ROUTABILITY PREDICTION FOR FIELD PROGRAMMABLE GATE ARRAYS WITH A ROUTING HIERARCHY

A Thesis
Presented to
The Faculty of Graduate Studies
of
The University of Guelph

by
ZHIBIN DAI

In partial fulfillment of requirements
for the degree of
Masters of Science
August, 2000

© Zhibin Dai, 2000
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.
ABSTRACT

ROUTABILITY PREDICTION FOR FIELD PROGRAMMABLE GATE ARRAYS WITH A ROUTING HIERARCHY

Zhibin Dai
University of Guelph, 2000

Advisor: Prof. D. K. Banerji

Field Programmable Gate Arrays (FPGAs) have emerged in the last fifteen years as a key technology for implementing digital circuits in VLSI. Much research has been done on their architecture and applications. One particularly important area of study is routing implementation, which is greatly affected by the routing architecture and routing resources.

This thesis explores the effective utilization of a routing hierarchy that can be present in conventional FPGAs as well as in hierarchical FPGAs (HFPGAs). A statistical model is adopted to investigate the routability on both kinds of FPGAs. The performance of our proposed FPGA models is compared to those without a routing hierarchy. Experimental methods are used to determine the flexibility and switch consumption of various routing resources. Results show that integrating a routing resource hierarchy into FPGAs consumes fewer routing resources and that the speed of designs implemented in the FPGAs can be greatly improved.
Acknowledgments

I would like to take this opportunity to express my sincere appreciation and thanks to my supervisor, Dr. Dilip K. Banerji, who had been a constant source of guidance, support, and encouragement during the course of this research.

I want to especially thank my wife Yan for her constant support and encouragement.

Many thanks to all my friends who exhorted me to work dedicatedly towards the fulfillment of the objectives of this research.
# Table of Contents

**Chapter 1 Introduction**

1. Overview of Field Programmable Gate Arrays (FPGAs) .......... 1
2. Hierarchical Routing Resources in FPGAs .......... 2
3. Thesis Motivation .......... 4

**Chapter 2 Background Information** .......... 8

1. Field Programmable Gate Arrays .......... 8
2. FPGA-based Design Procedure .......... 11
3. FPGA Routing Resources .......... 13
4. Xilinx FPGAs .......... 17

**Chapter 3 Previous Work** .......... 21

1. Stochastic Model .......... 21
2. Hierarchical FPGA Model .......... 26
3. Unresolved Problems .......... 28

**Chapter 4 Hierarchical Routing Resources** .......... 30

1. Overview .......... 30
2. Segmented Wires .......... 32
   1. Single-length Lines .......... 33
   2. Double-length Lines .......... 35
   3. Quad-length Lines .......... 37
3. Non-segmented Wires .......... 38
   1. Direct Connections .......... 39
   2. Long Lines .......... 40
4. Summary .......... 42

**Chapter 5 Routability Prediction for Conventional FPGAs**

with Hierarchical Routing Resources .......... 44

1. Motivation .......... 44
2. Architecture Assumptions .......... 45
5.3 Routability Prediction Model
5.4 Routability Prediction for Different Routing Wires
  5.4.1 Direct Connections
  5.4.2 Long Lines
  5.4.3 Single-length Lines
  5.4.4 Double-length Lines
  5.4.5 Quad-length Lines
5.5 Switches-used-per-tile Measurement

Chapter 6 Routability Prediction for Hierarchical FPGAs

Using Hierarchical Routing Resources
  6.1 Motivation
  6.2 HFPGA Model
  6.3 Parameters for the Enhanced HFPGA Model
    6.3.1 Number of Connections per Level
    6.3.2 Interconnection Length Distribution per Level
  6.4 Routability Prediction Models for HFPGAs

Chapter 7 Experimental Studies
  7.1 Introduction
  7.2 Implementation Process
  7.3 Performance of Conventional FPGA Model with a Routing Hierarchy
    7.3.1 Comparison with Brown's Model
    7.3.2 Effects of Different Routing Wires
    7.3.3 Length Probability Distribution Function
    7.3.4 Worst-case Delay
    7.3.5 Using CAD Software Tools for Validation
    7.3.6 Time Complexity Analysis
  7.4 Performance of Hierarchical FPGA with a Routing Hierarchy
7.4.1 Comparison with Li’s Model ...... 95
7.4.2 Length Probability Distribution in HFPGAs ...... 99

7.5 Conclusion ...... 102

Chapter 8 Conclusions ...... 103

8.1 Thesis Summary ...... 103
8.2 Suggestions for Future Work ...... 105

References ...... 107

Appendix ...... 110

1. Limitation of Xilinx Alliance Tools ...... 110
2. Obtaining the Real Used-switches-per-tile ...... 110
3. Analysis of Results ...... 111
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 2.1</td>
<td>Features of SRAM-based Xilinx FPGA Families</td>
<td>17</td>
</tr>
<tr>
<td>Table 2.2</td>
<td>Routing Resources for SRAM-based Xilinx FPGAs</td>
<td>19</td>
</tr>
<tr>
<td>Table 5.1</td>
<td>Ratio of Using Double-length Lines for Routing</td>
<td>57</td>
</tr>
<tr>
<td>Table 7.1</td>
<td>MCNC Benchmark Circuits Used for Experiments in Step 1</td>
<td>76</td>
</tr>
<tr>
<td>Table 7.2</td>
<td>MCNC Benchmark Circuits Used for Experiments in Step 2</td>
<td>78</td>
</tr>
<tr>
<td>Table 7.3</td>
<td>MCNC Benchmark Circuits Used for Experiments in Step 3</td>
<td>79</td>
</tr>
<tr>
<td>Table 7.4</td>
<td>Routing Parameters for Brown's Model and Our Model</td>
<td>80</td>
</tr>
<tr>
<td>Table 7.5</td>
<td>Switch Consumption and Prediction Time Comparison</td>
<td>80</td>
</tr>
<tr>
<td>Table 7.6</td>
<td>Routing Parameters for Xilinx XC4000XL Series FPGAs</td>
<td>90</td>
</tr>
<tr>
<td>Table 7.7a</td>
<td>Actual Routing Times for Alliance Software</td>
<td>90</td>
</tr>
<tr>
<td>Table 7.7b</td>
<td>Routability Prediction Results (Brown's Model)</td>
<td>90</td>
</tr>
<tr>
<td>Table 7.7c</td>
<td>Routability Prediction Results (Our Model)</td>
<td>91</td>
</tr>
<tr>
<td>Table 7.8</td>
<td>Routing Complexities for the Ten Benchmark Circuits</td>
<td>92</td>
</tr>
<tr>
<td>Table 7.9</td>
<td>Routing Parameters for Our HFPGA Model</td>
<td>96</td>
</tr>
<tr>
<td>Table 7.10</td>
<td>Routing Parameters for Li's Model</td>
<td>97</td>
</tr>
<tr>
<td>Table 7.11</td>
<td>Switch Consumption Comparison for the Two HFPGA Models</td>
<td>97</td>
</tr>
<tr>
<td>Table 7.12</td>
<td>Time Comparison for the Two HFPGA Models</td>
<td>98</td>
</tr>
<tr>
<td>Table A-1</td>
<td>Routing Information of Benchmark Circuits C432 and C499</td>
<td>111</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------</td>
<td>------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>1.2.1</td>
<td>Different Routing Resources</td>
<td>3</td>
</tr>
<tr>
<td>1.2.2</td>
<td>Hierarchy of Routing Resources in an FPGA</td>
<td>5</td>
</tr>
<tr>
<td>2.1.1</td>
<td>Different Types of Routing Architectures</td>
<td>10</td>
</tr>
<tr>
<td>2.2.1</td>
<td>A Typical FPGA-based Design Procedure</td>
<td>12</td>
</tr>
<tr>
<td>2.3.1</td>
<td>The FPGA Model</td>
<td>14</td>
</tr>
<tr>
<td>2.3.2</td>
<td>Connection Block Topology</td>
<td>15</td>
</tr>
<tr>
<td>2.3.3</td>
<td>Switch Block Topology</td>
<td>16</td>
</tr>
<tr>
<td>2.4.1</td>
<td>Xilinx FPGA Architecture</td>
<td>18</td>
</tr>
<tr>
<td>3.1.1</td>
<td>A Typical Connection in FPGAs</td>
<td>22</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Hierarchical FPGA Model</td>
<td>26</td>
</tr>
<tr>
<td>4.1.1</td>
<td>Hierarchy of Interconnections in FPGAs</td>
<td>32</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Single-length Line Structure</td>
<td>33</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Using Single-length Lines in Connections</td>
<td>34</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Double-length Lines Structure</td>
<td>36</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Quad-length Lines Structure</td>
<td>38</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Direct Connections Structure</td>
<td>40</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Long Lines Structure</td>
<td>41</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Routing Resources in C Block and S Block</td>
<td>42</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Different Routing Strategies</td>
<td>49</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Conditions for Using Long Lines in FPGAs</td>
<td>52</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Connection of Length of 2 Routed by Double-length Lines</td>
<td>55</td>
</tr>
<tr>
<td>5.3.4</td>
<td>Connections Routed by Double-length Lines</td>
<td>58</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Hierarchical FPGA Structure</td>
<td>64</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Hierarchical FPGA with a Routing Hierarchy</td>
<td>65</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Recursive Partitioning Scheme of a Design and the Physical Architecture for a</td>
<td>68</td>
</tr>
<tr>
<td></td>
<td>Two-dimensional Placement</td>
<td></td>
</tr>
</tbody>
</table>
Figure 6.3.2  Two Classes of Placement  .... 69
Figure 6.3.3  The Probability of Placement of an Interconnection  .... 70
Figure 7.2.1  CAD Process for Routability Prediction  .... 76
Figure 7.2.2  CAD Process for Xilinx Alliance Software Tools  .... 78
Figure 7.3.1  Switch Consumption Comparison  .... 81
Figure 7.3.2a  Prediction Time Comparison  .... 82
Figure 7.3.2b  Prediction Time Comparison (Continued)  .... 83
Figure 7.3.3  Relationship between Brown's Prediction Time and Our Prediction Time  .... 83
Figure 7.3.4  Proportion of Connections Routed by Each Type of Routing Wire for Circuit Des  .... 85
Figure 7.3.5  Connection Length Probability Distribution for Circuit Des (Partial)  .... 85
Figure 7.3.6  Length Probability Distribution Comparison  .... 88
Figure 7.3.7  Time Complexity Comparison  .... 92
Figure 7.3.8a  Time Complexity for Our Model  .... 93
Figure 7.3.8b  Time Complexity for Alliance Software  .... 94
Figure 7.4.1  Switch Consumption Comparison for Li's Model and Our Model  .... 98
Figure 7.4.2  Overall Length Probability Distribution for Circuit "Elliptic"  .... 99
Figure 7.4.3  Length Probability Distributions at Each Level  .... 100
Field Programmable Gate Arrays (FPGAs) have emerged as a key technology for implementing circuits in VLSI since their invention and introduction to the market. Much research has been done on their architecture and application. This thesis explores the effective utilization of routing hierarchy in both conventional FPGAs as well as hierarchical FPGAs (HFPGAs). A stochastic model is used to study the routability of both kind of FPGAs. Experimental methods have been used to determine the flexibility and cost of the routing architecture in order to show the effectiveness of methods developed in this thesis.

This chapter provides a brief introduction to the FPGAs and the hierarchical routing architecture. The motivation for and the organization of the thesis are also discussed.

1.1 Overview of Field Programmable Gate Arrays (FPGAs)

FPGAs are distinct from low density programmable logic devices (PLDs) and complex programmable logic devices (CPLDs) and typically offer the highest logic capacity amongst all programmable logic devices. A generic description of an FPGA is a programmable device with an internal array of logic blocks, surrounded by a ring of programmable input/output blocks, connected together via programmable interconnect elements. A typical FPGA contains from 64 to tens of thousands of logic blocks and an
even greater number of flip-flops. Chapter 2 presents a more detailed discussion on the different types of FPGAs.

Due to their relatively high density and programmability, FPGAs have been used in many applications since their introduction, especially for the design of low volume logic circuits. In much of the work to date, FPGAs have been found to be reasonable alternatives to custom hardware or software implementations of applications — they provide speed-ups over software through hardware implementation while still providing the flexibility to adapt the hardware to changing application needs. Static RAM (SRAM) based FPGAs have a great advantage over other types of FPGAs for quick prototyping; they can be re-programmed at users' site in a short time period. Any design error can be corrected easily and quickly, thus, reducing the overall development cost.

Despite their advantages, FPGAs also have some drawbacks. The major disadvantage is their speed. FPGAs are roughly three times slower than Mask Programmed Gate Arrays (MPGAs) [1], because of the increased resistance and capacitance of the routing resources on a chip. In addition, the logic density of an FPGA is about eight to twelve times less than that of an FPGA [1] due to the additional chip area used for switches and associated programming circuitry.

1.2 Hierarchical Routing Resources in FPGAs

Most FPGAs do not provide 100% interconnect between any two arbitrary logic blocks due to the cost and chip area required to do this. Instead, circuits are implemented in an FPGA by interconnecting its logic blocks through the routing wires and user-programmable routing switches, with the help of sophisticated software tools.
Routing switches are used to connect the logic blocks to the routing wires and to connect one routing wire to another. A number of switches can form a switch matrix which can interconnect the routing wires on its four sides. Routing switches consume significant chip area and have appreciable resistance and parasitic capacitance. Therefore, it is desirable to reduce the number of routing switches in implementing a design on an FPGA.

Routing wires are bi-directional solid metal lines. They can have various lengths and are spread out all over an FPGA. These wires are also called interconnection lines. There are mainly five types of interconnection lines, as shown in Figure 1.2.1:

- Direct Connections – lines connect adjacent logic blocks’ input and output pins directly, without going through a programmable switch matrix. This represents a zero-delay connection.
• Single-length Lines - a grid of horizontal and vertical lines which intersect at a switch matrix between each pair of logic blocks.

• Double-length Lines - a grid of horizontal and vertical lines which intersect at a switch matrix across TWO logic blocks. Thus, these lines bypass one switch matrix in their path.

• Quad-length Lines - a grid of horizontal and vertical lines which intersect at a switch matrix across FOUR logic blocks; these lines bypass three switch matrices in their path.

• Long Lines - form a grid of metal interconnection segments that run the entire length or width of the array.

The various routing resources in an FPGA can be viewed as a hierarchy, as shown in Figure 1.2.2. Different interconnect lengths between logic blocks can and should use appropriate routing resources to minimize switch consumption. This will reduce the total number of routing switches used and increase the speed of the implemented design. Furthermore, rich routing resources also provide more flexibility for mapping long interconnections. Chapter 4 provides a more detailed discussion on the hierarchical routing resources.

1.3 Thesis Motivation

The performance of an FPGA is dependent on the amount and flexibility of routing resources. The number of routing resources affects the logic density and operating speed. Flexibility is a key issue while implementing a design on FPGAs. Both
these aspects (number and flexibility) have a profound influence on the routability of a design.

A stochastic model for routability prediction in FPGAs is presented in [1]. This model probabilistically calculates the routability of a circuit in terms of percentage of successfully routed connections and conforms quite well to experimental results. The model assumes only Single-length Lines in an FPGA for routability prediction. Since most commercial FPGAs nowadays use wires of various lengths in their routing structures, it is desirable to consider the effect of different wire lengths on FPGA routability. The model presented in this thesis enhances the basic stochastic model to take advantage of hierarchical routing resources. This model, therefore, more realistically reflects the effect of different routing resources on the performance of FPGAs.
Furthermore, we apply our model to Hierarchical FPGAs (HFPGAs) [2,4] as well; such FPGAs contain both the logic blocks and routing resources organized in a hierarchy.

The model developed in this thesis can predict the routability of a design on a target FPGA. In addition, it can also be used to analyze the properties of different FPGA routing resources. Manufacturers can benefit by using the model to analyze a large number of benchmark circuits in a very short time, and to design new universal routing architectures or to optimize existing ones based on the analysis results. On the other hand, the end users can benefit from the prediction model as well. Due to the wide variety of FPGAs in the market that differ in size, speed, number, type of configurable logic blocks (CLBs), and routing architectures, selecting an FPGA for a given application remains a challenge. Since our prediction model can always find the minimum routing resource requirement for implementing a design, it provides a guideline to help the end users to choose an appropriate target FPGA. This is especially important for implementing designs for portable devices, due to the power constraints these devices have. In addition, the fast prediction speed is a great advantage. It can save both designers and end users a lot of time and help them to make a decision more quickly.

1.4 Thesis Organization

This thesis is organized as follows.

Chapter 2 provides brief background information on FPGA architectures and CAD tools available to the designer. The flexibility of routing resources is also discussed. This chapter also introduces a widely-used commercial FPGA – Xilinx FPGA.

Chapter 3 presents some previous work on FPGA routability prediction. A stochastic model is briefly reviewed. An existing enhanced model for Hierarchical
FPGA routability prediction is also introduced, and some remaining prediction problems not addressed by the existing models are discussed.

Chapter 4 presents a detailed hierarchical routing resource architecture used for FPGA routability prediction. Different types of routing resources are discussed.

Chapter 5 presents an enhanced routability prediction model for conventional FPGAs by using the hierarchical routing resources.

Chapter 6 extends the routability prediction model of chapter 5 to HFPGAs in which both the routing architecture and the logic blocks are organized in a hierarchy.

Chapter 7 describes the experimental procedure undertaken to study the performance of the proposed statistical models. Experimental results are analyzed in terms of flexibility, logic density, and speed.

