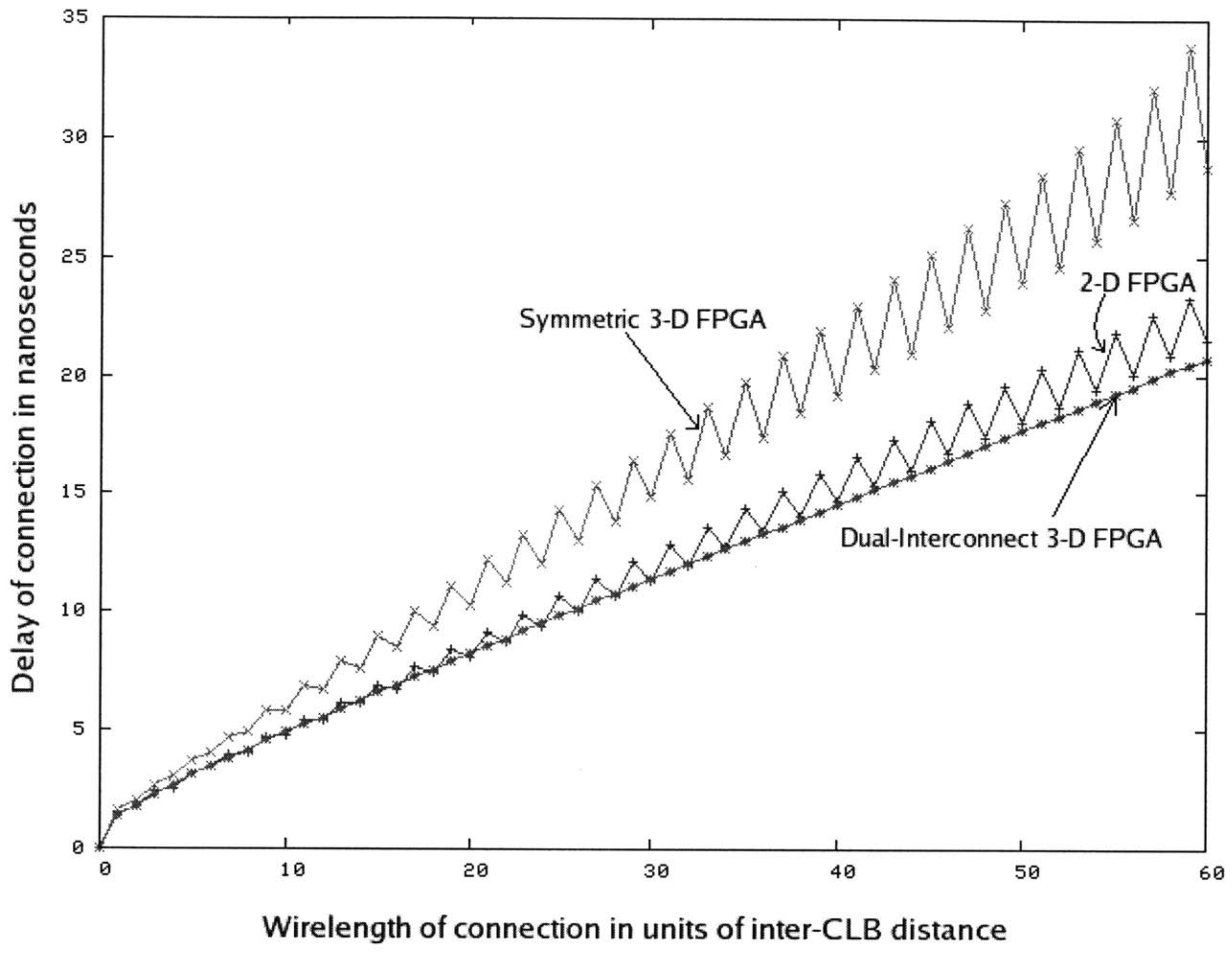


Figure 4-7: Average delay of point-to-point connections in all three architectures

4.4 Effect of I/O periphery

In the original chip designed in [14], the I/O pads are located only on the topmost layer of the FPGA. Though the location of the I/O periphery in the topmost layer is suitable for fabrication, such a restriction of the I/O block placement leads to performance issues. Several large circuits require complex routing of their primary input and output nets to their sinks that are distributed throughout the 3-dimensional grid. The effects of restricting the placement of I/O blocks are:

1. Effect on placement algorithm - Many of the CLBs that contain sinks of the I/O nets are placed closer to the I/O periphery in the top layer in order to

minimize the delay from the I/O pads to their sinks. The mapped circuit can be visualized as a mesh of CLBs that are connected to each other. As the placement algorithm accepts CLB swaps that minimize the distance between the I/O pads and CLBs containing their sinks, such CLBs are brought to the periphery of the mesh. Since the I/O blocks are located on the periphery of the top layer, the mesh is *stretched* across the top layer of the FPGA. The CLBs in the mesh that are in turn, connected to the sinks of the I/O pads get *pulled* up closer to the top layer. Therefore, fewer blocks are placed in the lower layers of the FPGA. Therefore, even though the 3-D architecture is capable of bringing the circuit's logic blocks closer to each other, the constraint of the I/O periphery does not allow the placement algorithm to do so.

2. Size of each layer - An important consequence of using only a single layer for the I/O periphery is that an increase in the number of layers does not necessarily imply a decrease in the number of CLBs in each layer of the FPGA. The length of each side of a layer of the 3-D FPGA, in units of number of CLBs, is given by

$$L = maximum(\sqrt{\frac{N_{blocks}}{N_Z}}, \frac{N_{input} + N_{output}}{4 * I/Oratio} + 1)$$

where N_{blocks} is the number of logic blocks in the circuit, N_{input} is the number of input pads, N_{output} is the number of output pads, N_Z is the number of layers of the FPGA and $I/Oratio$ is the number of I/O pads that can be physically packed into an area of the size of a CLB.

