Routing Algorithms and Architectures for Field-Programmable Gate Arrays

Stephen Dean Brown

January 1992

Routing Algorithms and Architectures for Field-Programmable Gate Arrays

by

Stephen Dean Brown

A thesis submitted in conformity with

the requirements for the degree of

Doctor of Philosophy

January 1992

Department of Electrical Engineering

University of Toronto

Toronto, Ontario

CANADA

"This is indeed a mystery, [remarked Watson] what do you imagine that it means?" "I have no data yet. It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts."

- "Sherlock Holmes," A. Conan Doyle

Abstract

Field-Programmable Gate Arrays (FPGAs) are a new type of user-programmable integrated circuits that supply designers with inexpensive, fast access to customized VLSI. A key component in the design of an FPGA is its *routing architecture*, which comprises the wiring segments and routing switches that interconnect the FPGA's logic cells. Each of the user-programmable switches in an FPGA consumes significant chip area and has appreciable capacitance and resistance, leading to a tradeoff in the design of a good routing architecture. Providing a large number of switches will yield a flexible architecture in which the logic cells are easily interconnected, but too many switches wastes area and degrades speed performance. On the other hand, fewer switches allows better speed performance and uses less area, but if there are too few switches then it may not be possible to implement the desired circuits. This thesis studies FPGA routing architectures with regard to this tradeoff, yielding three main contributions.

A novel detailed routing algorithm that can account for the limited connectivity in FPGA routing architectures has been developed. It can be used over a wide range of FPGA routing architectures, and represents the first published algorithm that approaches detailed routing in FPGAs in a general way. The algorithm addresses the unique issues in FPGA routing by accounting for the side-effects that the routing of one connection may have on others, allowing it to resolve contention for the routing resources. It is shown that the router yields excellent results for a set of relatively large industrial circuits implemented as FPGAs. The router is the principal tool that is used for the experimental study of FPGA routing architectures done in this thesis.

Experiments have been conducted to study the effects of the flexibility of FPGA routing architectures on the *routability*, which is the percentage of connections that can

be successfully completed, of circuits. *Flexibility* is a measure of the total number of routing switches and wiring segments in a routing architectures. The experiments show that a high flexibility is required in the connection blocks that join the logic cells to the routing channels, but a relatively low flexibility is sufficient in the switch blocks at the intersections of horizontal and vertical channels. It is also shown that a surprisingly small number of tracks per routing channel is sufficient to allow circuits to be configured, even when the flexibility is low.

Finally, a stochastic model has been developed that allows the study of FPGA routing architectures using a theoretical approach. In the model, both an FPGA and a circuit to be configured are represented as simple parameters, and probability theory is used to predict the effect of routing the circuit in the FPGA. The model corroborates the experimental results with the same circuits. It provides the foundation of a theoretical approach that can be used in future studies of FPGA routing architectures, without time-consuming experiments.

Acknowledgements

I would like to take this opportunity to express my sincere thanks and appreciation to my academic supervisors. Professor Zvonko G. Vranesic has provided a continual source of guidance, advice, encouragement, and friendship throughout my graduate studies. It has been my privilege to work with him. Professor Jonathan S. Rose has provided a great source of inspiration throughout my doctoral studies. Without his technical advice and friendship, this thesis could not have transpired.

Susan Lo, my fiancee, has been a constant source of support, providing a stable, happy personal life. I would also like to thank Tony for the endless hours of play.

My father and mother have always supported my studies and deserve much credit for enabling me to reach this milestone.

I would like to thank my academic supervisors, the Natural Sciences and Engineering Research Council, the Information Technology Research Centre, and Micronet for their financial support.

TABLE OF CONTENTS

1 Introduction

1.1 Introduction to Field-Programmable Gate Arrays

Field-Programmable Gate Arrays (FPGAs) are a revolutionary new type of user-programmable integrated circuits that provide fast, inexpensive access to customized VLSI. An FPGA consists of an array of logic cells that can be interconnected via programmable routing switches, where the routing structures are sufficiently general to allow the configuration of multiple levels of the FPGA's logic cells. FPGAs represent a combination of the features of Mask Programmable Gate Arrays (MPGAs) and Programmable Logic Devices (PLDs). From MPGAs, FPGAs have adopted a two-dimensional array of logic cells, and from PLDs the user-programmability. The research reported in this thesis is focused on FPGA routing algorithms and routing architectures.

Following their introduction in 1985, by the Xilinx Company [Cart86], FPGAs have evolved considerably as various new devices have been developed [ElGa88] [ElGa89] [Wong89] [Ahre90] [AMD90] [Gupt90] [Hsie88] [Hsie90] [Kawa90] [Marr89] [Ples89] [Plus90]. FPGAs have quickly gained widespread use, which can be attributed to the reduced manufacturing time and relatively low costs of these large-capacity user-programmable devices. As an implementation medium for customized VLSI circuits, FPGAs offer unique advantages over the alternative technologies (MPGAs, standard cells, and full custom design):

(1) FPGAs provide a reduction in the cost of manufacturing a customized VLSI circuit from tens of thousands of dollars to about one hundred dollars.

(2) FPGAs reduce the manufacturing time from months to minutes.

These advantages, which are attributable to the user-programmability of FPGAs, provide a faster time-to-market and less pressure on designers, because multiple design iterations can be done quickly and inexpensively. However, user-programmability also has drawbacks: the logic density and speed performance of FPGAs is considerably lower than those of the alternatives. While developments over the last few years have shown significant improvements in FPGAs, much research is still needed before the best FPGA designs are discovered.

1.2 Thesis Motivation

Circuits are implemented in an FPGA by interconnecting its logic cells through the user-programmable routing switches. Two distinct purposes are served by the *routing switches*: to connect the logic cells to the routing wires, and to connect one routing wire to another. One example of an FPGA routing switch is a CMOS pass-transistor controlled by a static memory bit [Cart86], but there are a number of other implementations that are used in commercial products. Regardless of the implementation, routing switches consume significant chip area and have appreciable resistance and parasitic capacitance. For these reasons, it is desirable to limit the number of routing switches in an FPGA.

All of the routing switches and wires in an FPGA, and their distribution over the surface of the chip, are collectively referred to as the FPGA's *routing architecture*. A measure of the connectivity provided by a routing architecture is its *flexibility*, which is a function of the total number of routing switches and wires. The design space for FPGA routing architectures is enormous. Choosing a good design involves a tradeoff among flexibility, logic density, and speed performance. A high flexibility yields an FPGA that is easily configured, but if the flexibility is too high then area will be wasted by unused

switches, leaving less area for the logic blocks and resulting in lower logic density. Moreover, since each routing switch introduces an RC-delay, high flexibility results in reduced speed performance. Low flexibility, on the other hand, allows higher logic density and lower RC-delay, but if the flexibility is too low, then it may not be possible to interconnect the logic cells sufficiently to implement circuits. A good routing architecture is one that achieves a balance between these competing factors.

The primary focus of this thesis is the study of FPGA routing architectures with regard to the flexibility, logic density, and speed performance tradeoff. The goal of the study is to determine the minimum flexibility that is necessary to provide sufficient interconnection capability to satisfy the requirements of real circuits, and yet low enough so that routing switches are not wasted. This research is part of a large project [Brown90] [Brown91] [Fran90] [Fran91] [Rose89] [Rose90a] [Rose90b] [Rose90c] [Rose91] [Sing91] that examines many aspects of the Computer-Aided Design (CAD) and architecture of FPGAs.

1.3 Research Approach

FPGA routing architectures are studied in this thesis using both an experimental and a theoretical approach. For the experimental study, a new type of detailed routing algorithm has been developed that is able to route a wide range of FPGA routing architectures. The experiments consist of varying the routing architecture flexibility and using the router to measure the resulting effects on the routability of circuits. The results of the experiments provide insights into the amount of routing resources that is sufficient to meet the requirements of real circuits and yet low enough so that the resources are not wasted. These issues are also studied using a theoretical approach that represents both a circuit and an FPGA as simple parameters of a stochastic model.

1.4 Dissertation Organization

This dissertation is organized as follows. Chapter 2 provides background information, including a discussion of general approaches to routing problems, the definitions of routing terminology, and a short history of routing algorithms. It also describes representative examples of commercially available FPGAs, including a brief description of the routing architecture contained in each chip.