Chapter 8 presents concluding remarks and outlines the directions for future work.
Chapter 2
Background Information

This chapter provides some necessary background information on FPGA classification, design automation steps, and routing resources.

Section 2.1 describes various types of FPGA architectures. Section 2.2 introduces a typical FPGA-based design procedure. Different FPGA routing resources are discussed in Section 2.3. The effect of flexibility of routing resources on FPGA routability is presented as well. Section 2.4 introduces a widely-used commercial FPGA family – Xilinx FPGAs.

2.1 Field Programmable Gate Arrays

There is a wide variety of architectures for FPGAs. The density and performance of these devices lie in the logic contained in their logic blocks, the technology used to manufacture the device, and the performance and efficiency of their routing architecture.

There are two major classes of FPGA architectures in terms of logic blocks: coarse-grained FPGAs and fine-grained FPGAs.

Coarse-grained architectures consist of fairly large logic blocks, often containing two or more look-up tables and two or more flip-flops. In a majority of these architectures, a four-input look-up table implements the actual logic. The larger logic block usually corresponds to improved performance.
The other type of architecture is called fine-grained. In these devices, there is a large number of relatively simple logic blocks which usually contain either a two-input logic function or a 4-to-1 multiplexer and a flip-flop.

Another difference in architecture is the underlying programming technology used in the device. Currently, high-density FPGAs are built using static memory (SRAM) programming technology, similar to microprocessors. SRAM-based FPGAs are reprogrammable, even in a system. The other common programming technology is called anti-fuse programming technology. An anti-fuse normally resides in a high-impedance state but can be “fused” into a low-impedance state when programmed by a high voltage. Anti-fuse-based FPGAs are one-time programmable (OTP).

All the internal connections in FPGAs are made up of metal wire segments with programmable switching points to implement routing. The arrangement of these wire segments and switching points are referred to as the routing architecture of an FPGA. Typically there are four routing architectures used in commercial FPGAs:

- Symmetrical Array

  In this architecture, all logic blocks are arranged as a symmetrical array. Routing resources spread out all over the array in horizontal and vertical channels to form a grid of routing architecture.

- Row-Based

  In this architecture, logic blocks are arranged in rows. Routing resources consist of horizontal wire segments of various lengths and are separated by routing switches. Dedicated vertical segments also exist for routing between the rows.
- **Sea-of-Gates**

  In this architecture, logic blocks are also arranged as a symmetrical array. Various lengths of routing resources are overlaid on top of the logic blocks. These logic blocks resemble the Sea-Of-Gates architecture used in some MPGAs. The wiring segments have various lengths: Local interconnect for short connections, Short Range interconnect for moderate-length connections, and Long Range interconnect for long connections.

- **Hierarchical**

  In this architecture, logic blocks and routing resources are arranged in a hierarchical fashion. Both intra-level and inter-level interconnects are provided.
Figure 2.1.1 shows the four routing architectures. The major differences between these architectures are the arrangement of logic blocks and the layout of the routing resources.

Despite these different architectures, the procedure for implementing a circuit in an FPGA is the same. Section 2.2 briefly discusses the design procedure.

2.2 FPGA-based Design Procedure

This section describes the process that is necessary to implement a customized logic circuit into an FPGA. A typical design flow is shown in Figure 2.2.1. The implementation is usually done with the help of CAD programs in the following major steps:

- Design Entry

The description of a logic circuit can be entered by using a schematic capture program. This involves using a graphical interface to interconnect circuit blocks/components selected from a component library. An alternative way to specify the logic circuit is to use Boolean expressions or state machine language. A number of different languages, such as VHDL and Verilog, are available to support this option.

- Logic Optimization

After the circuit has been fully entered, a logic optimization tool is used to perform optimizations on area, delay, or a combination of both. The design is then optimized by redundancy removal and subexpression elimination.
- Partition / Mapping

Once the design is optimized, another tool is used to transform the design into a circuit of FPGA logic blocks. Partitioning is the allocation of logic among multiple FPGAs, if required; mapping is the allocation of logic to logic blocks within a single FPGA. Good partitioning or mapping results in higher routing completion and better performance.

- Placement

Placement consists of assigning target-mapped functions to specific logic blocks within an FPGA. This can be done either manually or through the use of placement tools. The purpose of placement is to make the subsequent routing of connections feasible, as well as to optimize circuit delays.

- Routing

Following placement, the required interconnections among the logic blocks must be realized by selecting wire segments and routing switches within the FPGA’s
interconnection resources. The routing tools must ensure that 100 percent of the required connections are formed.

- Simulation

After placement and routing, the implemented design is simulated to ensure its functioning and to verify the timing issues. Design errors can be found and corrected at this stage.

- Programming Unit / Downloading

Once all the necessary steps are completed for implementing the design, the CAD system can feed the result to a programming unit that is used to configure the FPGA. After this stage, the programmable device is configured and ready for use.

Routing is perhaps the most crucial step in the design process since a circuit must be successfully routed in order to be usable. Several architectural parameters affect the routability of FPGAs; these parameters are discussed in Section 2.3.

2.3 FPGA Routing Resources

As stated in Chapter 1, routing resources in FPGAs include interconnection lines and programmable routing switches. All the interconnection lines are made of metal and have low resistance and capacitance. The programmable routing switches have a number of implementations depending on the manufacturer. For example, one implementation uses a CMOS pass-transistor controlled by a static memory bit [9]. Regardless of the particular implementation, routing switches typically consume significant chip area as compared to the metal interconnect lines.
In order to illustrate the FPGA routing resources, an appropriate model is adopted to define an FPGA [1], as shown in Figure 2.3.1. This model consists of logic blocks (L) and three routing resources: Connection blocks (C), Switch blocks (S), and routing channel segments. The L blocks contain the combinational and sequential logic that form the functionality of circuits. Each logic block has a number of input and output pins connected to the four adjacent C blocks. The routing resources are described as follows:

- **Connection Blocks**

  The C blocks are switch boxes that are used to connect the logic blocks to the routing channels via programmable switches. The topology is illustrated in Figure 2.3.2, where a routing switch is identified by a dot. The channel wires (vertical lines in the figure) pass uninterrupted through the C block and have the option of connecting to
the logic block pins through the switches. Depending on the topology of the C block, each logic block pin may be connectable to either all or some of the wiring segments that pass through the C block. The flexibility of the C block is represented by the parameter $F_c$, which defines the number of wires that each logic block pin can connect to. For the example shown in the figure, $F_c$ is 2. The lower the value of $F_c$, the harder the FPGA is to route. Previous research has shown that $F_c$ should be at least half of the number of wiring segments in order to achieve high routability [1]. The pattern of the switches in a connection block can also have effects on routability, particularly when $F_c$ is low [1]. But, that is beyond the scope of this thesis.

- Switch blocks

The S blocks are switch matrices. They are used to connect wires in one channel segment to those in another. The general nature of a switch block is illustrated in Figure 2.3.3. Depending on the topology, each wiring segment on one side of an S block may be connected to either all or some of the wiring segments on other sides of the S block. A connection passing through an S block may do so through a switch, which is shown as a dotted line in the figure. Alternatively, a connection may be hard-wired, which is shown as a solid line in the figure. The flexibility of an S block, $F_s$, is defined as the number of
other wiring segments that each wiring segment entering an S block can connect to. For the example shown in the figure, the wiring segment at the top middle of the S block can be switched to six other wiring segments and, therefore, $F_s$ is 6. A higher $F_s$ results in more possible paths for a connection, which will increase the routability of a design.

- Routing channels

Routing channels consist of a number of bi-directional signal wires. The flexibility of routing channels is denoted by the variable $W$, which specifies the number of wires in a routing channel. $W$ is also called the channel width. The more wires in a routing channel, the easier a connection can be routed. However, more wires also occupy more space on a chip. Furthermore, as wiring channels become wider, the C blocks and S blocks become bigger and more complex. Delay and chip area costs may increase more than linearly with wiring channel width.

As discussed, the flexibility of routing resources can be altered by changing the connectivity in the C blocks, the S blocks, or the number of tracks in each routing channel. Next chapter introduces a widely-used commercial FPGA family and explores their routing architectures.
2.4 Xilinx FPGAs

The first FPGA was introduced by Xilinx Corporation in 1985 [1]. Since then, FPGAs have evolved considerably, with several vendors offering a variety of products. This particular FPGA family (Xilinx) has been chosen for discussion because it is a representative example of a state-of-the-art device that is in widespread use. Our routability prediction model is based on the Xilinx FPGA architecture.

The general architecture of Xilinx FPGAs is shown in Figure 2.4.1. It consists of a two-dimensional array of programmable logic blocks, called Configurable Logic Blocks (CLBs), surrounded by I/O Blocks (IOBs). Each CLB consists of one or more Look-Up Tables (LUTs), as well as flip-flops. Horizontal and vertical routing channels are available between CLBs. There are several families of SRAM-based Xilinx FPGAs. The major features of these families are listed in Table 2.4.1.

<table>
<thead>
<tr>
<th>FPGA Families</th>
<th>CLB Matrix Size</th>
<th>Maximum IOB Size</th>
<th>CLB Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC3000</td>
<td>8x8–22x22</td>
<td>64–176</td>
<td>One 5-input LUT per CLB</td>
</tr>
<tr>
<td>XC4000</td>
<td>8x8–56x56</td>
<td>64–448</td>
<td>Two 4-input LUTs per CLB</td>
</tr>
<tr>
<td>XC5200</td>
<td>8x8–22x22</td>
<td>84–244</td>
<td>Four 4-input LUTs * per CLB</td>
</tr>
<tr>
<td>Virtex</td>
<td>16x24–64x96</td>
<td>180–512</td>
<td>Four 4-input LUTs per CLB</td>
</tr>
<tr>
<td>Spartan</td>
<td>10x10–28x28</td>
<td>77–205</td>
<td>Three 4-input LUTs per CLB</td>
</tr>
</tbody>
</table>

* The LUTs in XC5200 serial FPGAs are also called Logic Cells
Figure 2.4.1 Xilinx FPGA Architecture
Table 2.2 Routing Resources for SRAM-based Xilinx FPGAs

<table>
<thead>
<tr>
<th>SRAM-based Xilinx FPGA Families</th>
<th>Routing Resources</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>XC3000 Family</strong></td>
<td>Programmable Switch Matrix (PSM)</td>
</tr>
<tr>
<td></td>
<td>Direct Connections</td>
</tr>
<tr>
<td></td>
<td>General Purpose Interconnect</td>
</tr>
<tr>
<td></td>
<td>Longlines</td>
</tr>
<tr>
<td><strong>XC4000 Family</strong></td>
<td>Programmable Switch Matrix (PSM)</td>
</tr>
<tr>
<td></td>
<td>Single-length Lines</td>
</tr>
<tr>
<td></td>
<td>Double-length Lines</td>
</tr>
<tr>
<td></td>
<td>Quad-length Lines</td>
</tr>
<tr>
<td></td>
<td>Long Lines</td>
</tr>
<tr>
<td></td>
<td>Direct Connections (Only in XC4000X)</td>
</tr>
<tr>
<td><strong>XC5200 Family</strong></td>
<td>General Routing Matrix (GRM)</td>
</tr>
<tr>
<td></td>
<td>VersaBlock Routing (local routing)</td>
</tr>
<tr>
<td></td>
<td>Direct Connects</td>
</tr>
<tr>
<td></td>
<td>Single-length Lines</td>
</tr>
<tr>
<td></td>
<td>Double-length Lines</td>
</tr>
<tr>
<td></td>
<td>Long Lines</td>
</tr>
<tr>
<td><strong>Virtex Family</strong></td>
<td>General Routing Matrix (GRM)</td>
</tr>
<tr>
<td></td>
<td>Direct Connections</td>
</tr>
<tr>
<td></td>
<td>Feed-back Path</td>
</tr>
<tr>
<td></td>
<td>Single-length Lines</td>
</tr>
<tr>
<td></td>
<td>Hex-length Lines</td>
</tr>
<tr>
<td></td>
<td>Long Lines</td>
</tr>
<tr>
<td><strong>Spartan Family</strong></td>
<td>Programmable Switch Matrix (PSM)</td>
</tr>
<tr>
<td></td>
<td>Single-length Lines</td>
</tr>
<tr>
<td></td>
<td>Double-length Lines</td>
</tr>
<tr>
<td></td>
<td>Long Lines</td>
</tr>
</tbody>
</table>
Our major interest lies in the routing resources provided in the Xilinx FPGAs, which are listed in Table 2.4.2. Most SRAM-based Xilinx FPGAs provide Direct Interconnects, Single-length Lines, and Long Lines. The Direct Interconnects provide connections from the output of a CLB to its right, top, left, and bottom neighbors. For connections that span more than one CLB, the Single-length Lines provide horizontal and vertical wiring segments. Each wiring segment spans only the length or width of one CLB, but longer connections can be formed by using programmable switch matrices. Long Lines bypass the switch matrices and span the entire length or width of the FPGA. The XC4000 routing architecture is different from the earlier Xilinx FPGAs, with the most obvious difference being the enhancement of Single-length Interconnects with new resources, called Double-length Lines and Quad-length Lines. The Single-length Lines are only intended for relatively short connections or those that do not have critical timing requirements. The Double-length Lines are similar to the Single-length Lines, except that each one passes through half as many switch matrices. The Quad-length Lines are appropriate for longer connections since each one passes through only one-quarter of the switch matrices as compared to Single-length Lines. These lines offer lower routing delays for moderate and long connections that are not appropriate for the low-skew Long Lines.
Chapter 3
Previous Work

This chapter provides an overview of some previous work done on FPGA routing prediction. In Section 3.1, a stochastic model for FPGA routability is briefly reviewed. This model is used in [1] to study the flexibility of routing architecture in symmetrical FPGAs. An enhanced stochastic model for FPGAs with hierarchical architecture is introduced in Section 3.2. Some of the remaining problems associated with routability prediction are discussed in Section 3.3.

3.1 Stochastic Model

This model, first proposed by Brown at the University of Toronto, is discussed in [1]; it is applicable to symmetrical FPGAs. An FPGA is represented as NxN logic blocks connectable with routing resources. In this model, a circuit with a total of $C_T$ two-point connections is assumed to be routed on a symmetrical FPGA. It is further assumed that the number of connections per logic block can be drawn independently from a Poisson distribution with parameter $\lambda$, where $\lambda$ is defined as the ratio of total number of connections in a circuit to the total number of logic blocks in the FPGA. The $C_T$ connections are individually denoted as $C_1$, $C_2$, ..., $C_T$ and the statistical events for successfully routing the corresponding connections are denoted as $R_{C_1}$, $R_{C_2}$, ..., $R_{C_T}$.

Routing is done as a random process with the assumption that a connection is assigned a single path through the routing channels. The flexibility of the routing resources, namely
Figure 3.1.1 A Typical Connection in FPGAs

$W$, $F_c$, and $F_s$, are evaluated for a successfully routed connection. The model probabilistically routes the connections serially, and takes into consideration the effect that each successfully routed connection may have on other connections to be routed later.

Figure 3.1.1 shows how a typical connection is routed on a symmetrical FPGA. A two-point connection starts at the source logic block. It is connected to the routing channel through a connection block (C block). Passing through a series of switch blocks (S blocks), it ends at the destination logic block through the destination connection block (C block). For the general case of a connection $C_n$, the routing event is denoted as $R_{C_n}$ and $P(R_{C_n})$ denotes the probability of successfully routing $C_n$. The length of $C_n$, which is denoted as $L_{C_n}$, is defined in terms of logic block hops between the source and destination and is assumed that $L_{C_n} = n+1$. Variable $n$ is defined as the number of S blocks in the connection path. The statistical event that corresponds to this assumption is written as $L_{n+1}$. The event $R_{C_n}$ can be seen as a sequence of events $X_1, S_1, S_2, ..., S_n, X_2$ with the following definitions:
- X₁ – the event that the starting point (pin) of Cᵢ in source LB can be connected to a track in the first C block.
- S₁, S₂, ..., Sₙ – the events that Cᵢ can get connected through the first, the second, and up to the nᵗʰ S blocks. For a connection of length L_Cᵢ, there are L_Cᵢ - 1 such S block events.
- X₂ – the event that at least one of the tracks in the last C block can be used by Cᵢ to connect to the desired logic block pin at the destination.

From the preceding discussion, we can see that a connection of length Lᵢ₊₁ is routed successfully only if all the associated events occur. Therefore, the probability of routing a connection is equal to the product of the probabilities associated with each of these events:

\[ P(R_{Cᵢ} | L_{ᵢ₊₁}) = P(X₁ ∩ S₁ ∩ S₂ ∩ ... ∩ Sₙ ∩ X₂) \]

\[ = P(X₁) * P(S₁ | X₁) * P(S₂ | S₁ ∩ X₁) * ... * P(Sₙ | Sₙ₋₁ ∩ ... ∩ S₁ ∩ X₁) \]

\[ * P(X₂ | Sₙ ∩ ... S₁ ∩ X₁) \]  

(3.1)

All the events are not independent and, therefore, the values corresponding to each of these events are calculated separately. The formulae for these events are presented here. An elegant way of obtaining these formulae can be found in [1].

The probability of event X₁ is related to the flexibility of FPGA’s C block \( F_r \) and channel width \( W \):

\[ P(X₁) = \sum_{a=1}^{F_r} p(\lambda_a \frac{F_c}{W} F_c - a) \]  

(3.2)
where \( p(\lambda, \frac{F_c}{W}, F_c - a) \) is the Poisson distribution for channel density, with \( \lambda = \frac{\bar{R}}{2} \).