In the case of circuits that have a much larger ratio of I/O pads to logic blocks, the length of a side of each layer is determined by the number of I/O pads rather than the number of logic blocks. If the placement algorithm clusters the CLBs containing the sinks of I/O nets near the I/O periphery, then the length of the connections between such CLBs remains the same with any number of layers. If the placement algorithm attempts to cluster the CLBs together near the center of the FPGA or close to one side, some of the I/O nets will have wirelength nearly equal to a side of a single layer. Table 4.1 shows the count of logic blocks, input and output blocks as well as the minimum required horizontal size of the symmetric FPGA for each of the circuits in the MCNC'91 benchmark suite. The variation in the length of the horizontal dimension of the FPGA, with the number of layers is shown in Figure 4-8 for some selected circuits. It can be seen from Figure 4-8 that for the circuits *des* and *bigkey*, the required size of each layer of the FPGA does not change with the number of layers in the FPGA. In Table 4.1, it can be seen that these circuits have a high ratio of I/O blocks to logic blocks which necessitates the size of the I/O periphery to remain constant regardless of the number of layers in the FPGA. Therefore, a high ratio of I/O blocks to logic blocks in a circuit does not allow the size of each layer of the FPGA to decrease with an increase in the number of layers in the FPGA.

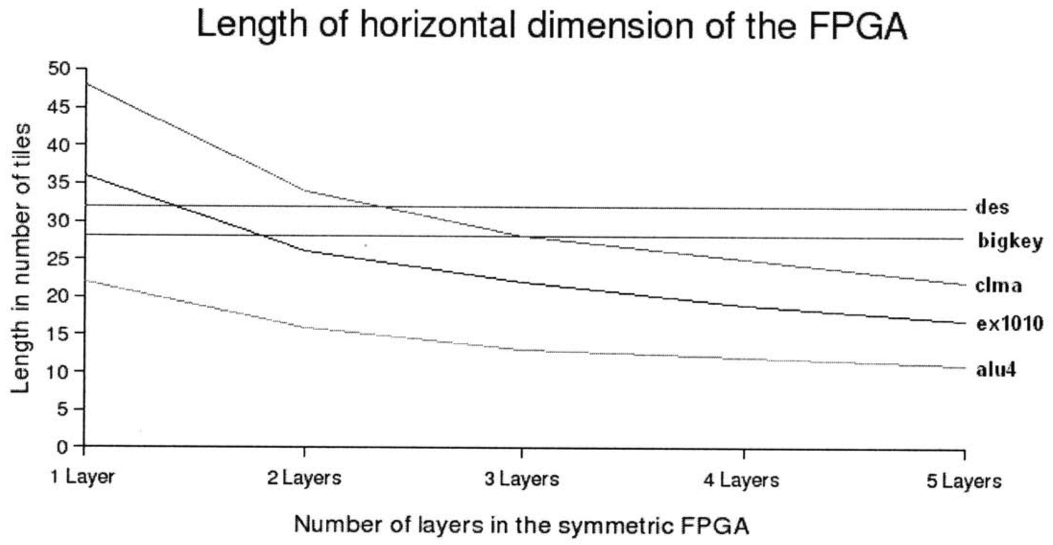


Figure 4-8: Variation of minimum horizontal dimension of 3-D FPGA with number of layers

4.5 I/O pipelining

A standard technique used in ASICs to eliminate I/O nets from the critical path is the introduction of registers between the I/O pads and the logic cells. Every primary

Circuit	N_{blocks}	N_{input}	N_{output}	$\frac{I/Oblocks}{Logicblocks}$	L_{1layer}	$L_{2layers}$	$L_{3layers}$	$L_{4layers}$	$L_{5layers}$
alu4	381	14	8	0.06	24	17	15	13	12
apex2	470	38	3	0.09	24	17	15	13	12
bigkey	427	229	197	1	28	28	28	28	28
clma	2096	62	82	0.07	48	34	28	25	22
diffeq	375	64	39	0.27	21	16	13	12	11
Des	398	256	245	1.26	32	32	32	32	32
dsip	343	229	197	1.24	28	28	28	28	28
elliptic	1146	131	114	0.21	36	26	22	19	17
ex1010	1150	10	10	0.02	36	26	22	19	17
ex5p	266	8	63	0.27	18	14	11	10	9
frisc	889	20	116	0.15	32	23	19	17	15
misex3	350	14	14	0.08	21	15	13	11	10
pdc	1144	16	40	0.05	36	26	22	19	17
s38417	1602	29	106	0.08	42	30	25	22	20
s38584	1612	38	304	0.21	42	30	25	22	22
seq	438	41	35	0.17	23	17	14	12	11
spla	923	16	46	0.07	32	23	20	17	16

Table 4.1: Minimum horizontal dimension of 3-D FPGA for MCNC'91 benchmarks

input or output net is pipelined through a series of registers that are clocked by the global clock. By inserting a sufficient number of registers in each I/O net, the delay between the I/O pads and the logic cells can be reduced to one cycle of the global clock.

This work proposes an extension of the same technique to FPGAs, called *I/O pipelining*, to eliminate the effect of I/O block placement on the circuit's performance. The registers required for pipelining the I/O of the FPGA can be found in the CLBs of the FPGA. However, due to the architecture of the CLB shown in Figure 1-2, the input of the registers in the CLB are connected *only* to the output of the LUTs in the same CLB. Therefore, if an LUT of the FPGA is used to implement the mapped circuit, the register connected to that LUT is no longer available for *I/O pipelining.* Even if the register in an LUT-register pair is left unused by the circuit, the input of the register is not available if the LUT is used to implement the circuit.