Chapter 3 presents a new detailed routing algorithm, designed specifically for FPGAs. The algorithm is unique in that it approaches FPGA routing in a general way, and is designed such that it can be used over a wide range of FPGA routing architectures. The algorithm is the main tool that is used to produce the experimental results that are shown in Chapter 4.

FPGA routing architectures are studied in Chapter 4 using an experimental approach. The algorithm in Chapter 3 is used to route a set of circuits in an FPGA based on a model that allows the routing structures in the FPGA to be changed. For each circuit, a range of flexibilities is evaluated by varying the number of routing switches and wires. The experiments measure the effect of the flexibility of the routing architecture on the percentage of interconnections that can be successfully routed for each circuit.

Chapter 5 investigates FPGA routing architectures using a theoretical approach. For this study, both the FPGA and a circuit are represented by simple parameters. A stochastic model is developed to predict the effect of routing the circuit in the FPGA. The model corroborates the experimental results from Chapter 4 and provides the foundation of a theoretical model that can be used in future studies of FPGA routing architectures.

Chapter 6 provides concluding remarks and directions for future research. References are listed at the end.

2 Background Information

2.1 Introduction

This chapter introduces the two main fields of research, FPGA routing algorithms and FPGA routing architecture, that are studied in this thesis. Section 2.2 provides some necessary background information that is assumed in various discussions, particularly in Chapter 3, about routing software. Section 2.3 describes several commercially available FPGA devices to provide a point of reference for the FPGA model that is used throughout this work, and particularly for the routing architecture results that are presented in Chapters 4 and 5.

2.2 Routing Algorithms

While the focus of this thesis is routing, this chapter begins with an overview of the entire CAD process that is necessary to implement a circuit in an FPGA. A typical CAD system for FPGAs consists of several interconnected programs as illustrated in Figure 2.1. The input to the CAD system is a functional description of a network, usually expressed in a standard format such as boolean equations. The equations are read by a logic optimization [Bray86] [Greg86] tool, which performs manipulations of the equations so as to optimize area, delay, or a combination of area and delay. This step usually performs the equivalent of an algebraic minimization of the boolean equations and is appropriate when implementing a circuit in any medium, not just FPGAs. To transform the boolean equations into a circuit of FPGA logic cells, the optimized network is fed to a technology mapping program [Kahr86] [Keut87] [Fran91]. This step maps the equations into logic cells, which also presents opportunity to optimize, either to minimize the total number of logic cells required (area optimization) or the number of logic cells in

time-critical paths (delay optimization). The circuit of logic cells is then passed to a placement program [Hana72] [Rose85] [Sech87], which selects a specific location in the FPGA for each logic cell. Typical placement algorithms usually attempt to minimize the total length of interconnect required for the resulting placement.

The final step in the CAD system is performed by the routing software, which allocates the FPGA's routing resources to interconnect the placed logic cells. The routing tools must ensure that 100 percent of the required connections are formed, and may be required to maximize the speed performance of time-critical connections. Finally, the CAD system's output is fed to a programming unit that is used to configure the FPGA. Since routing software is the key step in the CAD system for the purposes of this thesis, the remainder of this section provides a brief introduction to the subject.

2.2.1 Routing Terminology

Software that performs automatic routing has existed for many years, with the first algorithms designed to route printed circuit boards. Over the years there have been many publications concerning routing algorithms, so that the problem is well defined and

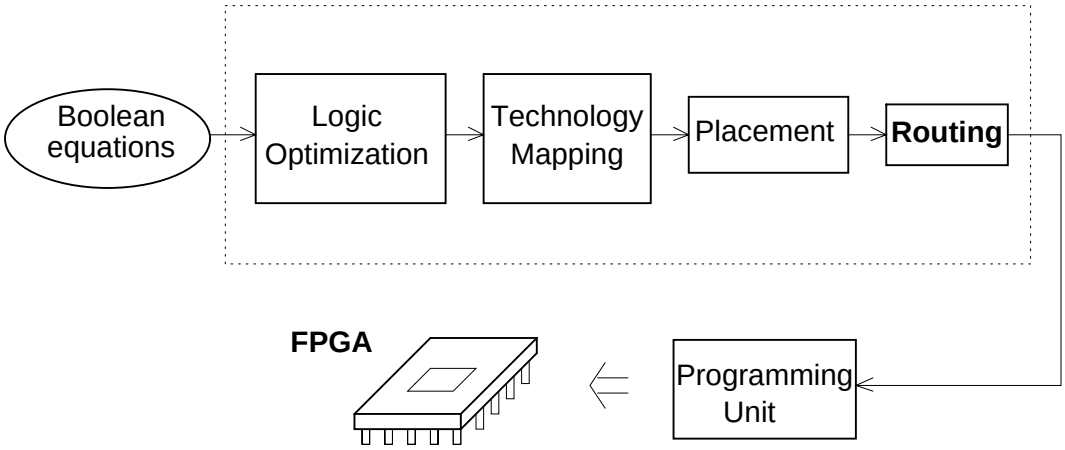

Figure 2.1 - *A Typical FPGA CAD System*

understood. The following list gives common routing terms, as they are defined for FPGA routing in this thesis:

- *Pin* - a logic cell input or output.
- *Connection* - a pair of logic cell pins that are to be electrically connected.
- *Net* - a set of logic cell pins that are to be electrically connected. A net can be divided into one or more connections.
- *Wiring segment* - a straight section of wire that is used to form part of a connection.
- *Routing switch* - a device that is used to electrically connect two wiring segments.
- *Track* - a straight section of wire that spans the entire width or length of a routing channel. A track can be composed of a number of wiring segments of various lengths.
- *Routing channel* - the rectangular area that lies between two rows or two columns of logic cells. A routing channel contains a number of tracks.

2.2.2 General Approach to Routing

Because of the combinatorial complexity involved, the solution of large routing problems usually requires a "divide and conquer" strategy. Following this philosophy, routing can be solved by a three-step process [Loren89]:

1. Partition the routing resources into routing areas that are appropriate for both the device to be routed and the routing algorithms to be employed.
2. Use a *global router* to assign each net to a subset of the routing areas. The global router does not choose specific wiring segments and routing switches for each connection, but rather it creates a new set of restricted routing problems.

3. Use a *detailed router* to select specific wiring segments and routing switches for each connection, within the restrictions set by the global router.

The advantage of this approach is that each of the routing tools can more effectively solve a smaller part of the routing problem. More specifically, since a global router need not be concerned with allocating wiring segments or routing switches, it can concentrate on more global issues, like balancing the usage of the routing channels. Similarly, with the reduced number of detailed routing alternatives that are available for each connection because of the restrictions introduced by a global router, a detailed router can focus on the problem of achieving connectivity. Its limited scope enables a detailed router to concentrate on resolving contention for routing resources that may exist among different nets.

The above routing strategy has been adopted in this thesis for FPGA routing. The routing resources are partitioned into horizontal and vertical routing channels.

2.2.3 Introduction to Global Routing

This section introduces global routing by describing the LocusRoute global routing algorithm [Rose90a] for standard cells. Although there are many other published techniques for global routing [Loren89] [Sech88] [Cong88], this specific algorithm is described as an example because a modified version of it is employed for FPGA global routing in this thesis. This algorithm has been chosen for FPGAs because, as described below, its primary goal is to balance the usage of the routing channels. This is important for FPGAs because the number of tracks per channel is pre-determined. Note that the description below is based on the standard-cell version of LocusRoute, and the main difference between this and the FPGA version is the definitions of the routing channels - the standard-cell program assumes only horizontal routing channels, whereas the FPGA

version uses both horizontal and vertical channels.

2.2.3.1 The LocusRoute Global Routing Algorithm

The LocusRoute global router views the global routing problem as consisting of three main tasks:

1. For nets comprising more than two pins, determine which pairs of pins to connect together. This step decomposes a multi-point net into a set of two-point connections.
2. Determine a path through the routing channels for each connection.
3. Optimize the solution so that the usage of all of the routing channels is balanced.