\( \bar{R} \) denotes the average connection length for the circuit, and \( \lambda \) is defined as the ratio of expected number of routed connections to the total number of logic blocks.

Followed by event \( X_1 \), the probability of events \( S_1, S_2, \ldots, \) and up to \( S_n \) is calculated recursively as:

\[
P(S_n \mid S_{n-1} \cap \ldots \cap S_1 \cap X_1) = \frac{\lambda}{\sum_{j=1}^{w} P(A_j^{S_{n-1}})} \left[ P(Z_1) + \sum_{k=1}^{w} \prod_{j=1}^{w} P(A_j^{S_{n-1}}) \right] * p(\lambda, \frac{\alpha_1 a}{W}, \alpha_1 a - k) + \]

\[
P(S_n \mid S_{n-1} \cap \ldots \cap S_1 \cap X_1) = \frac{\lambda}{\sum_{j=1}^{w} P(A_j^{S_{n-1}})} \left[ P(Z_2) + \sum_{k=1}^{w} \prod_{j=1}^{w} P(A_j^{S_{n-1}}) \right] * p(\lambda, \frac{\alpha_2 a}{W}, \alpha_2 a - k)
\]

(3.3)

where \( P(Z_1) \) is the probability that a connection runs straight through an \( S \) block, and \( P(Z_2) \) is the probability that a connection turns at an \( S \) block. Variables \( \alpha_1 \) and \( \alpha_2 \) are scaling factors corresponding to \( P(Z_1) \) and \( P(Z_2) \). \( P(\lambda_k^{S_n}) \) is given by

\[
P(\lambda_k^{S_n}) = \sum_{a=1}^{w} P(A_a^{S_{n-1}} \mid S_{n-1} \cap \ldots \cap S_1 \cap X_1) * p(\lambda, a, a - k)
\]

(3.4)

where \( A_k^{S_n} \) is the event that the \( n^{th} \) \( S \) block in the path has exactly \( k \) available outgoing tracks. And, finally, the probability of the event \( X_2 \) is given by:

\[
P(X_2 \mid SX) = 1 - \sum_{n=1}^{w} \frac{P(A_n^{S_n})}{\sum_{j=1}^{w} P(A_j^{S_n})} \left[ \frac{w}{w - F_c} \frac{C_a}{w} \right]
\]

(3.5)

where \( SX \) is a substitution for expression \( S_n \cap S_{n-1} \cap \ldots \cap S_1 \cap X_1 \), and \( bC_a \) refers to the number of combinations of “a” things taken “b” at a time.
To calculate $P(R_{C_i})$, define $LC_i = l_{\text{max}}$ as the maximum length of any connection and $L_{i_{\text{max}}}$ as the corresponding event. Next, consider events $L_1, \ldots, L_{i_{\text{max}}}$ corresponding to the possible values of $LC_i$. Since the occurrence of $R_{C_i}$ implies exactly one of $L_1, \ldots, L_{i_{\text{max}}}$, the probability can be calculated as:

$$P(R_{C_i}) = \sum_{i=1}^{l_{\text{max}}} P(L_i) \cdot P(R_{C_i} | L_i)$$  \hspace{1cm} (3.6)

where $P(L_i)$ is the probability distribution of connection length. In [1] it is assumed to be a geometric distribution with mean $\bar{R}$, and is given by:

$$P(L_i) = p \cdot q^{i-1}$$  \hspace{1cm} (3.7)

where $p = \frac{1}{R}$ and $q = 1 - p$. We will discuss the probability distribution of connection lengths in more details later.

The overall routability is defined as the percentage of connections that can be routed divided by the total number of connections in the circuit:

$$\text{Routability} = \frac{1}{C_T} \sum_{i=1}^{C_T} P(R_{C_i}),$$  \hspace{1cm} (3.8)

During the routing process, routing resources are consumed by the required connections. As the number of routed connections increases, the probability of routing a subsequent connection should decrease. Thus, after $n$ connections have been routed, the number of connections in a channel should get changed and parameter $\lambda$ can be recalculated as:

$$\lambda = \frac{1}{N^2} \sum_{i=1}^{n} P(R_{C_i})$$  \hspace{1cm} (3.9)
where $N^2$ is the total number of logic blocks in an FPGA. Parameter $\lambda_s$ is also modified accordingly. This is a key point in the stochastic model since it reflects the effect of routed connections on the connections to be routed later.

This stochastic model produces good results for predicting routability in symmetrical FPGAs. With some modifications, this model can be applied to Hierarchical FPGAs. This is discussed in the next section.

### 3.2 Hierarchical FPGA Model

Hierarchical FPGAs (HFPGAs) are different from symmetrical FPGAs by their logic block arrangement and routing architecture distribution. A model proposed in [2] is shown in Figure 3.2.1. The model groups logic blocks together to form multiple levels of hierarchy.

![Hierarchical FPGA Model](image.png)

Figure 3.2.1 Hierarchical FPGA Model
The basic logic blocks are considered to be level 0 elements. A logic circuit is mapped into these level 0 elements first. Then $K = k \times k$ level 0 elements and the corresponding routing resources are grouped into a higher level cluster which is called a 1st-level element. Next, the $K$ 1st-level elements are grouped into a 2nd-level element. This grouping process is performed recursively until the whole FPGA is covered by an $m^{th}$-level element. The elements in each level are symmetrically distributed. Routing is done through global and local routing resources. Each element uses global routing resources to get connected to other elements at the same level. The routing resources inside an element are used for local interconnections. Research shows that an HFPGA needs far fewer switches to implement a circuit than a symmetrical FPGA [3]. Additionally, the maximum delay between any two logic blocks is more predictable in a Hierarchical FPGA compared to that in a symmetrical FPGA [3].

An enhanced stochastic model has been developed for HFPGAs in [2]. The model can predict the routability of a circuit after technology mapping and before placement. Prediction is done at each level and then added up to give the overall routability of the whole circuit. In the stochastic prediction model, parameters like average connection length ($\overline{R}$) and total number of connections ($C_T$) are crucial. These values are not known before placement and routing. Therefore, Rent’s rule [23] is used to estimate these parameters from the netlist after technology mapping.

Rent’s rule describes the relationship between the number of elementary blocks, $B$, in a module of a mapped design and the number of external connections (pins) $P$ on the module:

$$P = CB^r,$$  \hspace{1cm} (3.10)
where $C$ is the *average* number of interconnections per elementary block in the module, and $r$ is called the Rent exponent. This exponent is a measure of the interconnection complexity in the design. Its value is bounded by 0 and 1, with increasing values for increasing interconnection complexity. By using Rent's rule, for an HFPGA with $M$-levels and $K$-element blocks at each level, the number of pins at $i^{th}$-level can be calculated as [2]:

$$K^{M-i} \cdot C \cdot K^{ir} - K^{M+(i+1)} \cdot C \cdot K^{(i+1)r}$$

(3.11)

Assuming that the number of connections at each level is proportional to the number of pins, the total number of connection at the $i^{th}$-level is given by [2]:

$$C_{T} = \alpha \cdot C \cdot (K^{M-i} \cdot K^{ir} - K^{M+(i+1)} \cdot K^{(i+1)r})$$

(3.12)

where, constant $\alpha = \frac{\# \text{ connections}}{\# \text{ pins}}$. For $K$ elements at the $i^{th}$-level, assume that they are fully connected to each other; then the average length for all the connections can be estimated as: $\frac{2}{3} \sqrt{K^{i+1}}$ [2]. The overall routability for an HFPGA with $M$ levels is given by:

$$Routability = \sum_{i=0}^{M-1} \frac{Routability_{i}}{M}$$

(3.13)

3.3 Unresolved Problems

The stochastic models for both the symmetrical FPGA and the Hierarchical FPGA can predict the routability of a design quite well. However, these models suffer from some inherent limitations, which must be addressed:
1. Although multi-length connection lines are frequently used in commercial FPGAs, like in Xilinx FPGAs, only Single-length Lines are considered in these models. This approach is taken because the channel density distribution in some FPGAs, such as those having tracks with segments that span multiple logic blocks, may not be Poisson distributed. Thus, the previous stochastic models do not apply in this situation, without suitable modifications.

2. The distribution function for connection length used in the models is assumed to be geometric. Experiments show that a better distribution function can be used instead.

3. In [2], the average interconnection length at each level is derived from the assumption that the HFPGA is fully connected. This is not always the case. Circuits with different complexities may have different average connection lengths.

4. Predicting routability for big circuits using existing models [1,2] is time-consuming. This is due to the increased complexity of computation involved in the stochastic models for big circuits.

Some improvements can be made to optimize the models and get better results. Next chapter presents a detailed description of hierarchical routing resources. Chapter 5 introduces an FPGA model using hierarchical routing resources to achieve better and more realistic results on routability prediction.
Chapter 4
Hierarchical Routing Resources

This chapter presents a detailed description of hierarchical routing resources available on Xilinx FPGAs. Section 4.1 briefly describes the basic concepts of measurement in routing. Sections 4.2 and 4.3 introduce different routing resources. Section 4.4 provides a summary of all the routing resources.

4.1 Overview

In FPGAs, all internal connections are composed of metal segments with programmable switch points and switch matrices to complete the desired routing. In general, a connection $Ci$ starts from a logic block located at $(x_1, y_1)$ and travels through routing channels to another logic block located at $(x_2, y_2)$. The length of $Ci$ is denoted as $LC_i$, which is the Manhattan distance in terms of logic block hops between the source and destination logic blocks

$$LC_i = |x_1 - x_2| + |y_1 - y_2|$$

(4.1)

Since $LC_i$ is composed of the distances on the x and y axes, we further define $|LC_i^x|$ and $|LC_i^y|$ to be the Manhattan distances in the x and y direction, respectively; $LC_i$ can then be represented as $|LC_i^x| + |LC_i^y|$. Furthermore, the number of S blocks that $Ci$ passes through is $|LC_i^x|-1$ on x-axis and $|LC_i^y|-1$ on y-axis.
The programmable switches in the connection blocks and switch blocks occupy significant chip area and cause signal delay. Reducing the number of switches used in interconnections is a primary goal in FPGA-based designs. A common method to measure the performance of FPGA routing is to use “switches-per-tile” count. A tile is a cluster consisting of one logic block, two connection blocks, and one switch block. The whole FPGA can be covered by replicated tiles across the chip. The fewer switches used per tile, the better performance the design has. The number of switches in a connection block and a switch block depends on the number of pins on each logic block (P), the number of sides that each pin appears on (T), and on the parameters Fc, Fs, and W. These numbers are computed as follows:

\[
\# \text{Switches in Connection Block} = \frac{1}{2} T P F_c \quad (4.2)
\]

\[
\# \text{Switches in Switch Block} = 2 F_s W \quad (4.3)
\]

T and P are set to be 2 and 7 respectively in [1] and [2]. To compare the switch count in our models with the results reported in [1] and [2], we use the same values of T and P. Therefore, the total number of switches per tile, \(S_r\), is given by

\[
S_r = 2 \# \text{Switches in Connection Block} + \# \text{Switches in Switch Block}
\]

\[
= 2 \frac{1}{2} \times 2 \times 7 \times F_c + 2 F_s W
\]

\[
= 14 F_c + 2 F_s W \quad (4.4)
\]

As stated in Chapter 2, there are different lengths of interconnections in FPGAs. An appropriate length can greatly reduce the number of switches used in connections. To
explore the effect of various interconnection lengths on the performance of FPGAs, we propose a hierarchical routing resource structure. Since our primary interest is interconnections, we focus on the routing wires only, as shown in Figure 4.1.1.

Sections 4.2 and 4.3 describe the different types of interconnections. The following aspects are covered for each type:

- Definition of the routing wire type
- Structural description
- Usage in FPGAs
- Switch consumption

4.2 Segmented Wires

As shown in Figure 4.1.1, segmented wires include Single-length Lines, Double-length Lines, and Quad-length Lines. They have to use programmable switch matrices to route connections in FPGAs. As the names indicate, the difference between these wires is
the segmented length used for routing. The following sections provide detailed descriptions of each wire type.

4.2.1 Single-length Lines

Single-length Lines are the routing channels between two adjacent logic blocks. These lines intersect at each switch matrix to form a grid of interconnections. The structure is shown in Figure 4.2.1.

Theoretically, any connection can be routed by using only Single-length Lines, as long as the routing parameters \((F_r, F_s, \text{ and } W)\) are chosen properly. However, such connections cause delays whenever they go through a switch matrix. Therefore, Single-length Lines are not suitable for routing signals traveling for long distances, especially for those critical signals which have time constraints. Single-length Lines are preferred for conducting signals within a localized area. In our model, a Single-length Line is considered to have been used whenever any switch in the associated segment gets used.

![Figure 4.2.1 Single-length Line Structure](image)
for a connection. For example, in connecting two adjacent logic blocks, a connection can be established through the C block between the logic blocks. However, the connection uses a routing track between two adjacent S blocks. Therefore, this connection, which has a length of 1 hop, is counted as a Single-length Line connection. Single-length Lines also can be used to connect diagonal logic blocks with a length of 2 hops. These two cases are shown in Figure 4.2.2.

Figure 4.2.2 Using Single-length Lines in Connections

In the figure, a connection with a length of 1 hop from pin b1 to pin d2 occupies one Single-length Line in the routing channel. Another connection with a length of 2 hops that connects diagonal logic blocks from pin c1 to pin b2 is established by using two Single-length Lines and a switch block between the two logic blocks.

Commercial FPGAs usually have Single-length Lines or equivalent routing lines in their routing structures. Other types of routing wires are also provided for longer connections. For example, Xilinx XC4000X series FPGAs use 8 Single-length Lines in
each vertical and horizontal routing channel, which occupy 20% of the total channel width.

Equation 4.5 can be used to calculate \( S_{T_s} \), which denotes switches-per-tile for Single-length Lines

\[
S_{T_s} = 14 * F_{cr} + 2 * F_{ss} * W_s
\]

(4.5)

where \( F_{cr} \), \( F_{ss} \), and \( W_s \) are the routing parameters for Single-length Lines. Different values of these parameters are used in our model to produce different levels of flexibility and performance in routing.

4.2.2 Double-length Lines

Double-length Lines are twice as long as the Single-length Lines and consist of a group of metal segments. They run past two logic blocks before entering a switch matrix, as shown in Figure 4.2.3. Double-length Lines are grouped in pairs with the switch matrices staggered, so that each line goes through a switch matrix at every other row or column of logic blocks. In every switch matrix, half of the Double-length Lines entering are hard-wired, that is, they go straight through without encountering any switches. The rest of the Double-length Lines use switches to connect with other segments to pass through or change direction.

Double-length Lines provide faster signal routing over intermediate distances while retaining routing flexibility. Compared to connections routed only by Single-length Lines, those routed by Double-length Lines use 50% of the switches. In addition, signal delays are also reduced in half.
Figure 4.2.3 Double-length Line Structure

In our model, a Double-length Line is considered to have been used for routing whenever any switch in the associated segment gets used for a connection. In general, for a connection of length $LC_i$ hops, if $|LC_i^x|$ is either 2 or 3 and $|LC_i^y|$ is less than 4, or vice versa, the connection can be routed by using a Double-length Line. In other words, connections of length between 2 hops (when $|LC_i^x| = 2$ and $|LC_i^y| = 0$, or vice versa) and 6 hops (when $|LC_i^x| = 3$ and $|LC_i^y| = 3$) can use Double-length Lines for routing. If either $|LC_i^x|$ or $|LC_i^y|$ is equal to or longer than 4 hops, Quad-length Lines are used instead, as described in Section 4.2.3.

To calculate the number of switches-per-tile for Double-length Lines, we can use the same equation for C block switch count. However, the equation for S block switch count needs to be modified. The routing parameters for Double-length Lines are denoted
as $F_{rd}$, $F_{sd}$, and $W_d$. Since only half of the $W_d$ lines entering a switch matrix use switches for routing, Equation 4.3 has to be revised as

$$\# Switches in Switch Block = 2 \times F_{sd} \times \frac{1}{2} W_d$$

$$= F_{sd} \times W_d$$ (4.6)

Hence, the number of switches-per-tile for Double-length Lines, $S_{T_d}$, is given by:

$$S_{T_d} = 14 \times F_{rd} + F_{sd} \times W_d$$ (4.7)

### 4.2.3 Quad-length Lines

Quad-length Lines are four times as long as the Single-length Lines. They run past four logic blocks before entering a switch matrix. They are grouped in fours, with the switch matrices staggered, so that each line goes through a switch matrix at every fourth logic block in a row or a column. The structure is shown in Figure 4.2.4. As illustrated in the figure, three-quarters of the Quad-length Lines are hard-wired to pass through the switch matrices.

Due to the greatly reduced number of switches in connections, Quad-length Lines provide the fastest available method of routing signals through S blocks for long distances across FPGAs. They are considered to have been used whenever any switch in the associated routing channel gets used for a connection. In our model, for a connection of length $LC_i$, if either $|LC_i^x|$ or $|LC_i^y|$ is 4 hops or longer, Quad-length Lines are used for routing.

Calculating the number of switches-per-tile for Quad-length Lines is similar to
that for Double-length Lines. The routing parameters for Quad-length Lines are denoted as $W_q$, $F_{rq}$, and $F_{sq}$. The number of switches in a switch block for Quad-length Lines is given by

$$\# \text{Switches in Switch Block} = 2 \cdot F_{sq} \cdot \frac{1}{4} W_q$$

$$= \frac{1}{2} F_{sq} \cdot W_q$$ (4.8)

The number of switches per tile, $S_{T_q}$, is given by

$$S_{T_q} = 14 \cdot F_{cq} + \frac{1}{2} F_{sq} \cdot W_q$$ (4.9)

4.3 Non-segmented Wires

Non-segmented wires include Direct Connections and Long Lines. These do not go through programmable switch matrices. Therefore, they provide the fastest method
for connecting two logic blocks. However, not all the connections can be routed by non-
segmented wires. Direct Connections can only be used between neighboring logic blocks
and Long Lines can only be used for connections that do not turn. That is, these
cannections must be in the same row or column.