Hence, the only registers in the FPGA available for pipelining are the registers connected to any unused LUTs. Since the CLBs in 2-D and 3-D FPGAs are identical, the following procedure can be used in either architecture. A logic block in the FPGA contains 4 LUTs and connected registers and can therefore provide up to 4 pipeline stages. At the most, 4 different nets can use these pipeline stages available in a single CLB. Even a single net can use two or more of these pipeline registers since the outputs of a CLB's registers are connected to the CLB's LUTs by the CLB's internal routing. It is possible to use an empty logic block to pipeline I/O nets by the following procedure:

1. The I/O nets that have to be pipelined, are connected to the CLB's input pins. Each net is connected to the last of the 4 inputs of an unused LUT inside the CLB using the programmable internal routing. The first 3 inputs of the LUT are connected to the 'ground' value.

2. Each of these "unused" LUTs is configured to a *pass-through* state. In this configuration, the LUT logic simply passes the value of its last input to the output. Since the first three inputs of the LUT are grounded, the bitwise input to the LUT is either "0000" and "0001" when the state of the last input is "0" or "1" respectively. These SRAM locations in the LUT are set to "0" and "1" respectively in the FPGA's configuration bitstream. Thus the 16 bit configuration of the LUT in the *pass-through* state is "0000000000000010".

3. The output of the register connected to the LUT's output is connected either to the internal routing of the CLB or directly to one of the CLB's output pins.

As can inferred from the above procedure, such an I/O pipelining technique requires several unused LUTs in the FPGA. Moreover, these unused LUTs cannot be simply located on the periphery of the circuit's logic blocks. Rather, these unused LUTs must be interspersed between the circuit's LUTs to ensure that the delay between an I/O pad and all its sinks or sources is a single clock cycle. In a 2-D FPGA, if LUTs are left unused in the FPGA during the placement process for I/O pipelining, it is evident that the total area required by the circuit will be quite large. Furthermore, implementing the I/O pipelining can require considerable *intra-layer* routing resources, leading to congestion. Both these factors are quite likely to increase the critical path delay of the mapped circuit. Hence, it is not clear whether this technique will be useful at all in the case of 2-D FPGAs.

However, 3-D FPGAs by the very nature of their 3-D interconnect system offer a solution to the hindrance in implementing I/O pipelining, at the cost of an additional layer. The topmost layer of the 3-D FPGA can be left unused by the placement algorithm when placing the logic blocks of the circuit. Then, it can be seen that the unused LUTs in the topmost layer, which are on *top* of the circuit's blocks can be used for pipelining the primary inputs and outputs of the circuit. The delay between a logic block of the circuit and a pipelined register is then bounded by the length of the longest possible vertical path in the FPGA. Thus the I/O pipelining does not push the logic blocks of the circuit away from each other by occupying LUTs between them. Secondly, the only routing resources required by the I/O pipelining technique are the horizontal channels in the topmost layer of the FPGA and the vertical channels connected to this layer. Hence, I/O pipelining does not cause any additional congestion during the routing of the circuit.

The real savings in the critical path delay are seen when the placement algorithm is made *aware* of the I/O pipelining technique. As mentioned before, after the block placement, I/O pipelining reduces the delay between a logic block and an I/O pad to a single clock cycle. Hence, during the timing-driven placement, the delay value between a logic block and an I/O pad can be assumed to be always equal to a constant. When the placement algorithm swaps two blocks, which are connected to the I/O pads, the contribution to the swap's cost by the I/O nets is zero. As mentioned earlier in this section, the effect of I/O nets during the placement stage is seen in the logic blocks being spread out in the FPGA closer to the I/O periphery. Since this effect has now been curtailed, the placement algorithm can be expected to compact the logic blocks of the FPGA towards each other. This would result in a vertical core of the circuit's logic blocks in the 3-D FPGA. Figure 4-9 shows the I/O-pipelined placements for circuits, which confirm this estimate about the final placement. These placements can be compared to the placements of the same circuits shown in Figure, where the