The first task is solved by finding a minimum-spanning tree [Prim57] for each net. Basically, this technique breaks a net into a set of two point connections such that the total amount of interconnect required is minimized.

To solve the second task, LocusRoute models each routing channel as an array of grids, as shown in Figure 2.2. Each grid location contains a counter, originally set to zero, which is incremented by one for each connection that is globally routed through it. In this way, the algorithm is able to maintain a detailed account of the usage of each routing channel, so that it can avoid congestion. The algorithm considers alternative ways of routing each connection and chooses the one that passes through the least congested routing grids. Note that LocusRoute does not consider all of the possible ways that a connection can be routed, but rather it evaluates only a subset of the paths that have "two or fewer bends", as explained in [Rose90a].

After all of the connections have been globally routed once, LocusRoute optimizes the solution by sequentially ripping up and re-routing each connection. After repeating

this procedure a small number of times, the final solution is output in a format suitable for the detailed router to be employed.

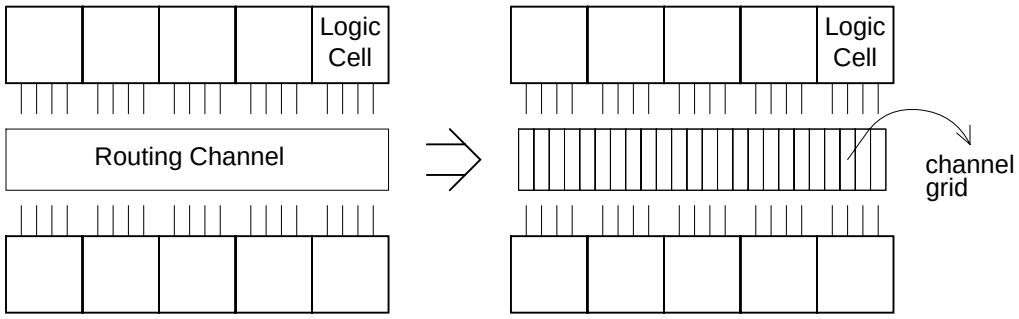

Figure 2.2 - *The Channel Grids Used by LocusRoute*

2.2.4 Introduction to Detailed Routing

This section provides an introduction to detailed routing by describing the *maze* routing technique. Although there exist many other detailed routing algorithms [Aker72] [Souk81] [Loren89], maze routing will be discussed because it is widely used due to its general applicability, and a variant of a maze router is employed as a comparison against the detailed routing algorithm for FPGAs that is described in Chapter 3.

2.2.4.1 The Lee Maze Router

Most maze routers can be considered to be a variant of the algorithm described in [Lee61]. This technique models the entire routing surface as a rectangular array of cells, where the size of each cell is defined so as not to violate the spacing rules for wiring segments. Connections are formed one at a time by selecting adjacent cells that reach from one end of a connection to the other. Once a grid location is occupied, either by a connection or by some sort of obstruction, it is marked as unusable. An array of routing cells is illustrated in Figure 2.3, where unusable cells are shaded and usable ones are not. The figure shows the detailed routes of three connections as they might be produced by a

maze router.

The Lee algorithm implements the array of cells as a regular graph, with one vertex for each cell and one edge joining each pair of adjacent cells. A connection is routed by beginning at one of its ends and traversing the graph in a breadth first fashion until the other end is reached. The result is a diamond shaped wavefront that emanates from the first point, as illustrated in Figure 2.4. The numbers in the figure correspond to each step as the wavefront is propagated.

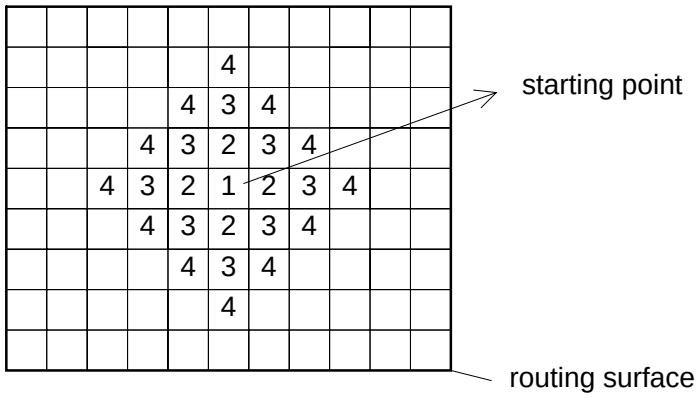

Figure 2.4 - *Maze Router Diamond Shaped Wavefront*

The main advantage of a maze router is that it is guaranteed to find a path from one end of a connection to the other, if one exists at the time the connection is routed. On the

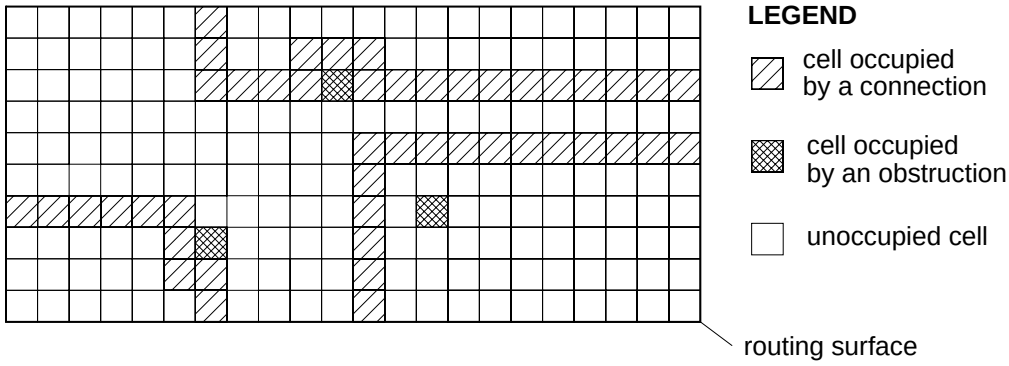

Figure 2.3 - *Maze Routers Model the Routing Surface by an Array of Cells*

other hand, because of its sequential nature a maze router is unable to consider the side-effects that the routing of one connection may have on another. Correspondingly, the main disadvantage of maze routing is the unnecessary blockage of as yet unrouted connections because of previous routing decisions.

2.3 Commercially Available FPGAs

This section provides a detailed description of three commercially available FPGA families, including those from Xilinx Co., Actel, and Altera. These particular FPGAs have been chosen because they are representative examples of state-of-the-art devices and they are in widespread use. Each device is described in terms of its general architecture, its choice of programmable cell, its routing architecture, and its CAD routing tools. Enough details are given, and in some cases specific comments are made, to show how the routing architecture of each device relates to the research contained in this thesis. In addition, at the end of the section, several recently introduced FPGAs are briefly described.

2.3.1 Xilinx FPGAs

The general architecture of Xilinx FPGAs is shown in Figure 2.5. It consists of a two-dimensional array of programmable cells, called Configurable Logic Blocks (CLBs), with horizontal routing channels between rows of cells and vertical routing channels between columns. Programmable resources are configured by Static RAM cells, and each routing switch is implemented as a specially designed transistor controlled by an SRAM bit. There are three families of Xilinx FPGAs, called the XC2000, XC3000, and XC4000 corresponding to first, second, and third generation devices. Table 2.1 gives an indication of the logic capacities of each generation by showing the number of CLBs and an equivalent gate count. The gate count measure is given in terms of "equivalent to an

MPGA of the same size." All FPGA manufacturers quote logic capacity by this measure, but it is questionable whether the figures quoted by each are realistic. The numbers given in Table 2.1, and in similar tables that appear later in this chapter, should be interpreted accordingly. The design of the Xilinx CLB and routing architecture differs for each generation, so they will each be described in turn.