4.3.1 Direct Connections

Direct Connections provide the most efficient connection of networks between
adjacent logic blocks. Signals that are routed from one logic block to another using Direct
Connections exhibit minimum propagation delays and use no other interconnection
resources. A typical Direct Connection structure is shown in Figure 4.3.1. The logic
block shown here has six pins. Four of them (A, B, C, and D) are input pins, and the
remaining two (X and Y) are output pins. For each logic block, the output pin X may be
connected directly to the input pin B of the logic block immediately to its right and to the
input pin C of the logic block to its left. The output pin Y can use direct interconnections
to drive the input pin D of the block immediately above and the input pin A of the block
below. Direct Connections should be used by place and route software whenever possible
to minimize interconnect delays.

Each Direct Connection uses only one switch on the input pin side of a logic
block. In un-programmed status, the switch is non-conducting. A connection that goes
through the switch can be established by routing software. The number of switches used
per tile, $S_{T_{de}}$, is given by

$$S_{T_{de}} = T*I$$

(4.10)
where $T$ is the number of sides that each pin appears on and $I$ is the number of input pins in a logic block. In our model, $T$ is set to be 2 and $I$ is set to be 4. Therefore, the total number of switches-per-tile for Direct Connections is 8.

### 4.3.2 Long Lines

Long Lines bypass all switch matrices and form a grid of interconnections that run vertically and horizontally the entire length or width of the logic block arrays. They are intended for high fan-out, time-critical signal nets, or nets that are distributed over long distances. A typical illustration of Long Lines is shown in Figure 4.3.2.

In commercial FPGAs, Long Lines are usually driven by tri-state or open-drain
drivers. Therefore, they can be used to implement unidirectional or bidirectional buses, wide multiplexers, or wired-AND functions. Logic blocks connect to Long Lines through C blocks. Theoretically, they also can route any length of straight connections in a channel. However, if a signal does not have critical time constraints, it is not necessary to use Long Lines for such connections. Instead, other types of routing wires should be considered. In our model, Long Lines route only straight connections in a channel that are longer than 5 hops and up to the entire length or width of FPGAs.

Switch count for Long Lines is different from that of segmented routing wires. Long Lines do not use any switch matrix. Therefore, only the switches in C blocks are counted. The number of switches-per-tile for Long Lines, $S_{T_l}$, is obtained by using only Equation 4.2 with routing parameter $F_{ct}$

$$S_{T_l} = 14 \times F_{ct}$$

(4.11)
4.4 Summary

In order to use the hierarchical routing resources, we combine all the connection lines described in the previous sections. The complete structure for a C block and an S block, along with the routing wires is shown in Figure 4.4.1.

![Image of Figure 4.4.1 Routing Resources in C Block and S Block]

The total number of switches-per-tile, \( S_T \), can be calculated by adding up Equations 4.5, 4.7, 4.9, 4.10, and 4.11:

\[
S_T = S_{T_d} + S_{T_s} + S_{T_d} + S_{T_q} + S_{T_l}
\]

\[
= 8 + 14 \cdot (F_{cs} + F_{cd} + F_{cq} + F_{cl}) + 2 \cdot F_{ss} \cdot W_s + F_{sd} \cdot W_d + \frac{1}{2} \cdot F_{sq} \cdot W_q \quad (4.12)
\]
In the next chapter, we present an enhanced routability prediction model that takes advantage of the hierarchical routing resources.
Chapter 5
Routability Prediction for Conventional FPGAs with Hierarchical Routing Resources

In this chapter, we present an improved stochastic model for routability prediction in conventional FPGAs with hierarchical routing resources. Different lengths of routing wires are considered and incorporated in our model. Section 5.1 describes the motivation for developing the improved model. Some assumptions are included in Section 5.2, and the detailed model is presented in Section 5.3.

5.1 Motivation

As we stated in Chapter 3, there are some limitations in the stochastic model presented in [1]. The primary one is that only Single-length Lines are used for routing, which is seldom the case in commercial FPGAs. Rich routing resources exist in commercial FPGAs, and as far as we know, no systematic study has been done on the flexibility and performance of these resources in the context of routability prediction. The motivation behind this thesis is to explore the effects of different routing resources on the routability of a design in FPGAs. The model developed can predict the routability of an application after technology mapping but before placement and routing. This helps a designer to estimate the routability of an application on a specific FPGA architecture at an early stage in the design process.
Another aspect of the improved model is the routing prediction speed. Predicting the routability of a large design is very time-consuming for the stochastic models in [1] and [2]. There are more long connections in large designs than in small ones. Routing long interconnections with only Single-length Lines results in more switches in the paths. It is much slower to predict the routability of a connection with a large number of switches than that for a connection with fewer switches. Since we use hierarchical routing resources in our model, the number of switches used in long connections is greatly reduced. Therefore, the routing prediction time is dramatically decreased.

5.2 Architectural Assumptions

Due to the various types of FPGA architectures, routing resources are different from one FPGA to another. Our model is based on a conventional FPGA with hierarchical routing resources. To simplify the model, and in order to keep it mathematically tractable, the following assumptions are made:

- Different types of routing wires are independent of each other. In other words, a connection in an FPGA is routed by only one type of routing wire, that is to say, combinations of routing wires cannot be used. This important assumption for our model will be discussed in more detail in Section 5.3.

- A logic block is implemented as a 4-input LUT with four input pins and two output pins. This assumption is made for comparing our results with those from existing models.

In Xilinx FPGAs, each pin in a logic block can connect to all the routing wires, except Direct Connections, by using the switches in the associated C blocks. Parameter
$F_c$ (connection block flexibility) varies with the different types of wires. Parameter $F_s$ (switch block flexibility) is set to be the same for all the routing resources. Long Lines do not pass through switch matrices, and they cannot make a turn to connect to other Long Lines in a path. We have incorporated these design features in our models.

The following section describes a conventional FPGA routability prediction model based on these assumptions.

5.3 Routability Prediction Model

In the stochastic model presented in [1], routing channels in a symmetrical FPGA are divided into segments that span the length or width of one CLB (Configurable Logic Block), and the channel density follows a Poisson distribution. For FPGAs with different segmented tracks spanning multiple CLBs in the same channel, the channel density may not follow a Poisson distribution. These results are taken from a previous work [5]. However, if the different types of multi-segment lines are formed independently of each other, are separated in a channel, and a connection can choose only one type of wire for routing, we can still apply the same channel density distribution to different routing resources. This is a key assumption used in our model.

To calculate the routability of a design in an FPGA with hierarchical routing resources, we use the divide-and-conquer technique. We will use the subscript type to refer to the routing wires, namely Direction Connections (DC), Single-length Lines (SL), Double-length Lines (DL), Quad-length Lines (QL), and Long Lines (LL). For a design that has a total number of $C_T$ connections, the probabilities of using each type of routing wire for routing, denoted as $P_{DC}$, $P_{SL}$, $P_{DL}$, $P_{QL}$, and $P_{LL}$, are calculated first. Since routing
a connection implies that exactly one type of routing wire is used, we have the following probability relationship:

$$P_{DC} + P_{SL} + P_{DL} + P_{QL} + P_{LL} = 1$$  \hspace{1cm} (5.1)

The number of connections made by each type of routing wire, $C_{\text{type}}$, can be obtained by multiplying Equation 5.1 by the total number of connections:

$$(P_{DC} + P_{SL} + P_{DL} + P_{QL} + P_{LL}) * C_{T} = C_{DC} + C_{SL} + C_{DL} + C_{QL} + C_{LL}$$  \hspace{1cm} (5.2)

Next, the stochastic approach presented in Chapter 3 is used to calculate the routabilities of different routing wires

$$\text{Routability}_{\text{type}} = \frac{1}{C_{\text{type}}} \sum_{i=1}^{C_{\text{type}}} P_{\text{type}}(R_{C_i})$$  \hspace{1cm} (5.3)

where $P_{\text{type}}(R_{C_i})$ is the routability of a connection using a specific type of routing wire. It can be calculated in a way similar to the earlier stochastic model, with some necessary modifications

$$P_{\text{type}}(R_{C_i}) = \sum_{l=l_{\text{min}}}^{l_{\text{max}}} P_{\text{type}}(L_{l}) * P_{\text{type}}(R_{C_i} | L_{l})$$  \hspace{1cm} (5.4)

where $l_{\text{min}}$ and $l_{\text{max}}$ are the length range of connections for a certain type of routing wire. $P_{\text{type}}(R_{C_i} | L_{l})$ is the routability of a connection of length $l$, and the connection is routed by a certain type of wire. $P_{\text{type}}(L_{l})$ is the probability distribution for connections with lengths from $l_{\text{min}}$ to $l_{\text{max}}$ hops for a specific type of routing wire. Finally, the overall routability of the design is given by

$$\text{Routability} = \sum \text{Routability}_{\text{type}}$$  \hspace{1cm} (5.5)
The following section presents a detailed description of how to calculate the routability of a connection that uses one of the different types of routing wires.

5.4 Routability Prediction for Different Routing Wires

With some necessary modifications, the stochastic model of [1] can be used to calculate the routability of conventional FPGAs with a routing hierarchy. A major problem in our prediction model is how to select the type of routing wire used for establishing a connection with length $LC_i$. We present several rules for making a decision:

1. Use Direct Connections and Long Lines whenever possible.
2. For a connection of length $LC_i$, we always choose a routing wire that can establish the connection with the minimum number of S blocks in the path.
3. If more than one type of routing wire satisfies rule No.2, we choose the one with a shorter segment length. This rule is designed to save chip-area.
4. Segmented lines (SL, DL, and QL) have to use at least one S block for routing. C blocks alone cannot route a connection. This rule increases the flexibility of routing a design in FPGAs.

Figure 5.3.1 illustrates these rules. In the example, a connection with a length of 3 hops is routed using different strategies. According to rule No.2, solution "a" and solution b1 are discarded since they use more S blocks (3 and 2, respectively) than other solutions (one S block). Applying rule No.3 to solutions b2 and c1, we find solution b2 is better since it uses shorter segment lines (Double-length Lines), while c1 uses longer segment lines (Quad-length Lines). In solution b2, every segment in a routing channel can connect
Routing between Logic Block A and Logic Block D

Solution a: Using Single-length Lines

Solution b1: Using Double-length Lines

Solution b2: Using Double-length Lines

Solution c1: Using Quad-length Lines

Solution c2: Using Quad-length Lines
(not allowed according to Rule 4)

Legend:

- Logic Block
- Unused S Block
- Used S Block
- By-passed S Block
- Routing Wires
- Connection between Logic Block and C Block

Figure 5.3.1 Different Routing Strategies
to other $F_{sd}$ segments in three directions through an S block. Therefore, in a channel with $W_d$ segments, the number of possible paths for a connection between logic block A and D is approximately $\frac{1}{3} * F_{sd} * W_d$. Compared to solution c2, using an S block in solution b2 provides more flexibility in routing the connection. This corresponds to the requirements of rule No.4.

The following subsections describe the process of calculating the routabilities of connections using different types of routing wires.

5.4.1 Direct Connections

Direct Connections link specific input and output pins of adjacent logic blocks without passing through a C block. A connection of length of one between two logic blocks can select either a Direct Connection or a Single-length Line for routing. The probability that it will use a Direct Connection depends on the source and destination pins used for the connection in logic blocks. Typically, the number of Direct Connections is fewer than that of other routing resources due to the chip-area constraints. The probability of using Direct Connections for routing is related to the connection length distribution, the number of Direct Connections between neighboring logic blocks, and the number of input and output pins in each logic block, and is given by

$$P_{DC} = P(L_1) \frac{N_{DC}}{2 * N_{in} * N_{out}}$$

(5.6)

where $P(L_1)$ is the probability that the length of a connection is one hop, $N_{DC}$ is the total number of Direct Connections between adjacent logic blocks, and $N_{in}$ and $N_{out}$ are the number of input and output pins in each logic block. For the example shown in Figure
4.3.1, \( N_{DC} \) is 2, and \( N_{in} \) and \( N_{out} \) are 4 and 2, respectively. Therefore, any connection with a length of one has a probability of \( \frac{1}{8} * \) \( P(L_1) \) to use Direct Connections and a probability of \( \frac{7}{8} * P(L_1) \) to use Single-length Lines for routing.

Since a Direct Connection is a point-to-point routing that does not go through any C blocks and S blocks, it can always be routed once selected. Therefore, the routability of a Direct Connection is always 1:

\[
\text{Routability}_{DC} = 1
\]

5.4.2 Long Lines

Long Lines are desirable for longer connections since they run across the entire length or width of FPGAs and do not use S blocks for routing. All the straight connections in a routing channel can use Long Lines. However, in order to take full advantage of these, we only use Long Lines to route those straight connections that have lengths equal to or longer than 5 hops and up to the entire length or width of the logic block arrays. According to Equation 3.1, the probability of successfully routing a connection using Long Lines is given by

\[
P_{LL}(R_{C_1} \mid L_1) = P(X_1) * P(X_2 \mid X_1) \quad (5 \leq l \leq N)
\]

where \( N \) is the size of the logic block arrays of FPGAs.

Long Lines overlap Quad-length Lines in routing long connections. As we can see, the premise of using Long Lines is that the source and destination logic blocks share a routing channel in a row or a column and are far apart (at least 5 hops away), as shown
in Figure 5.3.2. The destination logic block (LB) No.4 does not share any routing channels with the source LB. Therefore, it cannot use Long Lines, although it is 7 hops away from the source LB. The destination LB No.1 shares a routing channel with the source LB. However, the distance between the two LBs is too short. Both destination LBs No.2 and No.3 share a routing channel with the source LB and are far away from it (5 and 6 hops, respectively). Therefore, they can use Long Lines for connecting. From this discussion, we can conclude that Long Lines can be used in the following situations:

1) $|LC^x_i|$ (or $|LC^y_i|$) $\geq 5$ and $|LC^y_i|$ (or $|LC^x_i|$) = 0;

2) $|LC^x_i|$ (or $|LC^y_i|$) $\geq 5$ and $|LC^y_i|$ (or $|LC^x_i|$) = 1;

In other words, in order to use Long Lines, the connections have to be straight and have lengths equal to or longer than 5 hops and up to the entire length or width of the
logic block arrays. A connection of length $LC_i$ can be represented by a total of $4*LC_i$ combinations of $LC_i^x$ and $LC_i^y$. If $LC_i$ is 5 hops, only 4 out of $4*5=20$ cases can be routed by Long Lines, that is, when $LC_i^x$ is 0 and $LC_i^y$ is +5 or -5, and vice versa. If $LC_i$ is greater than 5 hops, 12 of the $LC_i^x$ and $LC_i^y$ combinations can be routed by Long Lines. The possible combinations are

- $LC_i^x = \pm LC_i, \quad LC_i^y = 0$;
- $LC_i^y = \pm LC_i, \quad LC_i^x = 0$;
- $LC_i^x = \pm (LC_i - 1), \quad LC_i^y = \pm 1$;
- $LC_i^y = \pm (LC_i - 1), \quad LC_i^x = \pm 1$;

Therefore, for any connection, the probability of using Long Lines is given by

$$P_{LL} = \frac{4}{20} \sum_{l=6}^{N} P(L_l) \cdot \frac{12}{4*l}$$

$$= \frac{1}{5} \sum_{l=6}^{N} P(L_l) \cdot \frac{3}{l} \quad (5.9)$$

where $L_l$ denotes a connection of length $l$ and $N$ is the size of the FPGA. $P(L_l)$ is the connection length distribution function for the whole design. The total number of connections routed by Long Lines is given by:

$$C_{LL} = P_{LL} \cdot C_T \quad (5.10)$$

In order to calculate the probability of using Long Lines for successfully routing connections, denoted as $P_{LL}(R_{C_i})$, we have to know the length distribution for Long Lines, denoted as $P_{LL}(L_l)$. This distribution function is different from $P(L_l)$ since it
represents the length distribution applied only to Long Lines and not to the whole design.