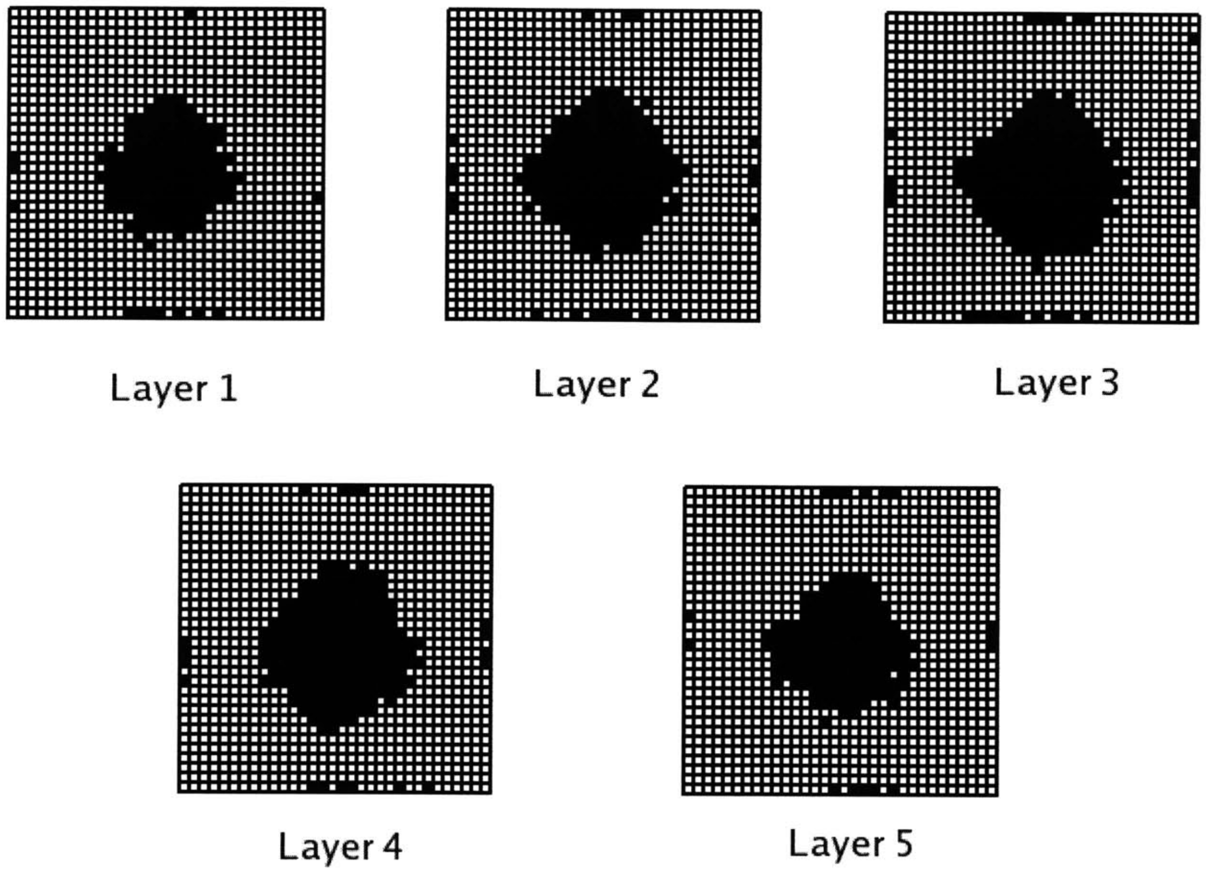


Figure 4-9: Final placement for the circuit *frisc*

I/O nets are not pipelined. The I/O pipelined placements are much more compact and it can be seen that the placement clusters in all the layers are vertically aligned above each other. The benefit of the I/O pipelining technique combined with the dual-interconnect architecture can be seen during the placement itself. Figure 4-10 shows the progress of the SA placement for the circuit *seq* mapped to the 2-D FPGA, 3-D symmetric FPGA and the Dual-Interconnect 3-D FPGA employing I/O pipelining. It can be seen from Figure 4-10 that the placement process attains significantly lower values for the critical path delay even from very early in the computation phase and ultimately converges to a superior solution.

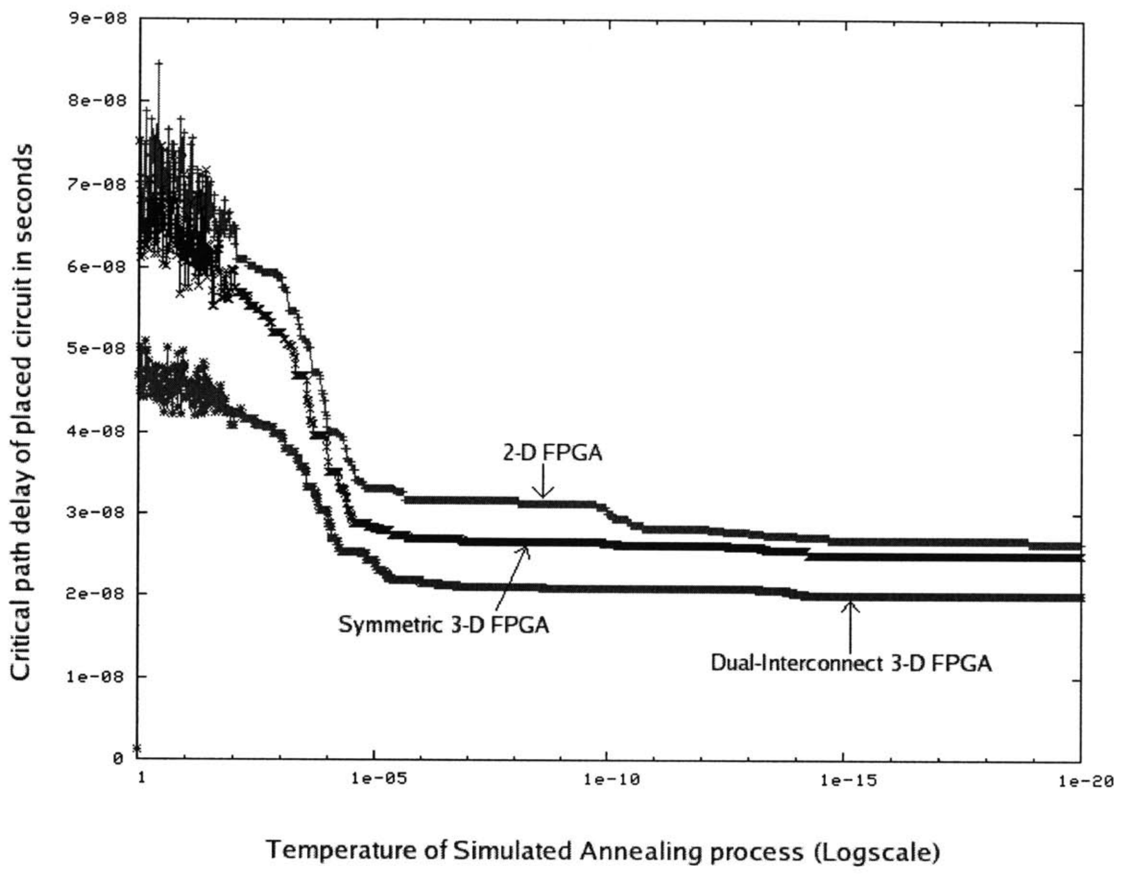


Figure 4-10: Progress of SA placement for the circuit *seq* in all three architectures

4.6 Performance of the Dual-interconnect 3-D FPGA architecture

The Dual-Interconnect architecture combined with the I/O pipelining technique, was evaluated using the MCNC'91 benchmark suite. The synthesized and packed circuits were placed and routed on FPGAs with 2,3,4 and 5 layers, containing the minimum number of CLBs. The proposed 3-D FPGA architecture and the designed CAD tool were evaluated using the MCNC'91 benchmark suite. Each of the circuits were synthesized using the Berkeley MVSIS tool, packed using the T-VPack tool from the VPR suite and were then placed and routed using the new CAD tool.