2.3.1.1 Xilinx XC2000

The XC2000 CLB, shown in Figure 2.6, consists of a four-input look-up table and a D flip-flop [Cart86]. The look-up table can generate any function of up to four variables

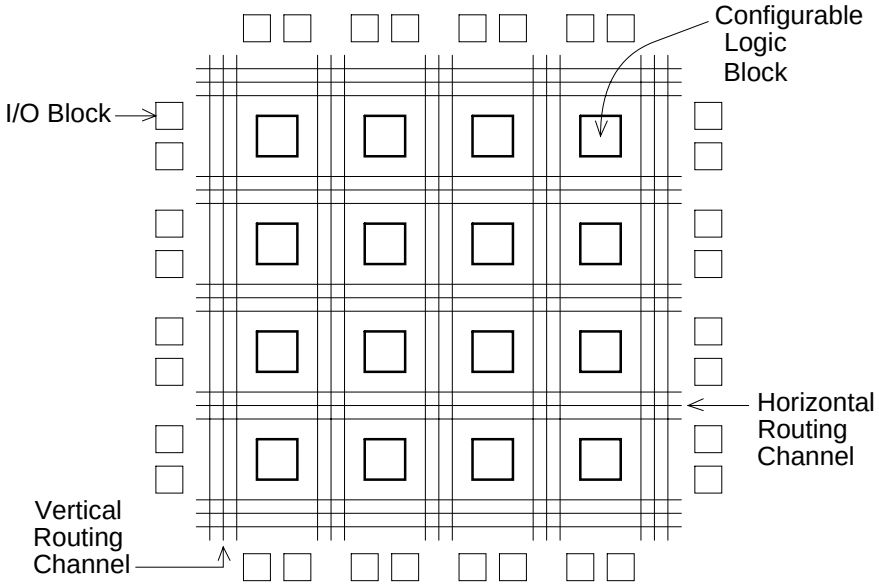

Figure 2.5 - *General Architecture of Xilinx FPGAs*

Series	Number of CLBs	Equivalent Gates
XC2000	64 - 100	1200 - 1800
XC3000	64 - 320	2000 - 9000
XC4000	64 - 900	2000 - 20000

Table 2.1 - *Xilinx FPGA Logic Capacities*

or any two functions of three variables. Both of the CLB outputs can be combinational, or one output can be registered.

As illustrated in Figure 2.7, the XC2000 routing architecture employs three types of routing resources: Direct interconnect, General Purpose interconnect, and Long Lines. Note that for clarity the routing switches that connect to the CLB pins are not shown in the figure. The Direct interconnect (shown only for the CLB marked with an '*') provides connections from the output of a CLB to its right, top, and bottom neighbours. For connections that span more than one CLB, the General Purpose interconnect provides horizontal and vertical wiring segments, with four segments per row and five segments per column. Each wiring segment spans only the length or width of one CLB, but longer wires can be formed because each switch matrix holds a number of routing switches that can interconnect the wiring segments on its four sides. Note that a connection routed with the General Purpose interconnect will incur significant routing delays because it must pass through a routing switch at each switch matrix. Connections that are required

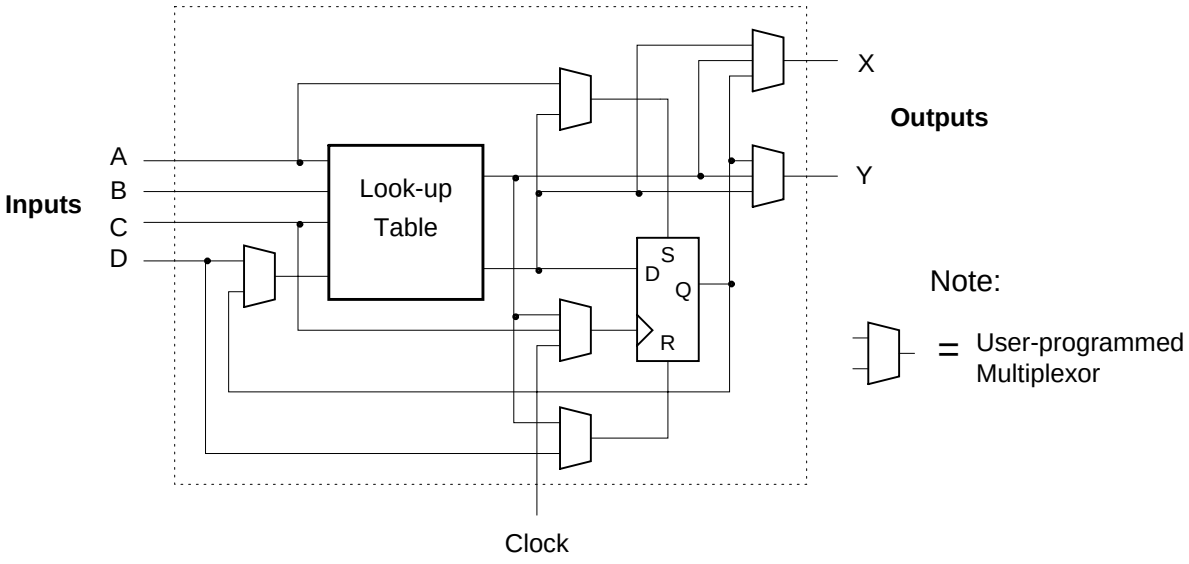

Figure 2.6 - *XC2000 CLB*

to reach several CLBs with low skew can use the Long Lines, which traverse at most one routing switch to span the entire length or width of the FPGA.

2.3.1.2 Xilinx XC3000

The XC3000 [Hsie88] is an enhanced version of the XC2000, featuring a more complex CLB and more routing resources. The CLB, as shown in Figure 2.8, houses a look-up table that can implement any function of five variables, any two functions of four variables, and some functions of up to seven variables. The CLB has two outputs, both of which may be either combinational or registered.

Figure 2.9 shows that the XC3000 routing architecture is similar to that in the XC2000, having Direct interconnect, General Purpose interconnect, and Long Lines. Each resource is enhanced: the Direct interconnect can additionally reach a CLB's left