$P_{LL}(L_i)$ can be obtained as the probability of each length in $P_{LL}$, which is given by

$$P_{LL}(L_i) = \begin{cases} \frac{1}{5} \cdot \frac{P(L_5)}{P_{LL}} & (l = 5) \\ \frac{3}{l} \cdot \frac{P(L_l)}{P_{LL}} & (6 \leq l \leq N) \end{cases}$$  (5.11)

The probability of using Long Lines for successfully routing connections, denoted as $Routability_{LL}$, can be obtained by combining Equations 5.8, 5.10, and 5.11:

$$Routability_{LL} = \frac{1}{C_{LL}} \sum_{i=1}^{C_{LL}} \sum_{l=5}^{N} P_{LL}(L_i) \cdot P_{LL}(R_{C_i} | L_i)$$  (5.12)

5.4.3 Single-length Lines

Single-length Lines are desirable for relatively short connections. They can be used to connect two adjacent logic blocks through a C block (connections with a length of one hop) or to connect diagonal logic blocks by using one switch matrix and two C blocks (connections with a length of two hops), as shown in Figure 4.2.2. According to Equation 3.1, the probabilities of routing connections with lengths of one and two hops are

$$P_{SL}(R_{C_i} | L_1) = P(X_1) \cdot P(X_2 | X_1)$$  (5.13)

$$P_{SL}(R_{C_i} | L_2) = P(X_1) \cdot P(S_1 | X_1) \cdot P(X_2 | S_1 \cap X_1)$$  (5.14)
As indicated in Section 5.3.1, Direct Connections can be used for routing some of the connections with a length of one hop. In addition, Double-length Lines can be used for routing half of the connections with a length of 2 hops, that is, these connections in the form of \( |LC^x| = 2 \) and \( |LC^y| = 0 \) or vice versa, as shown in Figure 5.3.3. Therefore, to obtain the number of connections using Single-length Lines for routing, we have to subtract the number of connections that use Direct Connections and Double-length Lines from the total number of connections having lengths of one and two hops. The probability of using Single-length Lines, \( P_{SL} \), can be obtained as follows:

\[
P_{SL} = \left[ P(L_1) - P_{DC} \right] + \left[ P(L_2) - \frac{1}{2} * P(L_2) \right] = (1 - \frac{N_{DC}}{2 * N_{in} * N_{out}}) * P(L_1) + \frac{1}{2} * P(L_2)
\]  

(5.15)

Figure 5.3.3 Connection of Length of 2 Routed by Double-length Lines
where \( P(L_1) \) and \( P(L_2) \) are the probabilities of connections with lengths of one and two hops, respectively. The total number of connections routed by Single-length Lines is given by

\[
C_{SL} = P_{SL} \cdot C_T
\]

The length distribution for Single-length Lines, denoted as \( P_{SL}(L_l) \), can be obtained in a manner similar to \( P_{UL}(L_l) \):

\[
P_{SL}(L_l) = \begin{cases} 
\frac{(1 - \frac{N_{DC}}{2 * N_{in} * N_{out}}) \cdot P(L_1)}{P_{SL}} & (l = 1) \\
\frac{\frac{1}{2} \cdot P(L_2)}{P_{SL}} & (l = 2) 
\end{cases}
\]

Finally, we have the probability of using Single-length Lines for successfully routing connections, denoted as \( Routability_{SL} \), by combining Equations 5.13, 5.14, 5.16, and 5.17:

\[
Routability_{SL} = \frac{1}{C_{SL}} \cdot \sum_{i=1}^{C_{SL}} \left[ (P_{SL}(L_1) \cdot P_{SL}(R_{C_i} \mid L_1) + P_{SL}(L_2) \cdot P_{SL}(R_{C_i} \mid L_2) \right] \quad (5.18)
\]

It is certainly possible to use Single-length Lines to route connections longer than two hops, but it is not desirable. Longer-segment lines can be used to save switches, as discussed in the following sections.

\subsection*{5.4.4 Double-length Lines}

For intermediate-length connections, Double-length Lines are preferred in order to reduce the number of S blocks in the path. They can be used to route connections with
lengths of 2 to 6 hops. Figure 5.3.4 shows all the connections with different combinations of $LC_i^x$ and $LC_i^y$ that can be routed by Double-length Lines. As stated in Section 5.3.3, a connection of length $LC_i$ can be represented by a total of $4*LC_i$ combinations of $LC_i^x$ and $LC_i^y$. The following table lists the ratio of connections routed by Double-length Lines to the total number of all the combinations of $LC_i^x$ and $LC_i^y$ for a given length:

Table 5.1 Ratio of Using Double-length Lines for Routing

<table>
<thead>
<tr>
<th>Length</th>
<th>C_Total *</th>
<th>C_DL **</th>
<th>C_DL/C_Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2*4</td>
<td>1*4</td>
<td>0.5</td>
</tr>
<tr>
<td>3</td>
<td>3*4</td>
<td>3*4</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4*4</td>
<td>3*4</td>
<td>0.75</td>
</tr>
<tr>
<td>5</td>
<td>5*4</td>
<td>2*4</td>
<td>0.4</td>
</tr>
<tr>
<td>6</td>
<td>6*4</td>
<td>1*4</td>
<td>0.17</td>
</tr>
</tbody>
</table>

* Total number of $LC_i^x$ and $LC_i^y$ combinations for a given $LC_i$ value

** Number of $LC_i^x$ and $LC_i^y$ combinations that can use Double-length Lines

The probability of using Double-length Lines for routing, denoted as $P_{DL}$, is given by

$$P_{DL} = 0.5*P(L_2) + P(L_3) + 0.75*P(L_4) + 0.4*P(L_5) + 0.17*P(L_6) \quad (5.19)$$

And the total number of connections routed by Double-length Lines, $C_{DL}$, is given by

$$C_{DL} = P_{DL} * C_T \quad (5.20)$$

The routabilities of connections with different lengths for Double-length Lines, denoted as $P_{DL}(R_{C_i} \mid L_i)$, are given by

$$P_{DL}(R_{C_i} \mid L_i) = P(X_1 \cap S_1 \cap \ldots \cap S_{(l-1)/2} \cap X_2) \quad (5.21)$$
Figure 5.3.4 Connections Routed by Double-length Lines
where \( \lceil (l-1)/2 \rceil \) is the ceiling of \((l - 1) / 2\), which has the following physical interpretation: routing a connection of length \( l \) by Single-length Lines needs \( l-1 \) switch matrices, while with Double-length Lines only half as many switch matrices are needed.

The length distribution for Double-length Lines, denoted as \( P_{DL}(L_d) \), can be obtained in a way similar to that for Single-length Lines. Finally, the probability of using Double-length Lines for successfully routing connections, denoted as \( Routability_{DL} \), is given by

\[
Routability_{DL} = \frac{1}{C_{DL}} \times \sum_{i=1}^{C_{DL}} \sum_{i=2}^{6} P_{DL}(L_i) \times P_{DL}(R_{Ci} | L_i) \quad (5.22)
\]

**5.4.5 Quad-length Lines**

For relatively long connections, Quad-length Lines can be used for routing. Compared to Single-length Lines, Quad-length Lines go into a switch matrix at every fourth logic block, which can reduce the number of switches in the connections by three-quarters (75%). Quad-length Lines are desirable for routing connections in which either \( |LC_{i,x}| \) or \( |LC_{i,y}| \) is equal to or longer than 4 hops and up to the maximum interconnection length. Quad-length Lines overlap with Double-length Lines and Long Lines in a number of connection lengths. One case in connections with a length of 4 hops (where \( |LC_{i,x}| =4 \) and \( |LC_{i,y}| =0 \), or vice versa) and two cases in connections with lengths of 5 hops (where \( |LC_{i,x}| =4 \) and \( |LC_{i,y}| =1 \), or vice versa) use Quad-length Lines for routing. In addition, all connections with lengths longer than 6 hops use Quad-length Lines, except those straight connections which use Long Lines for routing instead. Therefore, the probability of using Quad-length Lines, denoted as \( P_{QL} \), is given by

- 59 -
where $l_{\text{max}}$ is the maximum connection length in a design. The total number of connections routed by Quad-length Lines is

$$C_{\text{QL}} = P_{\text{QL}} \times C_T$$

The routabilities of connections with different lengths using Quad-length Lines, denoted as $P_{\text{QL}}(R_{C_i} \mid L_i)$, are given by

$$P_{\text{QL}}(R_{C_i} \mid L_i) = P(X_1 \cap S_1 \cap S_2 \cap \ldots \cap S_{[(l-1)/4]} \cap X_2)$$

where $[(l-1)/4]$ is the ceiling of $(l - 1) / 4$, which has the following physical interpretation: routing a connection of length $l$ by Single-length Lines needs $l-1$ switch matrices, while with Quad-length Lines, only one-quarter of the switch matrices are used.

The length distribution for Quad-length Lines, denoted as $P_{\text{QL}}(L_i)$, can be obtained in a manner similar to that for Single-length Lines. The probability of using Quad-length Lines for successfully routing connections, denoted as $\text{Routability}_{\text{QL}}$, is given by

$$\text{Routability}_{\text{QL}} = \frac{1}{C_{\text{QL}}} \sum_{i=1}^{C_{\text{QL}}} \sum_{l=4}^{l_{\text{max}}} P_{\text{QL}}(L_i) \times P_{\text{QL}}(R_{C_i} \mid L_i)$$

### 5.5 Switches-used-per-tile Measurement

In Chapter 4, we introduced the "switches-per-tile" measurement for FPGAs. However, this measurement only reflects the switch count for FPGAs, not switch consumption by the designs that are routed on the FPGAs. Typically, a design uses only part of the switches in a tile. To determine how many switches are actually used by a design, we have devised the "switches-used-per-tile" concept. This measurement more accurately reflects the switch consumption of a design.
For any Xilinx SRAM-based FPGA, we can obtain the total number of routing resources per tile from its routing architecture by checking the data book. For example, Xilinx XC4000X FPGAs have 2 Direct Connections, 16 Single-length Lines, 8 Double-length Lines, 24 Quad-length Lines, and 16 Long Lines per tile. These are the maximum routing resources available for each tile and are denoted as $C_{\text{max}}^{\text{type}}$. For any design that is routed in these FPGAs, we can obtain the total number of connections routed by each type of routing wire, as shown in Equation 5.2. By dividing these numbers by the total number of logic blocks used in the design, we determine the average routing resources used per tile. Therefore, the percentage of a routing resource used per tile is given by

$$\frac{C_{\text{type}}}{\omega} \frac{C_{\text{type}}}{C_{\text{max}}^{\text{type}}} = \frac{C_{\text{type}}}{\omega * C_{\text{max}}^{\text{type}}}$$

(5.27)

where $\omega$ is the total number of logic blocks used by a design. In Chapter 4, we listed the equations for calculating the number of switches per tile for each routing wire. Therefore, by multiplying Equation 5.27 and each of Equations 4.5, 4.7, 4.9, 4.10, or 4.11, we get the average number of switches used for a specific routing wire in a tile. Finally, by adding these results, we obtain the total number of switches used in a tile:

$$\# \text{Switches-used-per-tile} = \frac{C_{\text{DC}}}{\omega * C_{\text{max}}^{\text{DC}}} \times 8 + \frac{C_{\text{SL}}}{\omega * C_{\text{max}}^{\text{SL}}} \times (14 * F_{o} + 2 * F_{s} * W_{s}) +$$

$$\frac{C_{\text{DL}}}{\omega * C_{\text{max}}^{\text{DL}}} \times (14 * F_{\text{DL}} * F_{s} * W_{d}) + \frac{C_{\text{QL}}}{\omega * C_{\text{max}}^{\text{QL}}} \times (14 * F_{\text{QL}} * \frac{1}{2} * F_{s} * W_{q}) +$$

$$\frac{C_{\text{LQ}}}{\omega * C_{\text{max}}^{\text{LQ}}} \times (14 * F_{o}$$

(5.28)
The improved stochastic model developed in this chapter can be used to predict the routability of conventional FPGAs with hierarchical routing resource. Connections with different lengths can choose appropriate type of wires for routing to reduce signal delays.

In the next chapter, we apply the routing hierarchy to Hierarchical FPGAs (HFPGAs). Different properties of the HFPGA routing architecture are investigated.
Chapter 6
Routability Prediction for Hierarchical FPGAs Using Hierarchical Routing Resources

In this chapter, we apply the routability prediction model presented in Chapter 5 to Hierarchical FPGAs (HFPGAs) [2]. Both the logic blocks and the routing resources in an HPGA are structured in a hierarchy. Section 6.1 discusses the motivation for developing this architecture. Section 6.2 introduces an HPGA model with a routing hierarchy. Important parameters for the HPGA model are discussed in Section 6.3. In Section 6.4, a routability prediction model applied to the HFPGAs is presented.

6.1 Motivation

As discussed in Chapter 3, HPGA architecture has the potential to increase the area and speed performance of FPGAs. This advantage arises mainly due to the hierarchical arrangement of logic blocks and routing resources. For the model presented in [2], the local and global routing wires provide a routing hierarchy for the whole FPGA to a certain degree. However, only Single-length Lines are used at each level. This basic model can be further improved. We can extend the HPGA model by combining the logic block hierarchy and the routing resource hierarchy. The enhanced model can fully take advantage of the hierarchical structures. As a result, the number of switches used for routing can be further reduced. In return, the speed of designs implemented in HFPGAs
can be increased. In a manner similar to the conventional FPGA model, the routability prediction process can be done before placement-and-routing, which can help a designer to choose an appropriate HFPGA structure for a design at a fairly early stage in the design cycle.

6.2 HFPGA Model

Our HFPGA model is based on the work reported in [2] and [4]. Logic blocks and routing resources are grouped together to form a multiple-level hierarchy. At each level $k$, a basic element consists of a group of $K=k \times m$ elements from level $k-1$. $K$ is called the cluster size and is fixed for all levels in our HFPGA model. This grouping is expanded recursively until the whole FPGA is covered and becomes one $m^{th}$-level element. Such an HFPGA structure is referred to as a $K-M$ HFPGA. The grouping process is shown in Figure 6.2.1. Hierarchical routing structure is also used in this model. As far as a basic element at each level is concerned, the routing resources can be divided into two parts:
• Local routing resources: they are provided inside each element block to route local connections within the block. As they did in conventional FPGAs, the routing resources are formed in a hierarchy.

• Global routing resources: these are used to route connections amongst the blocks at the same level. These blocks, together with the global routing resources, form a higher-level block.

As we can see, the local routing resources of a block at \(k^{th}\) level are actually the global routing resources for \((k-1)^{th}\) level blocks. An illustration of the model is shown in Figure 6.2.2.

![Hierarchical FPGA with a Routing Hierarchy](image)

**Figure 6.2.2 Hierarchical FPGA with a Routing Hierarchy**

Previous research in [4] has indicated that the use of direct connections in HFPGAs could reduce the routing switch requirements, as well as provide a further improvement in timing. In this thesis, we use our model to explore the properties of all the hierarchical routing resources. We apply the routability prediction model developed
in Chapter 5 to Hierarchical FPGAs. Important parameters required in the enhanced HFPGA model are discussed in the following section.

6.3 Parameters for the Enhanced HFPGA Model

In our proposed HFPGA model, we assume that a design is already technology-mapped into a multi-level hierarchy. Since the structure of each element in our HFPGA is the same as that of a conventional FPGA, we can apply the conventional FPGA model individually to each level. The routability of the whole design can be obtained by adding up the routabilities at all levels. Due to the hierarchical structure, the following important parameters used for routability prediction are different for each level:

- $n_k$: Total number of connections at the $k^{th}$ level
- $l_d(L_i)$: Probability distribution function at the $k^{th}$ level for interconnections of length $i$

These parameters cannot be obtained directly before placement. The following subsections discuss how to estimate these parameters.

6.3.1 Number of Connections per Level

In order to estimate the number of connections at each level, we have to use Rent's rule [23]. In Chapter 3, we indicated that Rent's rule provides a relationship between the number of elementary blocks $B$ in a module of a partitioned design, and the number of external pins $P$ in the module. The relationship is given by

$$P = C \cdot B^r \quad (0 < r < 1)$$

(6.1)

where $C$ is the average number of interconnections per elementary block, and $r$ is called the Rent exponent.
For a design with a total of \( G \) gates, we divide \( G \) into modules of size \( B \). The total number of pins for sub-designs of size \( B \) is then given by [8]

\[
P_{\text{total}} = C \cdot B^r \cdot \frac{G}{B} = C \cdot G \cdot B^{r-1}
\]

(6.2)

For a \( K\)-M structure HFPGA, the total number of CLBs is \( k^M \). The total number of CLBs per element at level \( k \) can be calculated as \( k^k \). According to Equation 6.2, the total number of pins at the \( k^{th} \) level is:

\[
P_i(k) = C \cdot K^M \cdot K^{k(r-1)}
\]

(6.3)

Considering a module at hierarchical level \( k \), some of the pins inside the module are used for routing local interconnections amongst the sub-elementary blocks; the rest of the pins are used for global routing which consists of the interconnections leaving the module and going into other elements at the same level. These global connections can be viewed as local interconnections at the \((k+1)^{th}\) level. We have to extract the number of pins used for global routing from the total number of pins at the \( k^{th} \) level to obtain the number of pins required for local routing. The equation is given by

\[
P(k) = P_i(k) - P_i(k + 1) = C \cdot K^M \cdot K^{k(r-1)}(1 - K^{r-1})
\]

(6.4)

The total number of interconnections at each level, \( n_k \), can be found by assuming that, for point-to-point interconnections, \( n_k \) equals \( \frac{P(k)}{2} \):

\[
n_k = \frac{1}{2} C \cdot K^M \cdot K^{k(r-1)} \cdot (1 - K^{r-1})
\]

(6.5)
6.3.2 Interconnection Length Distribution per Level

Interconnection length distribution is a key parameter in our prediction model and has to be obtained from each level. A technique developed by Donath [6,7] can be used for estimating the interconnection length distribution. In Donath's technique, a design is partitioned hierarchically into sub-designs. For reasons of symmetry, there are four sub-designs at each hierarchical level in a two-dimensional placement procedure. Each sub-design at a certain hierarchical level itself consists of four sub-designs at the next lower level of hierarchy. This partition is done recursively, as shown in Figure 6.3.1.

![Recursive Partitioning Scheme of a Design and the Physical Architecture for a Two-dimensional Placement](image)

Figure 6.3.1 Recursive Partitioning Scheme of a Design and the Physical Architecture for a Two-dimensional Placement [10].