The circuits are mapped to the 2-D FPGA and the dual-interconnect 3-D FPGAs having 1,2,3,4 and 5 layers with each layer containing the minimum number of CLBs.

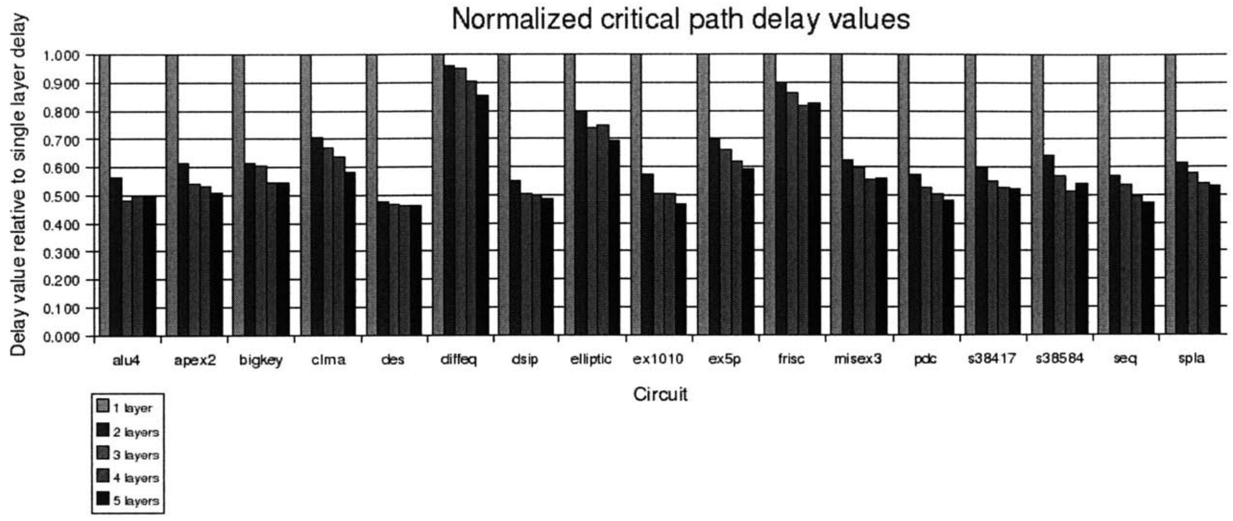


Figure 4-11: Normalized critical path delay values

Table 4.2 shows the critical path delay values (in nanoseconds) of the MCNC'91 benchmark circuits when mapped to the five FPGAs. A *monotonic decrease* in the critical path delay can be observed with increase in number of layers in the FPGA. Figure 4-11 shows the same data as in Table 4.2 in a normalized manner. Again, the delay value for every circuit mapped to an FPGA with a particular number of layers is normalized to the delay of the circuit mapped to a single layer FPGA. On an average, an FPGA with 2 layers yields a reduction of 35% in the critical path delay, an FPGA with 3 layers yields 39.2% reduction, an FPGA with 4 layers yields 41.7% reduction and an FPGA with 5 layers yields 43.4% reduction in critical path delay. The maximum reduction achieved by an FPGA with 5 layers is in the case of the *des* circuit with a reduction of 53.7%.

Figure 4-12 shows the individual and combined histograms of the wirelengths of the connections in the circuit *frisc* in the 3-D symmetric FPGA and the dual-interconnect 3-D FPGA. As can be seen from Figure 4-12, the distribution of wire-lengths in the mapped circuit nearly remains the same in both architectures. Thus the new 3-D architectures retains the ability of the symmetric architecture to reduce the wirelength of a circuit's connections.

Figure 4-13 shows the delay histograms for the circuit *des* mapped to all three

Circuit	Critical path delay in nanoseconds for circuits mapped to				
	2-D FPGA	3-D FPGA with 2 layers	3-D FPGA with 3 layers	3-D FPGA with 4 layers	3-D FPGA with 5 layers
alu4	27.8	15.6	13.4	13.9	13.9
apex2	31.0	19.0	16.7	16.5	15.8
bigkey	20.2	12.4	12.3	11.0	11.0
clma	64.9	45.7	43.3	41.4	37.7
des	27.1	12.8	12.6	12.5	12.5
diffeq	21.9	20.9	20.8	19.8	18.6
dsip	17.4	9.6	8.7	8.7	8.4
elliptic	36.9	29.4	27.3	27.7	25.7
ex1010	51.3	29.5	25.9	25.9	24.1
ex5p	27.1	18.8	17.8	16.8	16.0
frisc	44.4	39.9	38.3	36.2	36.7
misex3	24.2	15.1	14.4	13.5	13.6
pdc	48.4	27.8	25.6	24.5	23.3
s38417	47.6	28.3	26.2	25.1	24.9
s38584	33.7	21.6	19.1	17.3	18.2
seq	30.1	17.0	16.1	14.9	14.2
spla	40.9	25.2	23.6	22.0	21.7

Table 4.2: Critical path delays for MCNC'91 benchmarks mapped to 2-D FPGAs and Dual Interconnect I/O pipelined 3-D FPGAs

FPGA architectures described in this work. The combined delay histogram has a logarithmic scale on the Y-axis to clearly show the difference in the delay distributions of the three architectures. It can be seen from the combined delay histogram in Figure 4-13 that the Dual-Interconnect system significantly reduces the delays of all the circuit's point-to-point connections compared to the 3-D symmetric and the 2-D FPGA architectures.