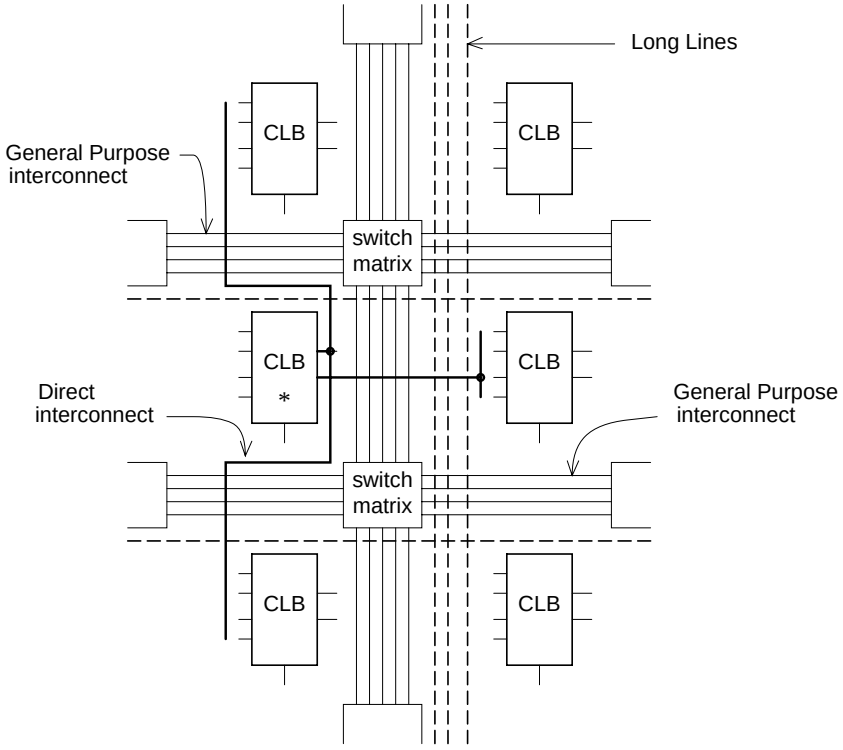

Figure 2.7 - *XC2000 Interconnect*

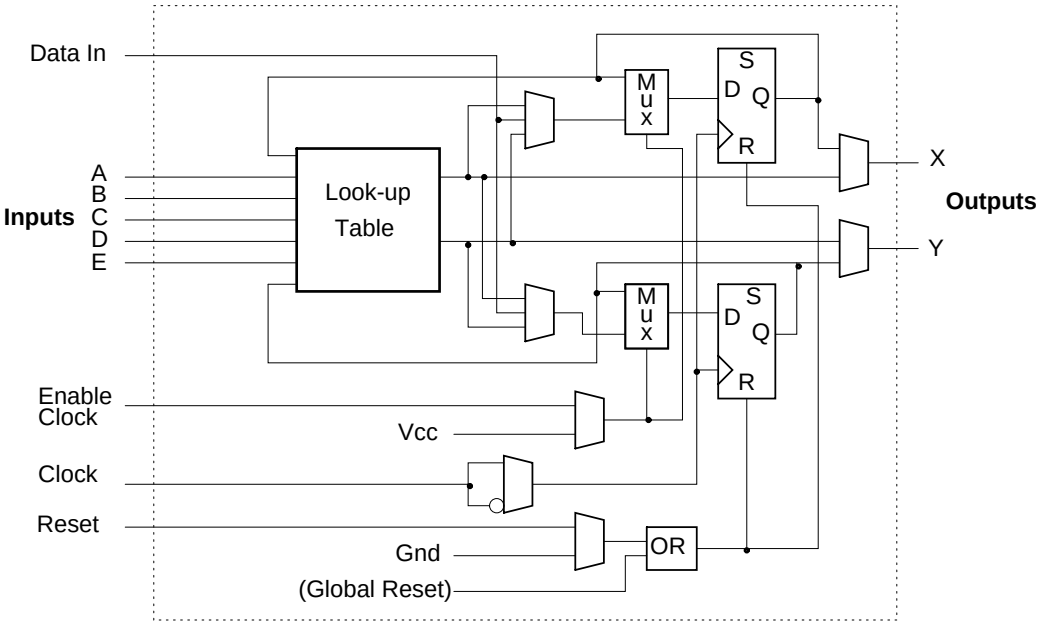

Figure 2.8 - *XC3000 CLB*

neighbour, the General Purpose interconnect has an extra wiring segment per row, and there are more Long Lines.

The XC3000 also contains switch matrices that are similar to those in the XC2000. Figure 2.9 depicts the internal structure of an XC3000 switch matrix by showing, as an example, that the wiring segment marked with an '*' can connect through routing switches to six other wiring segments. Although not shown in the figure, the other wiring segments are similarly connected, though not always to the same number of segments. This detail is included here because the results shown in Chapter 4 of this thesis suggest recommended values for the number of routing switches connectable to any wiring segment, as well as the number of wiring segments in a row or column. Those results indicate that, in terms of routability, the XC3000 contains too many routing switches per switch matrix and too few wiring segments in its rows and columns.

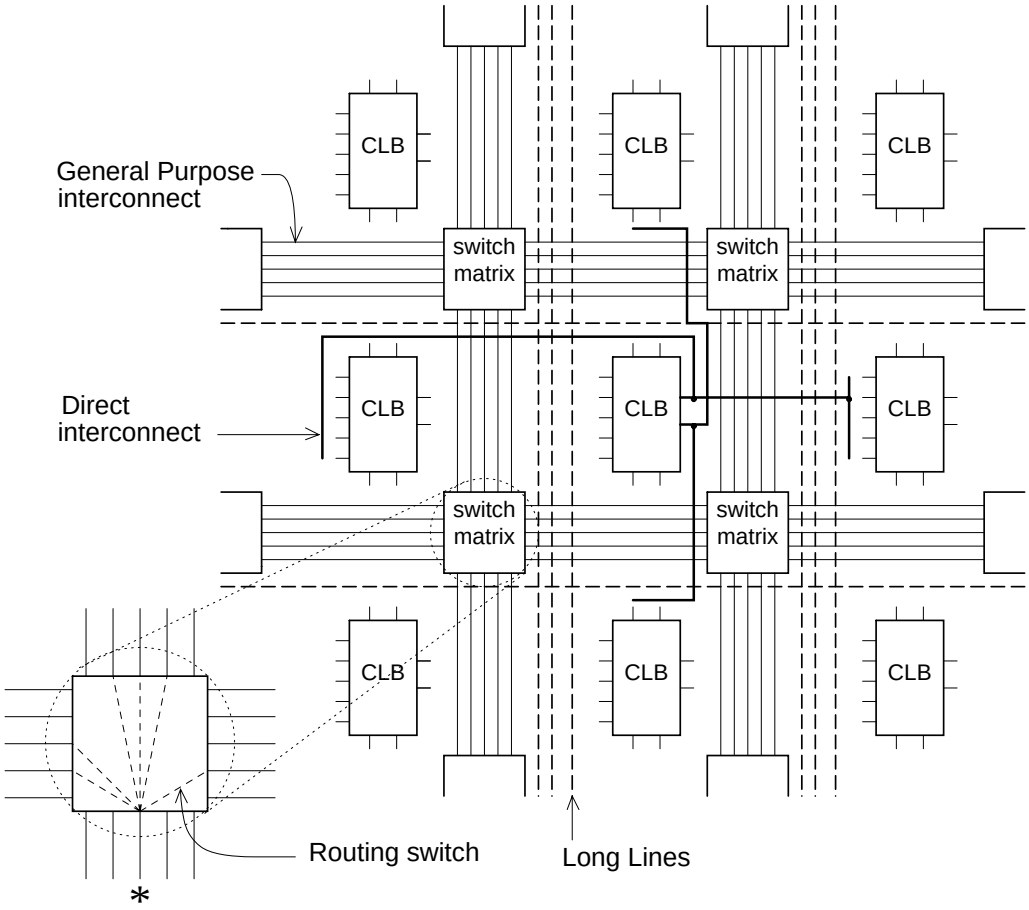

Figure 2.9 - *XC3000 Interconnect*

2.3.1.3 Xilinx XC4000

The XC4000 [Hsie90] features several enhancements over its predecessors. The CLB, illustrated in Figure 2.10, utilizes a hierarchical arrangement of look-up tables that yields a greater logic capacity per CLB than in the XC3000. The XC4000 CLB can implement two independent functions of four variables, any single function of five variables, any function of four variables together with some functions of five variables, or some functions of up to nine variables. The CLB has two outputs, which may be either combinational or registered.

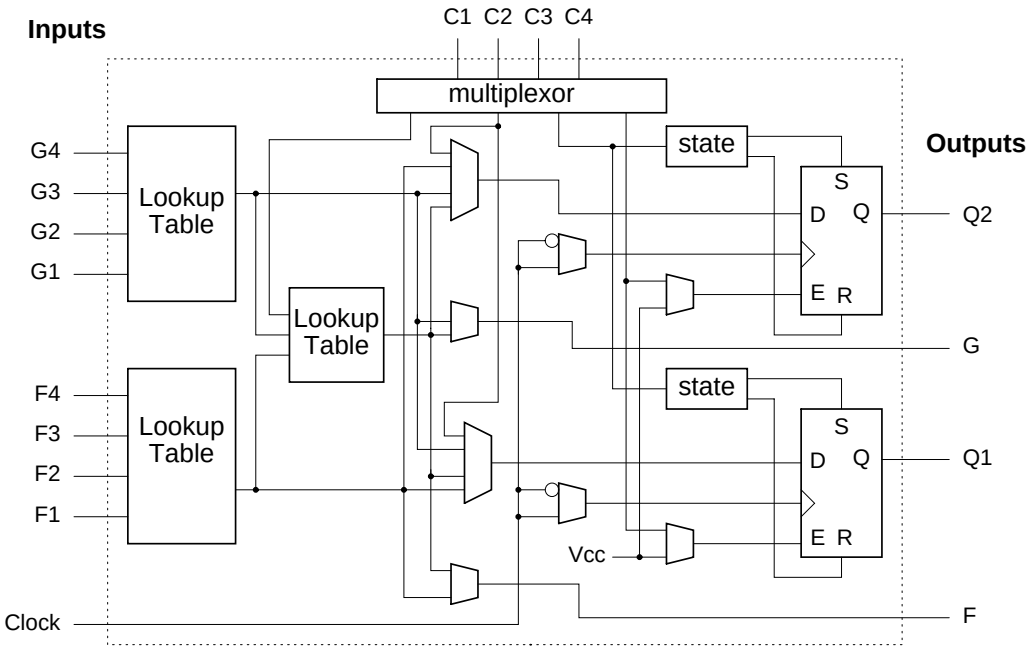

Figure 2.10 - *XC4000 CLB*

The XC4000 routing architecture is significantly different from the earlier Xilinx FPGAs, with the most obvious difference being the replacement of the Direct interconnect and General Purpose interconnect with two new resources, called Single-length Lines and Double-length Lines. The Single-length Lines, which are intended for relatively short connections or those that do not have critical timing requirements, are shown in Figure 2.11, where each X indicates a routing switch. This figure illustrates three architectural enhancements in the XC4000 series:

1. There are more wiring segments in the XC4000. While the number shown in the figure is only suggestive, the XC4000 contains more than twice as many wiring segments as does the XC3000.