At hierarchical level $k$, the average interconnection length between two sub-designs depends on the positions of those sub-designs in the physical architecture. In a two-dimensional placement, only two different combination classes are possible: either the sub-designs are adjacent or they are diagonally opposed, as shown in Figure 6.3.2. In this figure, variable $\lambda_k$ is defined as the side length of the sub-design at level $k$. The maximum interconnection length at $k^{th}$ level, which is the distance between two diagonal
elements at level $k$, can be calculated as $4^k \lambda_k$. The interconnection length distribution has been calculated and presented by Cotter and Christie in [10]. A normalized connection length distribution for a two-dimensional Manhattan grid at each level is given by

$$P_k(l) = \begin{cases} 
\frac{-l^3 + 4\lambda_k l^2 + l}{6\lambda_k^4}, & \text{for } 0 \leq l \leq \lambda_k \\
\frac{5l^3 - 36\lambda_k l^2 + (72\lambda_k^2 - 5)l - 32\lambda_k^3 + 8\lambda_k}{18\lambda_k^4}, & \text{for } \lambda_k \leq l \leq 2\lambda_k \\
\frac{-l^3 + 12\lambda_k l^2 - (48\lambda_k^2 - 1)l + 64\lambda_k^3 - 4\lambda_k}{18\lambda_k^4}, & \text{for } 2\lambda_k \leq l \leq 4\lambda_k \\
0, & \text{otherwise}
\end{cases} \quad (6.6)$$

where $P_k(l)$ is the interconnection length distribution at level $k$, $l$ is the interconnection length, and $\lambda_k$ is the side length of the sub-designs at the $k^{th}$ level. However, the placement schema in Donath's model is a random process, which results in deviations. Stroobandt improved Donath's technique by considering an optimal placement schema and the behavior of the global probability distribution in combination with the local distribution in a continuous form [8]. Since FPGAs also can be treated as two-
dimensional arrays, and Stroobandt's method can estimate the interconnection length distribution more accurately than Donath's, we have adopted Stroobandt's method.

In Stroobandt's method, a design is still recursively partitioned into 2x2 sub-designs to form a multi-level hierarchy. This is exactly the same architecture as that of our 4-M HFPGAs. Structural distribution and probability distribution are introduced for deriving the length distribution function. Structural distribution is the interconnection length distribution that only depends on the physical lengths of the interconnections. The distribution $P_k(l)$ presented by Cotter and Christie is the Structural distribution. We can also assign to each length the probability that the interconnection corresponding to that length would be laid out at a specific position in a real placement procedure. If the design is optimally placed in the physical architecture, the probability of a short connection will outnumber the probability of longer connections. In return, the interconnections between adjacent sub-designs should get a higher probability in the distribution. The probabilities are illustrated in Figure 6.3.3.

Figure 6.3.3 The Probability of Placement of an Interconnection (darker zones have higher probability)
This leads to *Probability distribution*, which is denoted as \( f(l) \). This distribution is assigned to each length for the probability that the interconnection corresponding to that length would be laid out at a specific position.

The continuous form of the structural distribution function \( P_k(l) \) is given by

\[
P_k(l) = \begin{cases} 
\frac{-l^3 + 4\lambda_k l^2}{6\lambda_k^4}, & \text{for } 0 \leq l \leq \lambda_k \\
\frac{5l^3 - 36\lambda_k l^2 + 72\lambda_k^2 l - 32\lambda_k^3}{18\lambda_k^4}, & \text{for } \lambda_k \leq l \leq 2\lambda_k \\
\frac{-l^3 + 12\lambda_k l^2 - 48\lambda_k^2 l + 64\lambda_k^3}{18\lambda_k^4}, & \text{for } 2\lambda_k \leq l \leq 4\lambda_k \\
0, & \text{otherwise}
\end{cases}
\]  

where \( l \) is the connection length, and \( \lambda_k \) is the side length of the sub-designs at the \( k \)th level and has the value of \( 2^{k-1} \) in terms of logic blocks in our HFPGA model. Function \( f(l) \) is given by

\[
f(l) = l^{2r-3}
\]  

where \( r \) is the Rent exponent.

The interconnection length distribution function at \( k \)th level, \( l_k(L_i) \), can be obtained by multiplying the structural distribution \( P_k(l) \) with the probability distribution \( f(l) \). However, since the structural distribution function \( P_k(l) \) is in continuous form, we have to obtain the result by integration. A normalized length distribution function \( l_k(L_i) \) is given by

\[
l_k(L_i) = \frac{\int_{i-1}^{i} P_k(i) \ast f(i) \, di}{\int_{0}^{4\lambda} P_k(i) \ast f(i) \, di} \quad (1 \leq i \leq 4\lambda)
\]
Using Equations 6.7, 6.8, and 6.9 we can derive the probabilities of connections of any length at each level. However, Equation 6.9 is only valid for Hierarchical FPGAs that have a cluster size of 4, which represents 4 sub-designs at each hierarchical level. For HFPGAs with any cluster size of $K$, $\lambda_k$ has to be changed accordingly; $K$ and $\lambda_k$ have the following relationship

$$\lambda_k = \sqrt[4]{K/4} (\sqrt{K})^{k-1} = \frac{1}{2} K^{\frac{k}{2}} \tag{6.10}$$

### 6.4 Routability Prediction Model for HFPGAs

As we showed in Section 6.3, Rent exponent $r$ plays a significant role in our parameter calculations. Typically, $r$ is unknown. We have to estimate it from a design’s netlist file. For any given application, we can first technology-map it into a certain type of logic blocks of a target HFPGA. Next, we may extract the following data from the netlist file of the mapped design:

- **#LBs**: total number of logic blocks in the design
- **#Pins**: total number of pins, including internal and external pins, used in the design
- **#IOs**: total number of external I/O pins used in the design

The average number of interconnections per logic block, denoted as $C$, is given by

$$C = \frac{\#Pin - \#IOs}{\#LBs} \tag{6.11}$$

According to Rent’s rule presented in Equation 6.1, we can extract the Rent exponent $r$ as follows:
Now we have all the necessary parameters. We can apply the routability prediction model developed in Chapter 5 to each level of our enhanced HFPGA model. The routability of the whole design in an HFPGA is given by

\[
Routability = \frac{M}{\sum_{k=1}^{M} R_k} \quad (6.13)
\]

where \(M\) is the number of levels in the hierarchy, and \(R_k\) is the routability at level \(k\), which is given by

\[
R_k = \sum Routability_{type} \quad (6.14)
\]

In the next chapter, we present experimental results that validate the proposed models in Chapter 5 and Chapter 6. The flexibility and performance of hierarchical routing resources are investigated as well.
Chapter 7
Experimental Studies

This chapter describes the experimental studies undertaken to evaluate the proposed conventional FPGA and hierarchical FPGA routability models developed in Chapter 5 and Chapter 6, respectively. Microelectronics Center of North Carolina (MCNC) benchmark circuits are used to compare the switch consumption in the proposed models with those in the stochastic models presented in [1] and [2]. A CAD software tool is used to verify the results. In addition, the impact of different types of routing resources on the routability of designs is also explored. The outcome of these experiments provides a more realistic picture of the routing problems in FPGAs with hierarchical routing resources.

7.1 Introduction

The models described in Chapter 5 and Chapter 6 have been implemented in Visual C++ and run on a Pentium 166 PC. The program can predict the routabilities of FPGAs with different types of routing structures, which include symmetrical FPGAs, conventional FPGAs with a routing hierarchy, and HFPGAs with a routing hierarchy. The time required for prediction is displayed at the end of program execution.

The experiments were conducted in three steps:

1. The stochastic model presented in [1] was used first to predict the routabilities of the selected benchmark circuits. Then, the same set of benchmark circuits
was fed to the model developed in Chapter 5 for routability prediction. The results from both models were compared to each other in terms of switch consumption and prediction time. The effects of different types of routing wires on the routability of circuits were studied. Furthermore, other prediction issues, including connection length probability distribution and worst-case delay, were also analyzed.

2. Another group of MCNC benchmark circuits was routed by Xilinx Alliance software on XC4000XL series FPGAs. We applied the routing parameters of selected FPGAs to both our conventional FPGA model and the stochastic model of [1]. Finally, these results were compared. The prediction time complexity is also discussed.

3. A set of large benchmark circuits was routed by using the HFPGA model developed in Chapter 6. Results were compared with the HFPGA model presented in [2]. The effect of different types of routing wires in each hierarchy level was studied.

These experiments revealed that fewer switches are used in our proposed models. In addition, the routability prediction time is dramatically reduced.

7.2 Implementation Process

All the circuits used in our experiments are taken from the MCNC benchmark suite.

The benchmark circuits used in step 1 are shown in Table 7.1. "r" is the Rent exponent estimated by using Equation 6.12. The process of predicting routability on an FPGA consists of the following sub-steps [11], as shown in Figure 7.2.1.
Table 7.1 MCNC Benchmark Circuits Used for Experiments in Step 1

<table>
<thead>
<tr>
<th>Circuits</th>
<th>CLBs</th>
<th>Total Pins</th>
<th>I/O Pins</th>
<th>#Connections</th>
<th>Estimated $r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>C432</td>
<td>74</td>
<td>335</td>
<td>43</td>
<td>261</td>
<td>0.55</td>
</tr>
<tr>
<td>Apex7</td>
<td>103</td>
<td>474</td>
<td>86</td>
<td>371</td>
<td>0.67</td>
</tr>
<tr>
<td>C499</td>
<td>114</td>
<td>570</td>
<td>73</td>
<td>456</td>
<td>0.60</td>
</tr>
<tr>
<td>f5</td>
<td>125</td>
<td>566</td>
<td>199</td>
<td>441</td>
<td>0.87</td>
</tr>
<tr>
<td>C880</td>
<td>142</td>
<td>663</td>
<td>86</td>
<td>521</td>
<td>0.62</td>
</tr>
<tr>
<td>Example2</td>
<td>147</td>
<td>651</td>
<td>151</td>
<td>504</td>
<td>0.76</td>
</tr>
<tr>
<td>Alu2</td>
<td>235</td>
<td>1055</td>
<td>16</td>
<td>820</td>
<td>0.24</td>
</tr>
<tr>
<td>X4</td>
<td>290</td>
<td>1336</td>
<td>165</td>
<td>1046</td>
<td>0.65</td>
</tr>
<tr>
<td>X1</td>
<td>582</td>
<td>2808</td>
<td>86</td>
<td>2226</td>
<td>0.46</td>
</tr>
<tr>
<td>Des</td>
<td>1591</td>
<td>7456</td>
<td>501</td>
<td>5865</td>
<td>0.64</td>
</tr>
<tr>
<td>Apex2</td>
<td>1878</td>
<td>8567</td>
<td>41</td>
<td>6689</td>
<td>0.29</td>
</tr>
</tbody>
</table>

Figure 7.2.1 CAD Process for Routability Prediction
I. Logic Optimization: SIS[12] is used to perform various optimizations of Boolean expressions representing a logic netlist.

II. Technology Mapping: Technology mapping is performed by FlowMap [13]. The optimized Boolean expressions are mapped to a netlist of 4-input LUT logic blocks and flip-flops.

III. Packing: The mapped LUTs and flip-flops are packed into target FPGA logic blocks by using VPack [11], a packing utility developed at the University of Toronto.

IV. Routability Prediction: Predictions are made using the models developed in [1] and in this thesis.

In Step 2, another group of MCNC benchmarks in "XNF" netlist format was routed by Xilinx Alliance software. Xilinx XC4000XL series FPGAs were used as the target FPGAs on which these circuits were routed. After mapping, some useful data is extracted from the report files and is shown in Table 7.2. Two circuits in Table 7.1 (C499 and C880) are also selected in Table 7.2. However, they have different mapping data. The circuits in Table 7.2 have fewer CLBs and connections. The reason is that in Step 1, we assume that each CLB has one 4-input LUT, while in Step 2, XILINX XC4000XL series FPGAs have two 4-input LUTs per CLB, which can implement more logic functions. In addition, Xilinx Alliance CAD tools are a set of commercial software programs, which may yield better mapping results. All these factors likely result in the reduced number of CLBs and connections in Table 7.2. The CAD process for implementing a circuit using Xilinx Alliance software is shown in Figure 7.2.2.
Table 7.2: MCNC Benchmark Circuits Used for Experiments in Step 2

<table>
<thead>
<tr>
<th>Circuits</th>
<th>CLBs</th>
<th>Total Pins</th>
<th>I/O Pins</th>
<th>#Connections</th>
<th>Estimated $r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>C432</td>
<td>30</td>
<td>334</td>
<td>43</td>
<td>242</td>
<td>0.44</td>
</tr>
<tr>
<td>C499</td>
<td>39</td>
<td>392</td>
<td>73</td>
<td>281</td>
<td>0.60</td>
</tr>
<tr>
<td>C880</td>
<td>58</td>
<td>591</td>
<td>86</td>
<td>430</td>
<td>0.56</td>
</tr>
<tr>
<td>C1355</td>
<td>38</td>
<td>383</td>
<td>73</td>
<td>278</td>
<td>0.60</td>
</tr>
<tr>
<td>C1908</td>
<td>75</td>
<td>740</td>
<td>58</td>
<td>572</td>
<td>0.43</td>
</tr>
<tr>
<td>C2670</td>
<td>127</td>
<td>1320</td>
<td>221</td>
<td>936</td>
<td>0.67</td>
</tr>
<tr>
<td>C3540</td>
<td>199</td>
<td>1841</td>
<td>75</td>
<td>1420</td>
<td>0.40</td>
</tr>
<tr>
<td>C5315</td>
<td>283</td>
<td>2748</td>
<td>301</td>
<td>2065</td>
<td>0.63</td>
</tr>
<tr>
<td>C7552</td>
<td>417</td>
<td>3641</td>
<td>313</td>
<td>2674</td>
<td>0.61</td>
</tr>
<tr>
<td>C6288</td>
<td>478</td>
<td>4072</td>
<td>64</td>
<td>3087</td>
<td>0.33</td>
</tr>
</tbody>
</table>

In Step 3, large MCNC benchmark circuits used in [2], which are shown in Table 7.3, were run using our HFPGA routability prediction program. These results were compared with the results from the HFPGA model presented in [2]. The CAD process of
predicting routability on an HFPGA is the same as the process in Step 1, except that the predictions were run on the HFPGA models.

Table 7.3 MCNC Benchmark Circuits Used for Experiments in Step 3

<table>
<thead>
<tr>
<th>Circuits</th>
<th>CLBs</th>
<th>Total Pins</th>
<th>I/O Pins</th>
<th>#Connections</th>
<th>Estimated $r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elliptic</td>
<td>3604</td>
<td>17246</td>
<td>245</td>
<td>12520</td>
<td>0.48</td>
</tr>
<tr>
<td>Frisc</td>
<td>3556</td>
<td>17098</td>
<td>136</td>
<td>12565</td>
<td>0.41</td>
</tr>
<tr>
<td>Spla</td>
<td>3690</td>
<td>17452</td>
<td>62</td>
<td>13762</td>
<td>0.31</td>
</tr>
<tr>
<td>S298</td>
<td>1931</td>
<td>8884</td>
<td>10</td>
<td>6945</td>
<td>0.10</td>
</tr>
<tr>
<td>Apex2</td>
<td>1878</td>
<td>8567</td>
<td>41</td>
<td>6689</td>
<td>0.29</td>
</tr>
</tbody>
</table>

7.3 Performance of Conventional FPGA Model with a Routing Hierarchy

The purpose of the first two experiments is to evaluate the performance of our proposed conventional FPGA model. We compare the stochastic model developed by Brown in [1] with our conventional FPGA model presented in Chapter 5, in terms of switch consumption and prediction time. Furthermore, we use commercial CAD software tools to validate our model.

7.3.1 Comparison with Brown's Model

We apply the eleven MCNC benchmark circuits (Table 7.1) on both Brown's model and our model. These circuits are deemed routable if the routabilities reach 99%. Different sets of routing parameters are used and the set with the minimal number of switches-per-tile is recorded in Table 7.4. The switches-per-tile and the switches-used-per-tile are also calculated, as well as the processing time. The results are shown in Table 7.5.
Table 7.4 Routing Parameters for Brown's Model and Our Model

<table>
<thead>
<tr>
<th>Benchmark Circuits</th>
<th>Brown's Model</th>
<th>Our Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single-length Lines</td>
<td>Double-length Lines</td>
</tr>
<tr>
<td></td>
<td>$W$</td>
<td>$F_x$</td>
</tr>
<tr>
<td>C432</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>Apex7</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>C499</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>I5</td>
<td>14</td>
<td>10</td>
</tr>
<tr>
<td>C880</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>Example2</td>
<td>14</td>
<td>10</td>
</tr>
<tr>
<td>Alu2</td>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>X4</td>
<td>14</td>
<td>12</td>
</tr>
<tr>
<td>X1</td>
<td>10</td>
<td>8</td>
</tr>
<tr>
<td>Des</td>
<td>14</td>
<td>12</td>
</tr>
<tr>
<td>Apex2</td>
<td>10</td>
<td>8</td>
</tr>
</tbody>
</table>

* Parameter $F_s$ is the same for all the routing wires in our model.