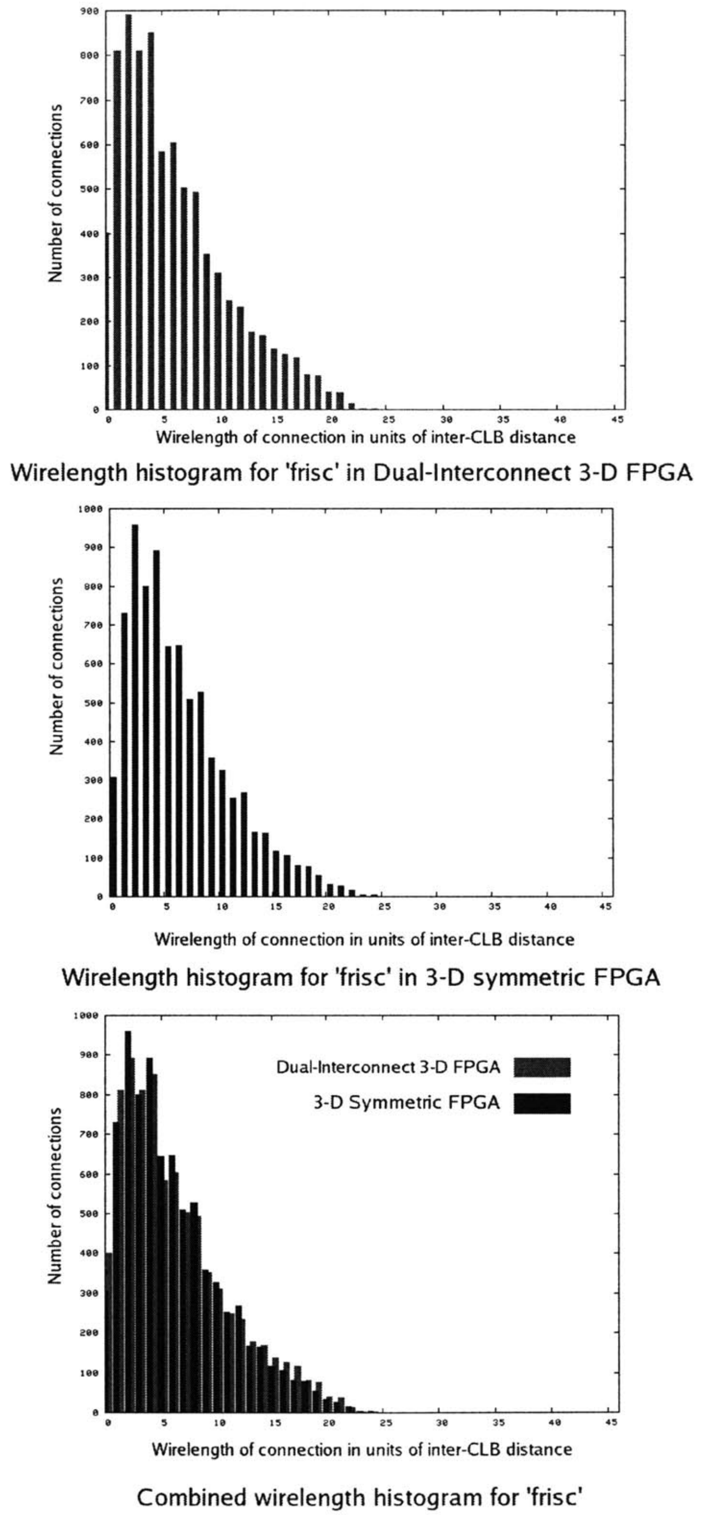
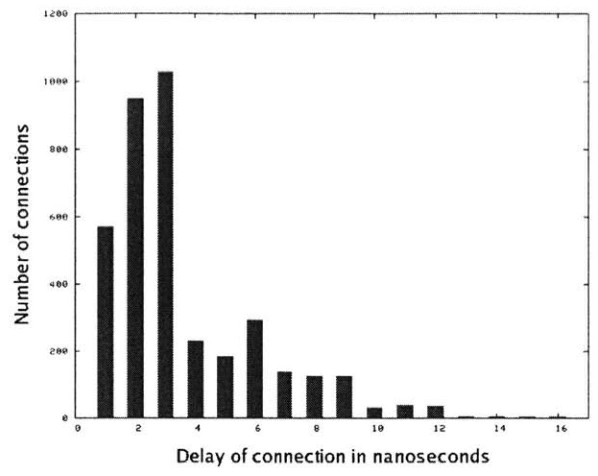
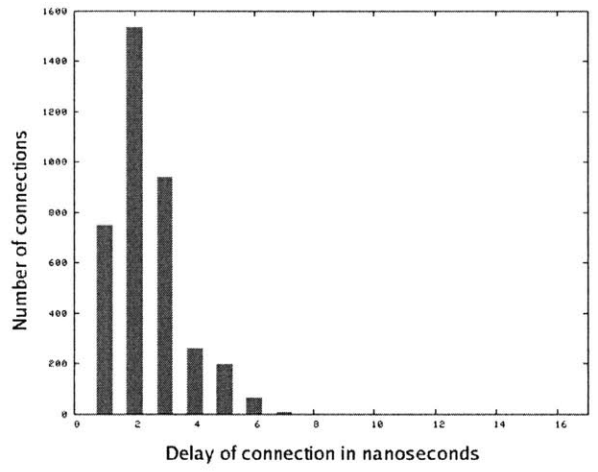