2. Most CLB pins can connect to a high percentage of the wiring segments. This represents an increase in connectivity over the XC3000.

3. Each wiring segment that enters a switch matrix can connect to only three others, which is half the number found in the XC3000.

It is interesting to note these three enhancements here because they are all supported by the architectural research that appears in Chapter 4 of this thesis.

The remaining routing resources in the XC4000, which includes the Double-length Lines and the Long Lines, are shown in Figure 2.12. As the figure shows, the Double-length Lines are similar to the Single-length Lines, except that each one passes through half as many switch matrices. This scheme offers lower routing delays for moderately long connections that are not appropriate for the low-skew Long Lines. For clarity, neither the Single-length Lines nor the routing switches that connect to the CLB pins are shown in Figure 2.12.

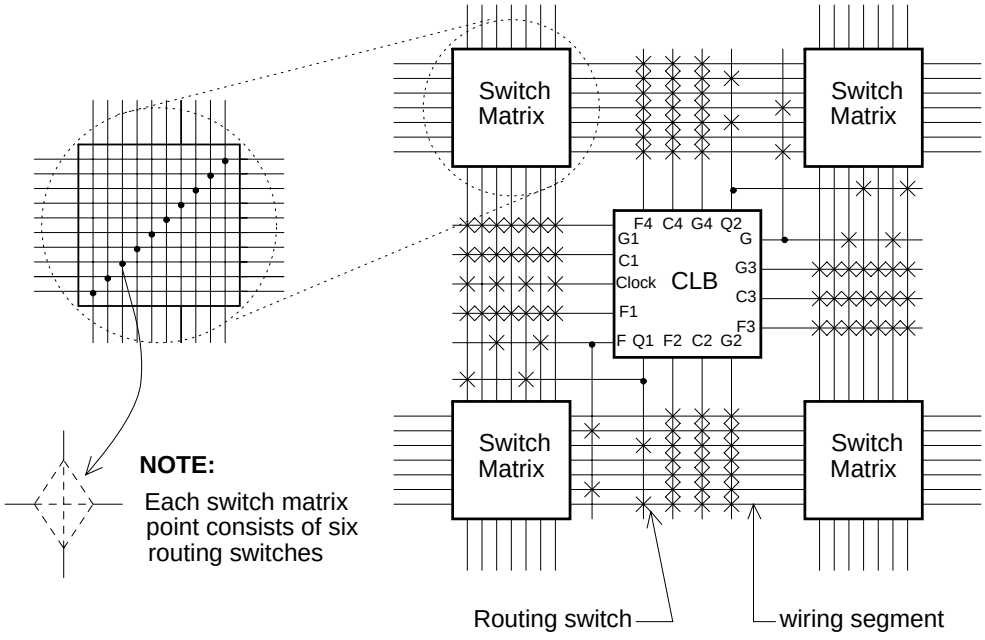

Figure 2.11 - *XC4000 Single-Length Lines*

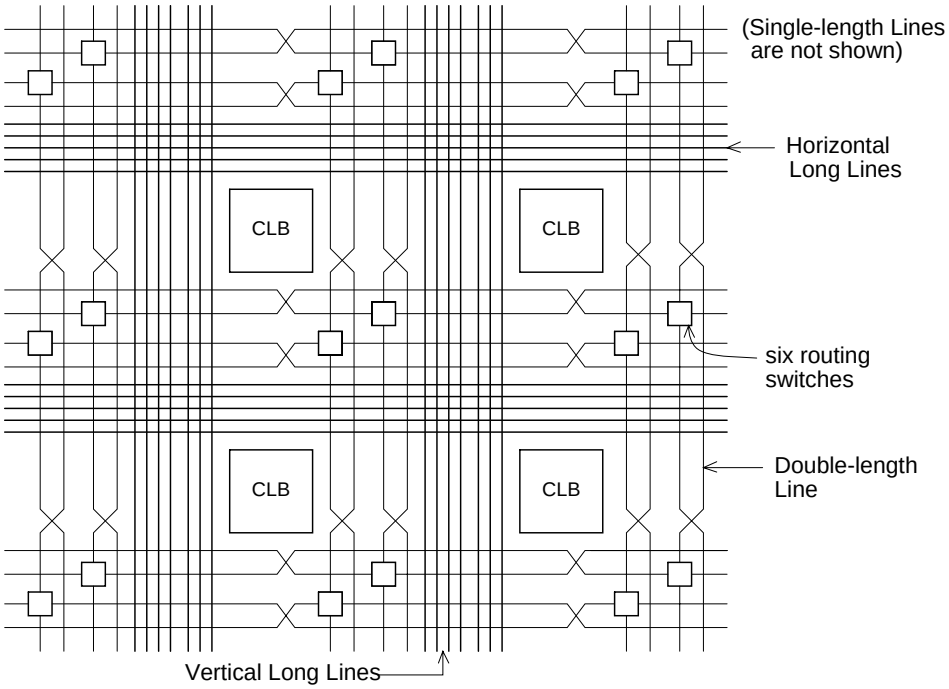

Figure 2.12 - *XC4000 Double-Length Lines and Long Lines*

2.3.1.4 Xilinx CAD Routing Tools

Xilinx routing tools are based on maze routers that are customized for the particular routing resources in each part. It was noted earlier in this chapter that maze routers are unable to consider the side effects that routing some connection in a particular fashion may have on other connections. This is a serious shortcoming because Xilinx routing structures have limited connectivity, and for this reason maze routing is probably not the best technique to use for Xilinx devices.

2.3.2 Actel FPGAs

The basic architecture of Actel FPGAs, depicted in Figure 2.13, is similar to that found in MPGAs, consisting of rows of programmable cells, called Logic Modules (LMs), with horizontal routing channels between the rows. Each routing switch in these FPGAs is implemented by a novel device called an anti-fuse [ElAy88], which normally

resides in a high-impedance state but takes on a low resistance (about 500 ohms) when "programmed" by a high voltage pulse. Actel currently has two generations of FPGAs, called the Act-1 [ElAy88] and Act-2 [Ahre90], whose logic capacities are shown in Table 2.2.

2.3.2.1 Actel Act-1

The Act-1 LM that is shown in Figure 2.14 illustrates a very different approach from that found in Xilinx FPGAs. Namely, while Xilinx utilizes a large, complex CLB, Actel advocates a small, simple LM. Research has shown [Sing91] that both of these approaches have their merits, and the best choice for a programmable cell depends on the

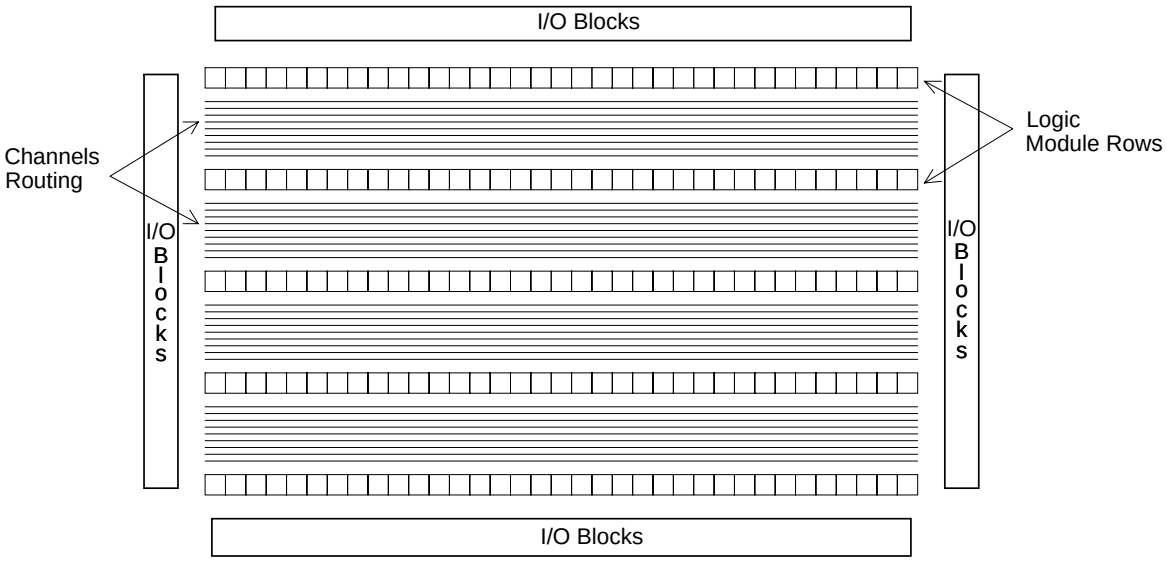

Figure 2.13 - *General Architecture of Actel FPGAs*

Series	Number of LMs	Equivalent Gates
Act-1	295 - 546	1200 - 2000
Act-2	430 - 1232	6250 - 20000