Table 7.5 Switch Consumption and Prediction Time Comparison

<table>
<thead>
<tr>
<th>Benchmark Circuits</th>
<th>Brown's Model</th>
<th>Our Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total Switches-per-tile</td>
<td>Used Switches-per-tile</td>
</tr>
<tr>
<td>C432</td>
<td>148</td>
<td>62</td>
</tr>
<tr>
<td>Apex7</td>
<td>172</td>
<td>75</td>
</tr>
<tr>
<td>C499</td>
<td>192</td>
<td>84</td>
</tr>
<tr>
<td>I5</td>
<td>252</td>
<td>105</td>
</tr>
<tr>
<td>C880</td>
<td>192</td>
<td>83</td>
</tr>
<tr>
<td>Example2</td>
<td>252</td>
<td>91</td>
</tr>
<tr>
<td>Alu2</td>
<td>132</td>
<td>34</td>
</tr>
<tr>
<td>X4</td>
<td>280</td>
<td>105</td>
</tr>
<tr>
<td>X1</td>
<td>192</td>
<td>80</td>
</tr>
<tr>
<td>Des</td>
<td>308</td>
<td>162</td>
</tr>
<tr>
<td>Apex2</td>
<td>172</td>
<td>48</td>
</tr>
</tbody>
</table>

* Based on a Pentium 166 processor with 32 MB memory
The switch consumption for both models is shown in Figure 7.3.1. The figure indicates the total number of switches-per-tile in our model is higher than Brown's model. This discrepancy occurs because Brown's model is limited and it only considers Single-length Lines, which results in a low switch count per tile. Each track has a number of switches in the C and S block. Therefore, more switches exist in each tile in our model due to different types of routing wires. The excess tracks can provide more routing paths for connections, which increases the flexibility of routing. The number of switches used per tile in our model is much lower than that in Brown's model. On average, our model uses about 56% fewer switches than Brown's model does. This result illustrates that, with hierarchical routing resources, an FPGA can greatly reduce the number of switches used.
for routing. Part of the connections between adjacent logic blocks can take advantage of Direct Connections by skipping switches in the C and S blocks. The rest of the connections can be routed with multi-segment routing wires and Long Lines to bypass some of the S blocks. Therefore, the number of switches required to successfully route a circuit is greatly reduced. In turn, the speed of the circuit routed on the target FPGA is increased. This improvement makes it possible to implement circuits with critical time constraints on FPGAs.

The routability prediction times are also compared and the results are shown in Figure 7.3.2a and Figure 7.3.2b. As stated before, it is more time-consuming to predict the routability of a long connection passing through a number of switches than to predict the same for a connection passing through fewer switches. Since hierarchical routing resources are used in our model, the number of switches in long connections is greatly reduced. Therefore, the routing prediction time is cut down dramatically.

![Figure 7.3.2a Prediction Time Comparison](image)

Figure 7.3.2a Prediction Time Comparison
The relationship between the prediction times for the two models (Brown's symmetrical FPGA model and our conventional FPGA model) is plotted in Figure 7.3.3.
The figure indicates that the relationship follows a cubic function perfectly. With the help of SPSS, a statistical analysis software, the cubic function can be expressed as follows:

\[ Y = | -0.16X^3 + 4.72X^2 + 2.33X + 1.18 | \]  

(7.1)

where \( X \) is the prediction time for our conventional FPGA model and \( Y \) is the prediction time for Brown's model. The absolute value sign "| |" is used to obtain a positive value for time. Relation 7.1 indicates that our prediction model is far more efficient than Brown's model.

From the preceding comparisons, we can conclude that the performance of our conventional FPGA model is better than Brown's symmetrical FPGA model in terms of flexibility, switch consumption, and speed. This enhanced performance is due to the use of routing hierarchy, whereby each connection is mapped to an appropriate wire type depending on its length.

**7.3.2 Effects of Different Routing Wires**

Five different types of routing wires are utilized in our model. The number of connections routed by each type of wire in proportion to the total number of connections for a circuit (Des) is shown in Figure 7.3.4. The figure shows that the longer the segment that a routing wire has, the smaller the portion of the connections that it routes. More than half of the connections are routed by Single-length Lines in this example. 21% of the connections are routed by Double-length Lines. Connections routed by Quad-length Lines and Long Lines are only 15% and 5%, respectively. Similar results are obtained from other circuits. Therefore, connections in a circuit are not uniformly routed over all
types of wires. More short-segment wires are used than long-segment ones. This conclusion is supported by the circuit’s connection length probability distribution, which is shown in Figure 7.3.5. This figure shows the probability of a connection of specific length in circuit Des. For simplicity, only the first 20 length probabilities are presented.
As indicated in the figure, the probabilities of short connections (connections with lengths between 1 to 3 hops) are much higher than those for long connections (connections with lengths equal to or longer than 4 hops). With the increase in connection length, the associated probabilities drop rapidly. In fact, for a connection of length shorter than 10 hops, the accumulated probability is nearly 90%. The distribution reveals that for a design implemented on an FPGA, most of the connections are short. The reason is that, during the placement step, the target-mapped functions that require connections are always assigned to the logic blocks that are as near to each other as possible within an FPGA. This placement scheme makes the routing process relatively easy to implement. Therefore, after placement, the required connections among the logic blocks are relatively short. As stated in Chapter 4, choosing an appropriate type of wire for routing a connection depends on the length of the connection. As a result, more connections are routed by short-segment wires than by long-segment wires in the circuit. The only exception is Direct Connection. Due to its constraint and limited number, only a small portion of connections is routed by it. Although the share of long connections is very small, a large number of switches would be required if they were routed by Single-length Lines. Therefore, using long-segment routing wires like Double-length Lines and Quad-length Lines for these connections can greatly reduce the number of switches needed in the paths. The more long connections in a design, the more switches can be saved.

Although more Single-length Lines are used than longer-segment wires for routing, the channel width \((W)\) for Single-length Lines is equal to or less than Double-length Lines and Quad-length Lines, which means that there are more longer-segment wires in a tile. On average, Single-length Lines occupy only about 21% of the total
channel width (including all types of wires) in our model. The reason is that routing short connections by Single-length Lines uses at most one S block. With the appropriate number of tracks in the routing channel, the probability of successfully making a connection through the one S block is very high. However, routing a long connection may require more than one S block in the path. Such a connection can be successfully routed only if all the routing channels and S blocks along the path are available. Any track or switch occupied by previous connections may cause a conflict in the present routing. The connection’s routability decreases with the increase of its length. To avoid the conflict, we have to either increase the switch block flexibility $F_s$, or add more tracks to a channel. Moreover, since $F_s$ is fixed in our model for all routing wires, we have to expand the channel width for long-segment routing wires. Consequently, in designing an FPGA routing architecture, higher channel widths are required for Double-length Lines and Quad-length Lines. There is a relationship between the total channel width and the channel width required for an individual type of routing wire to obtain the best routability and the least switches-per-tile. This relationship, however, will be left for future study.

7.3.3 Length Probability Distribution Function

The length probability distribution function is very important in our model since it determines the number of connections to be routed by each type of wire. The distribution function is estimated after technology mapping, but before placement. In the stochastic model presented in [1], the distribution function is assumed to be geometric. In our model, we adopt Donath’s wire length distribution [7]. This probability distribution

87
function is a theoretical derivation from Rent's rule and is based on experimental study. The function is given by

\[ f_k = \frac{k^{2r-3}}{L \sum_{k=1}^{L} k^{2r-3}} \]  

(7.2)

where \( k \) is the connection length, \( r \) is the Rent exponent, and \( L \) is an upper bound on \( k \). The accuracy of Donath’s distribution function can be verified by comparing its output to the real length distribution in a design after placement. Figure 7.3.6 shows the result of such a comparison. The design is that of a hand calculator logic chip given by Donath [7]. Its Rent exponent was found to be 0.59 in [7]. The geometric distribution is also shown in the figure. For the sake of simplicity, only the first ten length distributions are shown.

![Figure 7.3.6 Length Probability Distribution Comparison](image)

As seen in the figure, Donath’s length probability distribution function corresponds more closely to the real length distribution than does the geometric distribution. Although Donath’s distribution function was developed for logic circuit chips, it can also be applied to the FPGAs considered here, because both the placement
procedures of the devices are on a two-dimensional square Manhattan grid and the
definitions of the routing channels are similar.

7.3.4 Worst-case Delay

For our conventional FPGA model, the longest path is the connection that links
the upper-left logic block to the bottom-right logic block. The delay encountered in this
path represents the worst-case delay. The connection is routed by Quad-length Lines and
the number of S blocks used in the path can be calculated using the following formula:

\[
\# S \text{ Blocks}_{\text{worst}} = 2 \cdot \left\lceil \frac{N}{4} \right\rceil - 1
\]  

(7.3)

where \( N \) is the matrix size of the symmetrical FPGA and \( \lceil \cdot \rceil \) refers to the ceiling of the
result. If the delay in an S block is denoted as \( \text{Delay}_{S\text{-block}} \), then the worst-case delay can
be calculated as

\[
\text{Delay}_{\text{worst}} = \text{Delay}_{S\text{-block}} \cdot \# S \text{ Blocks}_{\text{worst}}
\]  

(7.4)

7.3.5 Using CAD Software Tools for Validation

To further validate our model, ten MCNC benchmark circuits were routed using
Xilinx Alliance software on XC4020XL and XC4062XL FPGAs. All the circuits
achieved 100% routability. The actual routing times are shown in Table 7.7a. Important
mapping data were extracted from the report files produced by Alliance software (Table
7.2). We then applied the routing parameters of the XC4000XL series of FPGAs, shown
in Table 7.6, and the extracted mapping data to both Brown’s model and our model for
routability prediction. The results are shown in Table 7.7b and Table 7.7c. Since the
Table 7.6 Routing Parameters for Xilinx XC4000XL Series FPGAs

<table>
<thead>
<tr>
<th>Single-length Lines</th>
<th>Double-length Lines</th>
<th>Quad-length Lines</th>
<th>Long Lines</th>
<th>Direct Connections</th>
<th>All</th>
</tr>
</thead>
<tbody>
<tr>
<td>$W_s$</td>
<td>$F_{cr}$</td>
<td>$W_d$</td>
<td>$F_{cd}$</td>
<td>$W_a$</td>
<td>$F_{ca}$</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
<td>8</td>
<td>8</td>
<td>24</td>
<td>24</td>
</tr>
</tbody>
</table>

Table 7.7a Actual Routing Times for Alliance Software

<table>
<thead>
<tr>
<th>Benchmark Circuits</th>
<th>Time (Sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C432</td>
<td>50</td>
</tr>
<tr>
<td>C499</td>
<td>54</td>
</tr>
<tr>
<td>C880</td>
<td>68</td>
</tr>
<tr>
<td>C1355</td>
<td>60</td>
</tr>
<tr>
<td>C1908</td>
<td>98</td>
</tr>
<tr>
<td>C2670</td>
<td>193</td>
</tr>
<tr>
<td>C3540</td>
<td>234</td>
</tr>
<tr>
<td>C5315</td>
<td>334</td>
</tr>
<tr>
<td>C7552</td>
<td>508</td>
</tr>
<tr>
<td>C6288</td>
<td>515</td>
</tr>
</tbody>
</table>

Table 7.7b Routability Prediction Results (Brown's Model)

<table>
<thead>
<tr>
<th>Benchmark Circuits</th>
<th>Total Switches-per-tile</th>
<th>Used Switches-per-tile</th>
<th>Time (Sec.)</th>
<th>Routability (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C432</td>
<td>320</td>
<td>112</td>
<td>4.66</td>
<td>99.99</td>
</tr>
<tr>
<td>C499</td>
<td>320</td>
<td>128</td>
<td>5.33</td>
<td>99.95</td>
</tr>
<tr>
<td>C880</td>
<td>320</td>
<td>137</td>
<td>8.12</td>
<td>99.32</td>
</tr>
<tr>
<td>C1355</td>
<td>320</td>
<td>130</td>
<td>5.27</td>
<td>99.95</td>
</tr>
<tr>
<td>C1908</td>
<td>320</td>
<td>121</td>
<td>14.34</td>
<td>99.94</td>
</tr>
<tr>
<td>C2670</td>
<td>320</td>
<td>185</td>
<td>32.57</td>
<td>98.27</td>
</tr>
<tr>
<td>C3540</td>
<td>320</td>
<td>123</td>
<td>57.45</td>
<td>99.87</td>
</tr>
<tr>
<td>C5315</td>
<td>320</td>
<td>202</td>
<td>93.86</td>
<td>93.15</td>
</tr>
<tr>
<td>C7552</td>
<td>320</td>
<td>184</td>
<td>164.78</td>
<td>96.51</td>
</tr>
<tr>
<td>C6288</td>
<td>320</td>
<td>320</td>
<td>189.6</td>
<td>99.96</td>
</tr>
</tbody>
</table>
Table 7.7c Routability Prediction Results (Our Model)

<table>
<thead>
<tr>
<th>Benchmark Circuits</th>
<th>Total Switches-per-tile</th>
<th>Used Switches-per-tile</th>
<th>Time (Sec.)</th>
<th>Routability (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C432</td>
<td>744</td>
<td>110</td>
<td>0.94</td>
<td>99.99</td>
</tr>
<tr>
<td>C499</td>
<td>668</td>
<td>90</td>
<td>1.32</td>
<td>99.99</td>
</tr>
<tr>
<td>C880</td>
<td>668</td>
<td>93</td>
<td>1.82</td>
<td>99.95</td>
</tr>
<tr>
<td>C1355</td>
<td>668</td>
<td>91</td>
<td>1.32</td>
<td>99.99</td>
</tr>
<tr>
<td>C1908</td>
<td>686</td>
<td>98</td>
<td>2.2</td>
<td>99.99</td>
</tr>
<tr>
<td>C2670</td>
<td>668</td>
<td>90</td>
<td>5.17</td>
<td>99.98</td>
</tr>
<tr>
<td>C3540</td>
<td>668</td>
<td>91</td>
<td>5.61</td>
<td>99.99</td>
</tr>
<tr>
<td>C5315</td>
<td>668</td>
<td>90</td>
<td>11.97</td>
<td>99.95</td>
</tr>
<tr>
<td>C7552</td>
<td>628</td>
<td>74</td>
<td>18.29</td>
<td>99.99</td>
</tr>
<tr>
<td>C6288</td>
<td>668</td>
<td>83</td>
<td>12.47</td>
<td>99.99</td>
</tr>
</tbody>
</table>

XC4000XL series FPGAs have rich routing resources, both models predicted high routabilities for the circuits. The prediction time in each model is less than the actual routing time. The prediction results from our model are slightly better than those from Brown’s model due to the use of hierarchical routing resources. The used-switches-per-tile and the prediction time are much better in our model. This verifies that our model can predict the routability of a design more accurately and more quickly, in addition to using fewer switches.

7.3.6 Time Complexity Analysis

In an FPGA-based design, the time required for routing can be affected by many factors, including the size of the target FPGA, the available routing resources, the complexity of the design, and the efficiency of routing tools. Two of the most important factors are the complexity of a design, represented by Rent exponent $r$, and the number of connections that need to be routed. We use the product of Rent exponent and the number of connections in the design to represent the complexity of the routing task, as follows:

$$\text{Routing Complexity} = r \times \text{Number of Connections} \quad (7.5)$$
Table 7.8 shows the routing complexities of the ten benchmark circuits used in our experiment in Step 2. The routing times for Alliance software and for our model, in terms of routing complexity, are shown in Figure 7.3.7.

<table>
<thead>
<tr>
<th>Benchmark Circuits</th>
<th>Estimated Rent Exponent</th>
<th>Number of Connections</th>
<th>Routing Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>C432</td>
<td>0.44</td>
<td>242</td>
<td>106.48</td>
</tr>
<tr>
<td>C499</td>
<td>0.60</td>
<td>281</td>
<td>168.60</td>
</tr>
<tr>
<td>C880</td>
<td>0.56</td>
<td>430</td>
<td>240.80</td>
</tr>
<tr>
<td>C1355</td>
<td>0.60</td>
<td>278</td>
<td>166.80</td>
</tr>
<tr>
<td>C1908</td>
<td>0.43</td>
<td>572</td>
<td>245.96</td>
</tr>
<tr>
<td>C2670</td>
<td>0.67</td>
<td>936</td>
<td>627.12</td>
</tr>
<tr>
<td>C3540</td>
<td>0.40</td>
<td>1420</td>
<td>568.00</td>
</tr>
<tr>
<td>C5315</td>
<td>0.63</td>
<td>2065</td>
<td>1300.95</td>
</tr>
<tr>
<td>C7552</td>
<td>0.61</td>
<td>2674</td>
<td>1631.14</td>
</tr>
<tr>
<td>C6288</td>
<td>0.33</td>
<td>3087</td>
<td>1018.71</td>
</tr>
</tbody>
</table>

Figure 7.3.7 Time Complexity Comparison

Figure 7.3.7 indicates that the time complexity of our prediction model is much lower than the real routing software. We use SPSS again to estimate the functions in
Figure 7.3.7 and find that the time complexity for our model is *linear* and that for the Alliance tools is a Power function, as shown in Figures 7.3.8a and 7.3.8b. The estimated functions are as follows:

\[ Y_{\text{Alliance}} = X^{0.83} \]  
\[ Y_{\text{Our}} = 0.01 \times X - 0.5 \]  

where \( X \) is the routing complexities of circuits routed by Alliance software and predicted by our model; \( Y_{\text{Alliance}} \) and \( Y_{\text{Our}} \) are the routing and prediction time for the circuits, respectively.

![Time Complexity for Our Model](image)

*Figure 7.3.8a Time Complexity for Our Model*
The time complexity comparison reveals that using our model to predict the routing result of a circuit is much faster than actually routing the circuit with software. The more complex the routing task, the more time our model can save.