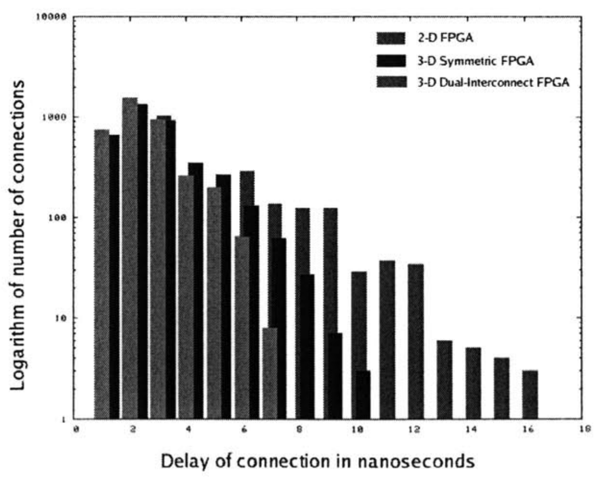
Figure 4-12: Wirelength histograms for circuit *frisc*



Delay histogram for 'des' in 2-D FPGA



Delay histogram for 'des' in 3-D dual-interconnect FPGA



Combined delay histogram for 'des'

Figure 4-13: Delay histograms for circuit *frisc*

Chapter 5

Conclusion

In this thesis, a complete architecture and a CAD tool is developed for a 3-dimensional FPGA. The CAD tool was extended from the Versatile Place and Route (VPR) CAD tool designed for 2-D FPGAs. The placement algorithm of the CAD tool was modified to directly optimize the critical path delay of the circuit through a two-stage Simulated Annealing process. A primary symmetric architecture was proposed for the 3-D FPGA that consisted of a disjoint switch matrix with symmetric connections between the horizontal and vertical channels. This architecture was shown to reduce the wirelengths of the point-to-point connections of the circuits mapped to the FPGA. However, the delays of the point-to-point connections were not significantly reduced. Furthermore, it was shown that the symmetric architecture provides additional connectivity at the cost of increased parasitics. The average delay for a connection of a *particular wirelength* was shown to be higher for a symmetric 3-D FPGA compared to a 2-D FPGA. This property was seen to mitigate the reduction in wirelength achieved by the symmetric architecture over the 2-D FPGA.

A new dual-interconnect architecture was proposed for the 3-D FPGA which was proved to have nearly the same parasitics as a 2-D FPGA and yet have the same connectivity as a 3-D FPGA. The nets in the 3-D FPGA were divided into *intra-layer* nets and *inter-layer* nets, which were then routed on separate interconnect systems. Such an architecture was shown to have lesser delay for any possible point-to-point connection compared to the symmetric FPGA. Furthermore, the average delay of the point-to-point connection was shown to be either the same or better than a 2-D FPGA. A technique known as *I/O pipelining* was proposed to counter the effects of a high ratio of I/O pads to logic blocks in the mapped circuit. This technique, to the best of my knowledge, has not been proposed so far in the literature regarding 3-dimensional FPGAs. The topmost layer of the 3-D FPGA was dedicated to the placement of the I/O blocks which were then connected to their destinations through pipelines of unused registers. The combination of the Dual-Interconnect architecture and the I/O pipelining technique enabled a 3-D FPGA with 5 layers to achieve an average of 43% delay improvement over the 2-D FPGA, and in the best case, upto 54% delay improvement.

Bibliography

[1] "Switch Box Architectures for Three-Dimensional FPGAs", *Proceedings of 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, April 2006, Pages:335-336

[2] M. Lin, A. El Gamal, S. Wong, "Performance Benefits of Monolithically Stacked 3-D FPGA", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, February 2007, Pages:216-229

[3] M. Lin, A. El Gamal, "A routing fabric for monolithically stacked 3D-FPGA", *Proceedings of the 2007 ACM/SIGDA 15th international symposium on Field programmable gate arrays*, 2007, Pages:3-12

[4] M.J. Alexander, J.P. Cohoon, J.L. Colflesh, J. Karro, G. Robins, "Three-dimensional field-programmable gate arrays", *Proceedings of the Eighth Annual IEEE International ASIC Conference and Exhibit*, Sep 1995, Pages:253-256

[5] J. Karro, J.P. Cohoon, "A Spiffy tool for the simultaneous placement and global routing for three-dimensional field-programmable gate arrays", *Proceedings of Ninth Great Lakes Symposium on VLSI*, March 1999, Pages:230-231

[6] C. Ababei, H. Mogal, K. Bazargan, "Three-dimensional place and route for FPGAs", *Proceedings of the Asia and South Pacific Design Automation Conference*, January 2005, Pages:773- 778

[7] W.M. Meleis, M. Leeser, P. Zavracky, M.M. Vai, "Architectural design of a three dimensional FPGA", *Proceedings of the Seventeenth Conference on Advanced Research in VLSI*, September 1997, Pages:256-268

[8] S.M.S.A. Chiricescu, M.M. Vai, "A three-dimensional FPGA with an integrated memory for in-application reconfiguration data", *Proceedings of the 1998 IEEE International Symposium on Circuits and Systems*, June 1998, Pages:232-235

[9] O. Ozturk, F. Wang, M. Kandemir, Y. Xie, "Optimal topology exploration for application-specific 3D architectures", *Proceedings of the 2006 conference on Asia South Pacific design automation*, 2006, Pages:390-395