Table 2.2 - *Actel FPGA Logic Capacities*

speed performance of the routing architecture. As Figure 2.14 shows, the Act-1 LM is based on a configuration of multiplexers, which can implement any function of two variables, most functions of three, some of four, up to a total of 702 logic functions [Mail90].

The Act-1 routing architecture is illustrated in Figure 2.15, which for clarity shows only the routing resources connected to the LM in the middle of the picture. The Act-1 employs four distinct types of routing resources: Input segments, Output segments, Clock tracks, and Wiring segments. Input segments connect four of the LM inputs to the Wiring segments above the LM and four to those below, while an Output segment connects the LM output to several channels, both above and below the module. The Wiring segments consist of straight metal lines of various lengths that can be connected together through anti-fuses to form longer lines. The Act-1 features 22 tracks of Wiring segments in each routing channel and, although not shown in the figure, 13 vertical tracks that lie directly on top of each LM column. Clock tracks are special low-delay lines that are used for signals that must reach many LMs with minimum skew.

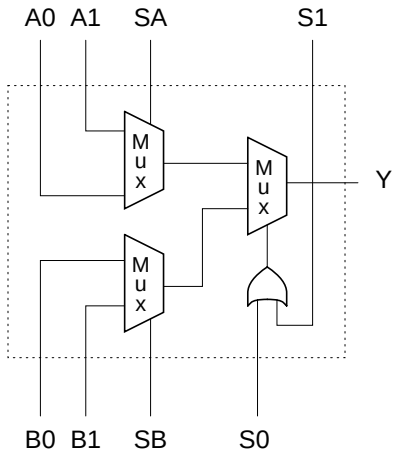

Figure 2.14 - *Act-1 LM*

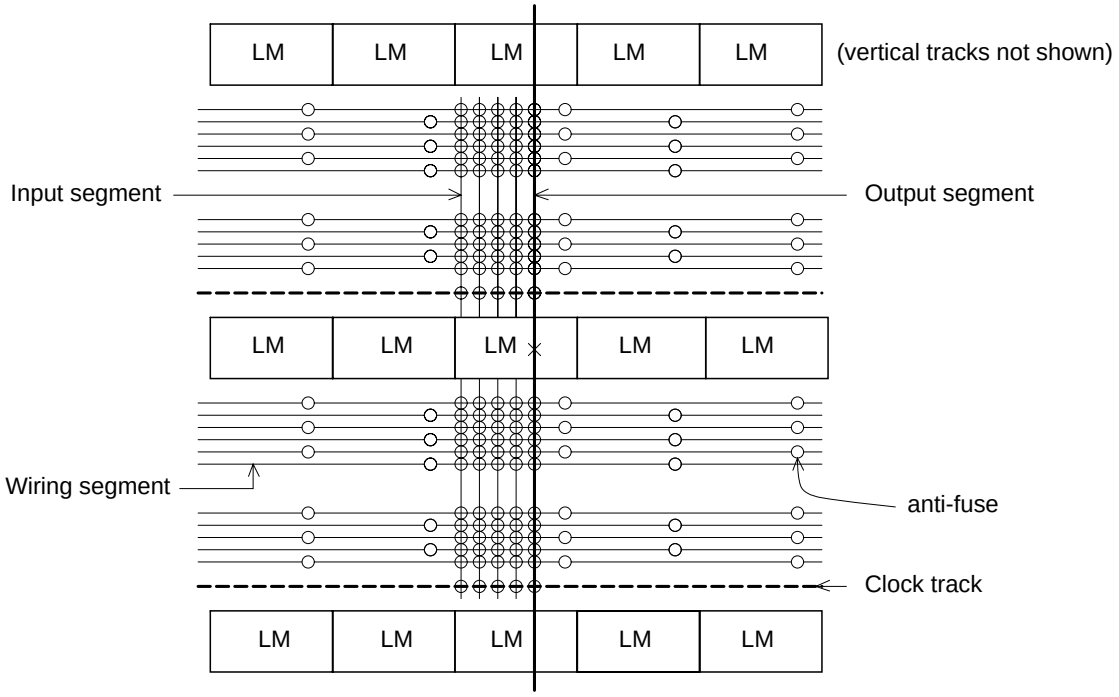

Figure 2.15 - *Act-1 Programmable Interconnect Architecture*

2.3.2.2 Actel Act-2

The Act-2 device, an enhanced version of the Act-1, contains two different programmable cells, called the C (Combinational) module and the S (Sequential) module. The C module is very similar to the Act-1 LM, although slightly more complex, while the S module is optimized to implement sequential elements.

The Act-2 routing architecture is also similar to that found in the Act-1. It features the same four types of routing resources, but the number of tracks is boosted to 36 in each routing channel and 15 in each column.

2.3.2.3 Actel CAD Routing Tools

The key CAD tool that is used to route Actel FPGAs is the segmented channel router described in [Green90]. This router uses a novel algorithm that guarantees that every connection will pass through at most a given maximum number of anti-fuses, if

such a solution exists, and in this sense the algorithm produces an optimal result. Although channel routers are not generally appropriate for FPGAs, for reasons given in Chapter 3, it is possible to use this technique for Actel designs because of their high connectivity. Every LM input connects to all of the tracks either above or below it and each LM output connects to all the tracks in the channels spanned by its output segment. However, it is worthy of note that the research reported in Chapter 4 of this thesis indicates that this connectivity can be reduced, in which case it might be necessary to modify the routing algorithm to handle the reduced horizontal-vertical connectivity.

2.3.3 Altera FPGAs

Altera FPGAs [Alt90] are considerably different from the others discussed above because they resemble large Programmable Logic Devices. Nonetheless, they are functionally equivalent to FPGAs because they employ a two-dimensional array of programmable cells and a programmable routing structure, they can implement multi-level logic, and they are user-programmable. Altera's general architecture, which is based on an EPROM programming technology, is illustrated in Figure 2.16. It consists of an array of programmable cells, called Logic Array Blocks (LABs), interconnected by a routing resource called the Programmable Interconnect Array (PIA). The logic capacities of the two generations of Altera FPGAs are listed in Table 2.3.

The Altera LAB is by far the most complex logic cell of any of the FPGA families described thus far. A LAB can be thought of as an efficient PLD, as will be explained in the following paragraphs. Each LAB, as seen in Figure 2.17, consists of two major blocks, called the Macrocell Array and the Expander Product Terms.