7.4 Performance of Hierarchical FPGA with a Routing Hierarchy

Experiments were also conducted to evaluate the performance of our proposed hierarchical FPGA (HFPGA) model. We compared the HFPGA model developed by Li in [2] with our HFPGA model presented in Chapter 6 in terms of switch consumption. In addition, for each hierarchy level, the length distribution and the effect of different types of routing wires were also studied.
7.4.1. Comparison with Li's Model

We applied the same five MCNC benchmark circuits used in Li's model (Table 7.3) to our HFPGA prediction model. A 16-3 HFPGA structure was adopted for both models. Since the logic blocks are arranged in a hierarchical fashion, the basic element at each level (except level 0) is not a single logic block, but a level block. Therefore, we use switches-per-block to replace switches-per-tile measurement. However, the switches-per-block at each level is not a good indicator of the overall switch consumption in an HFPGA. We chose to compare the total number of switches available in an HFPGA for both models. The equation for calculating switch consumption is given by

$$\#Total\ Switches = \sum_{level=0}^{max} Switches\ -\ per\ -\ block_{level} * \#blocks_{level} \quad (7.8)$$

where max is the maximum level of hierarchy in an HFPGA and \#blocks_{level} is the number of blocks at a specific level. In our 16-3 HFPGA structure, it is $16^3$, $16^2$, and $16^1$ for level 0, level 1, and level 2, respectively.

Equation 7.8 is used to obtain the switch consumption based on routing parameters in the experiments. The routing parameters for Li's model are obtained from [2] and are shown in Table 7.10. For our model, one set of routing parameters with the minimal number of switches-per-block at each level is shown in Table 7.9. The two tables show that more routing resources are required at higher levels. For example, while routing the circuit "Elliptic" with our model, the overall channel width at level 0 is 5 (2+2+1), and increases to 30 (2+2+21+5) at level 2. In Li's model also, the same kind of increment exists. It is further revealed in our model that the increased number of tracks is due mostly to Quad-length Lines. This observation can be explained by the connection length probability distribution, which is discussed in the next sub-section.
Table 7.9 Routing Parameters for Our H FPGA Model

<table>
<thead>
<tr>
<th>Routing Hierarchy</th>
<th>Benchmark Circuits</th>
<th>Elliptic</th>
<th>Frisc</th>
<th>Spla</th>
<th>S298</th>
<th>Apex2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Level 0</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single-length Line</td>
<td>$W_s$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$F_{cs}$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Double-length Line</td>
<td>$W_d$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$F_{cd}$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Quad-length Line</td>
<td>$W_q$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$F_{cq}$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Long Line</td>
<td>$W_l$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>$F_{cl}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>All</strong></td>
<td>$F_s$</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td><strong>Level 1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single-length Line</td>
<td>$W_s$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$F_{cs}$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Double-length Line</td>
<td>$W_d$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$F_{cd}$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Quad-length Line</td>
<td>$W_q$</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$F_{cq}$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Long Line</td>
<td>$W_l$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>$F_{cl}$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td><strong>All</strong></td>
<td>$F_s$</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td><strong>Level 2</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single-length Line</td>
<td>$W_s$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$F_{cs}$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Double-length Line</td>
<td>$W_d$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>$F_{cd}$</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Quad-length Line</td>
<td>$W_q$</td>
<td>21</td>
<td>19</td>
<td>12</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>$F_{cq}$</td>
<td>20</td>
<td>18</td>
<td>10</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>Long Line</td>
<td>$W_l$</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>$F_{cl}$</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td><strong>All</strong></td>
<td>$F_s$</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
Table 7.10 Routing Parameters for Li’s Model

<table>
<thead>
<tr>
<th>Benchmark Circuits</th>
<th>$F_c$ Req'd for each level</th>
<th>$F_r$ Req'd for each level</th>
<th>Channel Width Req'd for each level</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elliptic</td>
<td>8,16,40</td>
<td>4,4,4</td>
<td>10,18,45</td>
</tr>
<tr>
<td>Frisc</td>
<td>8,16,41</td>
<td>4,4,5</td>
<td>10,18,46</td>
</tr>
<tr>
<td>Spla</td>
<td>8,17,42</td>
<td>4,4,5</td>
<td>10,18,45</td>
</tr>
<tr>
<td>S298</td>
<td>4,4,6</td>
<td>3,3,3</td>
<td>6,7,8</td>
</tr>
<tr>
<td>Apex2</td>
<td>4,6,8</td>
<td>3,3,3</td>
<td>6,8,12</td>
</tr>
</tbody>
</table>

The relative switch consumption for the two models is shown in Table 7.11. Since the data for used switches-per-block and prediction time are not available in [2], we only list the total number of switches in an HFPGA. The comparison results are plotted in Figure 7.4.1. As the figure indicates, our HFPGA model requires fewer switches than Li’s. Therefore, together with the results obtained from the first two experiments, we conclude that utilizing hierarchical routing resources in an FPGA’s routing architecture can greatly enhance its performance by reducing the number of switches used for routing. As a result, the speed of the circuits implemented in the FPGA is increased.

Table 7.11 Switch Consumption Comparison for the Two HFPGA Models

<table>
<thead>
<tr>
<th>Benchmark Circuits</th>
<th>Total Switches in an HFPGA (Li’s Model)</th>
<th>Total Switches in an FPGA (Our Model)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elliptic</td>
<td>895360</td>
<td>545832</td>
</tr>
<tr>
<td>Frisc</td>
<td>897184</td>
<td>545823</td>
</tr>
<tr>
<td>Spla</td>
<td>885278</td>
<td>549823</td>
</tr>
<tr>
<td>S298</td>
<td>404032</td>
<td>246960</td>
</tr>
<tr>
<td>Apex2</td>
<td>413568</td>
<td>252936</td>
</tr>
</tbody>
</table>
Although the prediction time for Li's model is not available, it can be estimated by using the time complexity function of our model (Equation 7.7) and the relation between Brown's model and our model (Equation 7.1). Due to the hierarchical structure of HFPGAs, we have to estimate the prediction time based on each hierarchy level. The overall prediction time can be obtained by adding up the prediction times for all levels. Table 7.12 shows the time comparison between Li's model and our model.

Table 7.12 Time Comparison for the Two HFPGA Models *

<table>
<thead>
<tr>
<th>Benchmark Circuits</th>
<th>Estimated Prediction Time for Li's Model (Second)</th>
<th>Prediction Time for Our Model (Second)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elliptic</td>
<td>1818.04</td>
<td>41.43</td>
</tr>
<tr>
<td>Frisc</td>
<td>758.06</td>
<td>28.41</td>
</tr>
<tr>
<td>Spla</td>
<td>461.78</td>
<td>16.50</td>
</tr>
<tr>
<td>s298</td>
<td>287.74</td>
<td>11.96</td>
</tr>
<tr>
<td>Apex2</td>
<td>557.42</td>
<td>14.85</td>
</tr>
</tbody>
</table>

* Based on a Pentium 166 processor with 32MB memory
Due to the hierarchical structure, the connection length probability distribution functions are different in HFPGAs than in conventional FPGAs. Since we predict routability on the basis of hierarchical levels, we have to estimate the length distribution function in the same way. Stroobandt’s method [8] is adopted for obtaining the distribution functions in our HFPGA model. Figure 7.4.2 shows the connection length distribution for circuit “Elliptic”, which is implemented in a 16-3 HFPGA. Comparing this distribution with the one for a conventional FPGA in Figure 7.3.5, we find that the patterns of the two distributions are quite similar. In both types of FPGAs, short connections have much higher probabilities than long ones. This also verifies the correctness of the distribution functions we apply in our HFPGA model.

The connection length distribution functions for each level are shown in Figure 7.4.3. Analyzing the figure, we make the following observations:
I. At each level, the probability of shorter connections is greater than that of longer connections, which is also the case for conventional FPGAs. However, the pattern at each level is smoother than that in the conventional FPGA model.

II. As the level increases, the ratio of the probabilities for short connections to the total probability decreases. In other words, the higher the hierarchy level, the bigger the portion of the long connections.

Figure 7.4.3 Length Probability Distributions at Each Level
III. A connection of given length, especially when the length is short, has a higher probability at a lower level than at a higher level.

Similar to the patterns for conventional FPGAs, the patterns here can be explained by means of the placement procedure for HFPGAs. If a technology-mapped circuit is optimally placed in an HFPGA, logic blocks with high connectivity amongst them should be grouped together and placed inside a low-level block. Then, these low-level blocks with high connectivity amongst them are placed near each other in a higher-level block. This process continues until the whole circuit is placed in the HFPGA. Since the placement procedure at each level follows the same rule, which favors placing blocks with high connectivity closely, all the length distributions display the same trend. At higher hierarchy levels, due to the increased block size, the average connection length is longer than that at lower levels with smaller-sized blocks. As a result, as the hierarchy level increases, the proportion of longer connections also increases, and longer-segment routing wires are required for routing.

In a hierarchical structure, the higher the level, the fewer the number of blocks it has. In our HFPGA model, the number of connections per level is also decreasing along with the increment of hierarchy. However, for complex designs, the decrement rate of the number of the connections is slower than that of the number of the blocks. This means on average, a higher-level block has more connections to be routed than a lower-level block does. To successfully route all the connections, the program requires more routing resources with greater flexibility at higher levels. Therefore, longer connection length and heavy traffic in a channel result in high channel width for Quad-length Lines. This
explains why at level 2, more Quad-length Lines are required and the flexibility of switch blocks \( F_r \) needs to be higher than at lower levels.

7.5 Conclusion

In this chapter, experimental results are discussed to determine the different features of hierarchical routing resources. The following conclusions can be drawn from the experimental results, which may help manufacturers design an optimal HFPGA routing architecture:

i. FPGAs with hierarchical routing resources use far fewer number of routing switches to implement a circuit compared to that required by symmetrical FPGAs with no routing hierarchy.

ii. Due to the placement schema of FPGAs, most of the connections are short. Only a small portion of the connections are very long.

iii. More than half of the connections are routed by Single-length Lines in conventional FPGAs. However, the channel width required for Single-length Lines is surprisingly low, while the channel width for longer-segment routing wires, especially Quad-length Lines is relatively high.

iv. In HFPGAs, the slopes of connection length distributions at each level are similar to those in conventional FPGAs. However, the ratio of long connections to the total number of connections increases as the hierarchy level increases. Therefore, the required channel width increases with each increment of hierarchy level.

v. In HFPGAs with a routing hierarchy, more Quad-length Lines are required at higher levels.
Chapter 8
Conclusions

8.1 Thesis Summary

Routing is a very crucial step in the FPGA-based designs, since it determines if a circuit can be implemented on an FPGA. Its efficiency is greatly affected by the FPGA routing architecture and routing resources. Some research has been done on the impact of routing architecture on the performance of an FPGA [1-3]. However, only a single type of routing wire is considered in these studies, and little systematic work has been done on the effect of different routing resources. This thesis examines hierarchical structure of routing resources, and the model developed can be applied to different types of routing architectures (symmetrical and hierarchical). Wires with different segment lengths provide the advantage of fewer switches used in an FPGA-based design. As a result, the speed of designs implemented on an FPGA can be greatly increased.

Chapter 4 introduced the hierarchical routing resources, as well as their usage. Two FPGA models that integrate the routing hierarchy are presented in Chapters 5 and 6. These models can be used to predict the routability of designs on FPGAs which have either a symmetrical or hierarchical routing architecture. It is demonstrated that the models proposed in this thesis can estimate the routability well when compared to the real routing results obtained from commercial CAD software tools. Furthermore, the proposed models use far fewer switches for routing the designs compared to existing methods. The main purpose of this thesis is to investigate the properties of different types
of routing wires and their effect on routability. Experiments were conducted to compare the routing resource consumption of FPGAs with two different routing architectures and to gain insight into the flexibility required for different routing wires in these architectures. Appropriate software tools were developed to perform these studies. These tools are used to predict the routability of a netlist on an FPGA directly after technology-mapping, but before placement. The tools can be used to predict the routability of a circuit on an existing FPGA or to estimate the effectiveness of different routing resources for a new type of FPGA. We also analyze the number of routing resources required in different routing architectures. The conclusions reached are summarized as follows:

- The number of routing switches required by a design on FPGAs with a routing hierarchy is lower than that required on FPGAs with no routing hierarchy.

- In an FPGA with hierarchical routing resources, most of the connections are routed by Single-length Lines. However, these do not require high channel width ($W$) and high switch flexibility ($F_c$ and $F_s$) for routing. On the other hand, although the number of connections routed by longer-segment lines (Double-length Lines and Quad-length Lines) is very small, their flexibility requirements ($W$, $F_c$, and $F_s$) are relatively high.

- Hierarchical routing resources can further improve the performance of HFPGAs by reducing the number of switches required for in routing.

- When different types of routing wires are used in HFPGAs, more Quad-length Lines are required at higher-levels.
8.2 Suggestions for Future Work

In our models, different types of routing wires are treated independently of each other. Any connection is assumed to be routed by only one type of routing wire, and combinations of routing wires are not considered. This assumption is made so that the channel densities for all wires still follow the Poisson distribution. In commercial FPGAs, routing a connection by joining different types of wires is common. In fact, the combination makes the usage of routing resources more efficient and can further reduce the number of switches. However, due to the various types of routing wires and the enormous number of possible combinations, taking this kind of usage into consideration would require an enormously complex model that is likely beyond the scope of this thesis.

Many aspects of hierarchical routing resources require further study. Some of the aspects deserving further study are:

a) Relationship between the total channel width (including all types of wires) and the channel widths required for individual types of routing wires to obtain the best routability with minimal routing resource consumption,

b) The effect of using different switch block flexibilities ($F_s$) in the hierarchical routing resources.

c) The optimal values of routing parameters ($W$, $F_e$, and $F_s$) for different routing resources, in association with FPGA size and the complexity of a design.

Taking these considerations into account would result in a large number of possible combinations and possibly an enormous amount of processing time. However, a
modification of our models incorporating variations in these flexibility parameters would be worth investigating.

Other directions for future work are to manage time constraints of the applications in the prediction models and to study the effect of hierarchical routing resources on other types of FPGAs, such as row-based FPGAs.
References

[1] Stephen D. Brown, Robert J. Francis, Johnathan Rose, and Zvonko G. Vranesic, 


Field Programmable Gate Arrays”, MSc Thesis, Department of Electrical and Computer Engineering, University of Toronto, 1994.


Master Slice Integrated Circuits”, IEEE Transactions on Circuits and Systems, 


pp.152-155.


1. Limitations of Xilinx Alliance Tools

In our Step 2 experiment, Xilinx Alliance software package is used to validate our proposed FPGA model with a routing hierarchy. Ten MCNC benchmark circuits in "XNF" netlist file format were fed to the Alliance software package and routed on Xilinx 4000XL series FPGAs. Important information, which includes number of CLB used, number of connections, number of I/O pins, and total number of pins, is extracted from the report files. However, the report files do not provide any information about the total number of switches used in routing. Therefore, we cannot compare the used-switches-per-tile with our model and Brown’s model.

2. Obtaining the Real Used-switches-per-tile

Although the Alliance software package does not provide the number of switches used in routing, it is possible to obtain it. A simple way is to count the switches used in a routed design manually. However, this works only with small designs. Large designs will have thousands of switches. Counting them manually will be very difficult and really time consuming.

Here we present a method to count the number of switches used in small designs routed by Alliance software package. The package contains a program called FPGA Editor, which is used to view and edit a routed FPGA. It shows the placement and routing results graphically in an Array Window and the properties of the routed design in
a List Window. Select "All Net" in the List Window. It will show all the nets in the design. A net is an electrical connection between components or nets. It is the same as a signal. A routed design can have from several to thousands of nets. When a net is selected in the list, the corresponding routed connections are highlighted in the Array Window. The switches are shown on these connections as small dots. By counting the number of dots, we get the number of switches used in each net. Therefore, the total number of switches can be obtained by adding up all the dots in each net.

3. Analysis of Results

Two simple MCNC benchmark circuits, C432 and C499 were routed by the Alliance software package. Their real number of switches used is obtained by using the method presented in the previous section. The results are as follows:

Table A-1 Routing Information of Benchmark Circuits C432 and C499

<table>
<thead>
<tr>
<th>Circuit Name</th>
<th>C432</th>
<th>C499</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of CLBs</td>
<td>30</td>
<td>39</td>
</tr>
<tr>
<td>Number of Nets</td>
<td>92</td>
<td>111</td>
</tr>
<tr>
<td>Total Number of Switches</td>
<td>742</td>
<td>1013</td>
</tr>
<tr>
<td>Used-switches-per-tile (Real)</td>
<td>25</td>
<td>26</td>
</tr>
<tr>
<td>Used-switches-per-tile (Our Model)</td>
<td>110</td>
<td>90</td>
</tr>
<tr>
<td>Used-switches-per-tile (Brown’s Model)</td>
<td>112</td>
<td>128</td>
</tr>
</tbody>
</table>

The table indicates that the real number of used-switches-per-tile for each circuit is less than the predictions. This is due to the use of different routing strategies. In the models presented in [1] and [2], and this thesis, it is assumed that a circuit with a total of
two-point connections is to be routed on an FPGA. However, Xilinx Alliance software package routes all the connections by net. A net is usually composed of many two-point connections. Therefore, routing by net may save a number of redundant tracks and switches, compared to routing by two-point connections. Furthermore, different types of routing wires are used independently in our models, while commercial FPGAs allow the user to route a net by mixing different wires. Short-segment lines and long-segment lines can be connected to each other to implement routing. This kind of mixing can further reduce the number of switches. These reasons (routing by net and mixing wire types) result in less number of switches used per tile by Alliance tools. Taking this routing strategy into consideration would require an enormously complex model, which is left for future study.

Our models can predict the routability of FPGAs. In addition, our models can be used to analyze the properties of FPGA routing resources. They can help manufacturers to design new routing architectures or to optimize existing ones.