[10] C. Ababei, H. Mogal, K. Bazargan, "3D FPGAs: placement, routing, and architecture evaluation", *Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays*, 2005, Page: 263

[11] , C. Ababei, H. Mogal, K. Bazargan, "Three-dimensional place and route for FPGAs", *Proceedings of the 2005 conference on Asia South Pacific design automation*, 2005, Pages:773-778

[12] S. T. Obenaus, T. H. Szymanski, "Gravity: Fast placement for 3-D VLSI", *ACM Transactions on Design Automation of Electronic Systems*, 2003, Pages:298-315

[13] S. Das, A. Chandrakasan, R. Reif, "Three-dimensional Integrated Circuits: Performance, Design Methodology, and CAD Tools", *Proceedings of IEEE Computer Society Annual Symposium*, 20-21 Feb. 2003, Pages:13 - 18

[14] Payam Lajevardi, "Design of a 3-Dimension FPGA", S.M. Thesis, Massachusetts Institute of Technology, July 2005.

[15] S. Das, A. Chandrakasan, and R. Reif. "Calibration of Rents-rule models for three-dimensional integrated circuits". *IEEE Transactions on VLSI Systems.*

[16] J. A. Burns, C. Keast, K.Warner, P.Wyatt, and D. Yost. "Fabrication of 3-dimensional integrated circuits by layer transfer of fully depleted SOI circuits". *Proceedings of MRS Symposium*, volume 768, April 2003.

[17] A. Rahman and R. Reif. "System-level performance evaluation of three-dimensional integrated circuits". *IEEE Transactions on VLSI Systems*, 2000, Pages:671-678

[18] C. W. Eichelberger. "Three-dimensional multi-chip module system". United States Patent 5,111,278, May 1992.

[19] C. Val. "3-D packaging - applications of vertical multi-chip modules (MCM-V) for micro-systems". *Proceedings of IEE/CPMT IEMT Symposium*, 1994.

[20] E. Beyne. "3D interconnection and packaging: Impending reality or still a dream?". *Proceedings of International Solid-State Circuits Conference*, February 2004, Pages:138-139

[21] E. Beyne. "Technologies for very high bandwidth electrical interconnects between next generation VLSI circuits". *Proceedings of IEDM*, Pages:23.3.1-23.3.4, 2001.

[22] J. H. Lau. "Low Cost Flip Chip Technologies: For DCA, WLCSP, and PBGA Assemblies". *McGraw-Hill*, New York, 2000.

[23] G. Roos, B. Hoefflinger, M. Schubert, and R. Zingg. "Manufacturability of 3D epitaxial-lateral-overgrowth CMOS circuits with three stacked channels". *Microelectronic Engineering*, Pages:191-194, 1991.

[24] V. Subramanian, P. Dankoski, L. Degertekin, B. T. Khuri-Yakub, and K. C. Saraswat. "Controlled two-step solid-phase crystallization for high-performance polysilicon TFTs". *IEEE Electronic Device Letters*, Pages:378-381, Aug. 1997.

[25] G. Carchon, K. Vaesen, S. Brebels, W. De Raedt, E. Beyne, and B. Nauwelaers. "Multilayer thin-film MCM-D for the integration of high-performance RF and microwave circuits". *IEEE Transactions on Components and Packaging Technologies*, 24(3):510-519, September 2001.

[26] V. Betz, J. Rose, A. Marquardt, "Architecture and CAD for Deep-Submicron FPGAs", *Kluwer Academic Publication*, 1999.

[27] Xilinx Inc., Two Flows for Partial Reconfiguration: Module Based or Difference Based, 2004. http://www.xilinx.com/bvdocs/appnotes/xapp290.pdf

[28] A. Rahman, A. Fan, R. Reif, "Comparison of key performance metrics in two and three-dimensional integrated circuits", *Proceedings of IEEE International Interconnect Technology Conference*, June 2000, Pages:18 - 20

[29] S.M. Alam, D.E. Troxel, C.V. Thompson, "A comprehensive layout methodology and layout-specific circuit analyses for three-dimensional integrated circuits", *Quality Electronic Design, 2002. Proceedings. International Symposium* , 18-21 March 2002, Pages:246 - 251

[30] J. Cong and Yuzheng Ding, "FlowMap: an optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol.13, Iss.1, Jan 1994, Pages:1-12

[31] R. Francis, J. Rose and Z. Vranesic, "Chortle-crf: fast technology mapping for lookup table-based FPGAs", Proceedings of 28th ACM/IEEE Design Automation Conference, 1991, Pages:227-233

[32] B. W. Kernighan and S. Lin, "An efficient heuristic procedure for partitioning graphs", Bell Syst. Tech. J., 1970, Vol. 49, no. 2, pp. 291-308

[33] A. Rahman, R. Reif, "System-Level Performance Evaluation of Three-Dimensional Integrated Circuits", *IEEE Transaction on Very Large Scale Integration (VLSI) Systems*, Vol. 8, No. 6, December 2000

[34] J. Burns, L. McIlrath, C. Keast, C. Lewis, A. Loomis, K. Warner, P. Wyatt, "Three-Dimensional Integrated Circuits for Low-Power, High-Bandwidth System on a Chip", *Solid-State Circuits Conference, 2001. Digest of Technical Papers. ISSCC. 2001 IEEE International* , 5-7 Feb. 2001, Pages:268 - 269, 453

[35] A. Rahman, S. Das, A. Chandrakasan, R. Reif, "Wiring Requirement and Three-Dimensional Integration Technology for Field Programmable Gate Ar-

rays", *IEEE Transaction on Very Large Scale Integration (VLSI) System*, Vol. 11, No. 1, February 2003.