The Macrocell Array is a one-dimensional array of elements called Macrocells, where the number of elements in the array varies with each Altera device. As illustrated

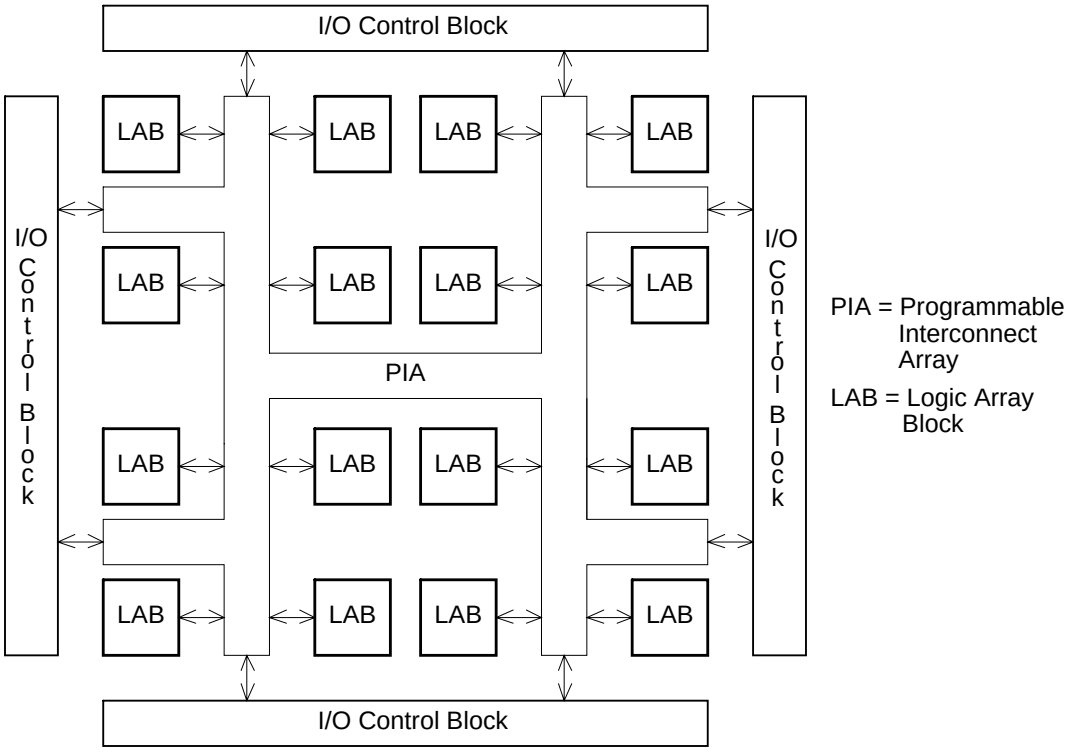

Figure 2.16 - *General Architecture of Altera FPGAs*

in Figure 2.18, each Macrocell comprises three wide AND gates that feed an OR gate which connects to an XOR gate, and a flip-flop. The XOR gate generates the Macrocell output and can optionally be registered. In Figure 2.18, the inputs to the Macrocell are shown as single-input AND gates because each is generated as a wired-AND (called a p-term) of the signals drawn on the left-hand side of the figure. A p-term can include any signal in the PIA, any of the LAB Expander Product Terms (described below), or the output of any other Macrocell. With this arrangement the Macrocell Array functions much like a PLD, but with fewer product terms per register (there are usually at least eight product terms per register in a PLD). Altera claims [Alt90] that this makes the LAB more efficient because most logic functions do not require the large number of p-terms found in PLDs and the LAB supports wide functions by way of the Expander Product Terms.

Series	Number of LABs	Equivalent Gates
EPM5000	1 - 12	2000 - 7500
EPM7000	N/A	2000 - 20000

Table 2.3 - *Altera FPGA Logic Capacities*

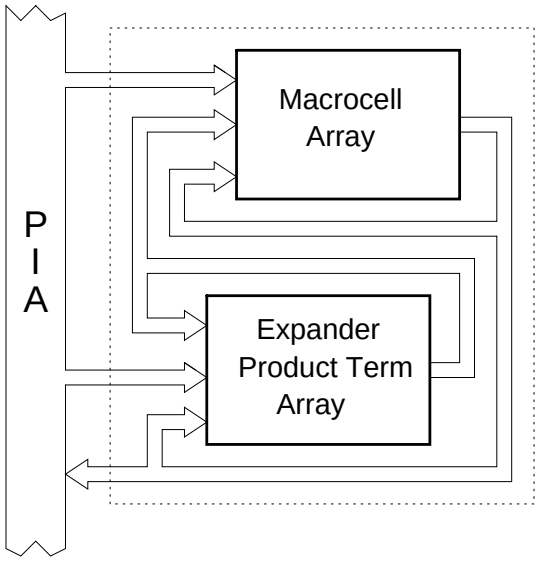

Figure 2.17 - *Altera LAB*

As illustrated in Figure 2.19, each Expander Product Terms block consists of a number of p-terms (the number shown in the figure is only suggestive) that are inverted and fed back to the Macrocell Array, and to itself. This arrangement permits the implementation of very wide logic functions because any Macrocell has access to these extra p-terms.

The Altera routing structure, the PIA, consists of a number of long wiring segments that pass adjacent to every LAB. The PIA provides complete connectivity because each LAB input can be programmably connected to the output of any LAB, without constraints. With this arrangement, routing an Altera FPGA is trivial, since there are no routing constraints. However, as mentioned previously for Actel FPGAs, this level of

connectivity is excessive and could probably be reduced, given an appropriate routing algorithm.

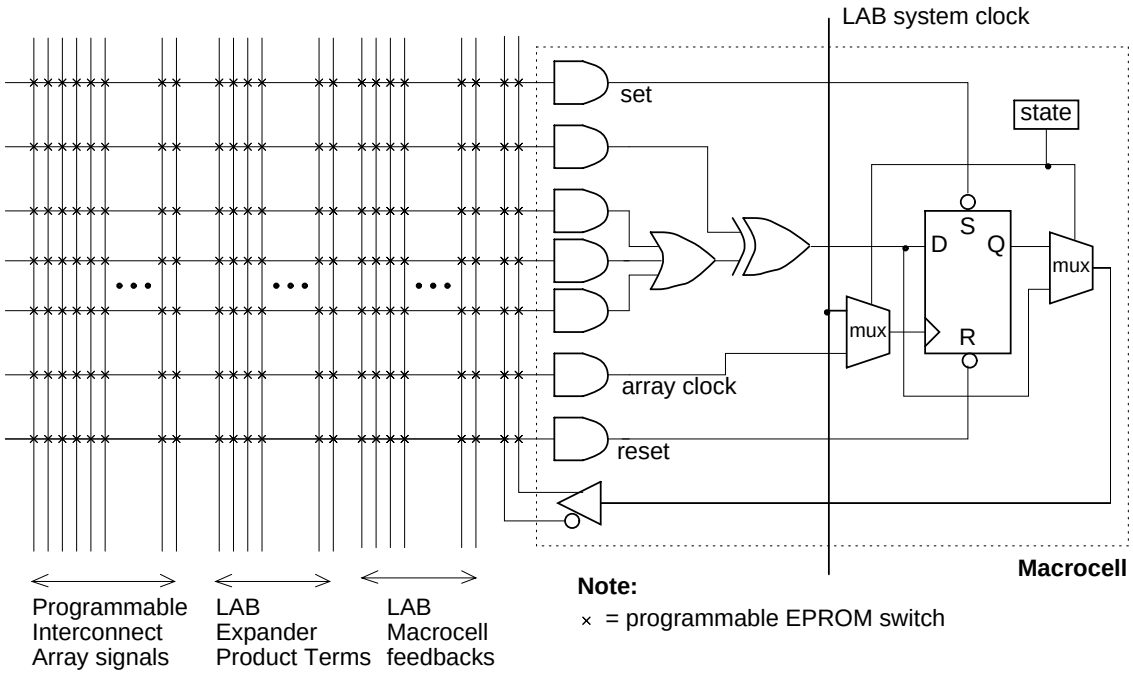

Figure 2.18 - *Altera Macrocell*

2.3.4 Other FPGAs

Four recently introduced FPGAs are described briefly in this section, including those from Plessey Co., Plus Logic, Advanced Micro Devices, and Quicklogic.

2.3.4.1 Plessey FPGAs

The Plessey FPGA, described in [Ples89], is called an Electrically Reconfigurable Array. It consists of a two-dimensional array of logic cells overlayed with a dense interconnect resource. With the routing resources placed on top of the logic cells, these devices resemble the Sea-Of-Gates architecture used in some MPGAs. Each Plessey logic cell is relatively simple, containing an eight-to-two line multiplexer that feeds a NAND gate, and a transparent latch. The multiplexer is controlled by a Static RAM block and is

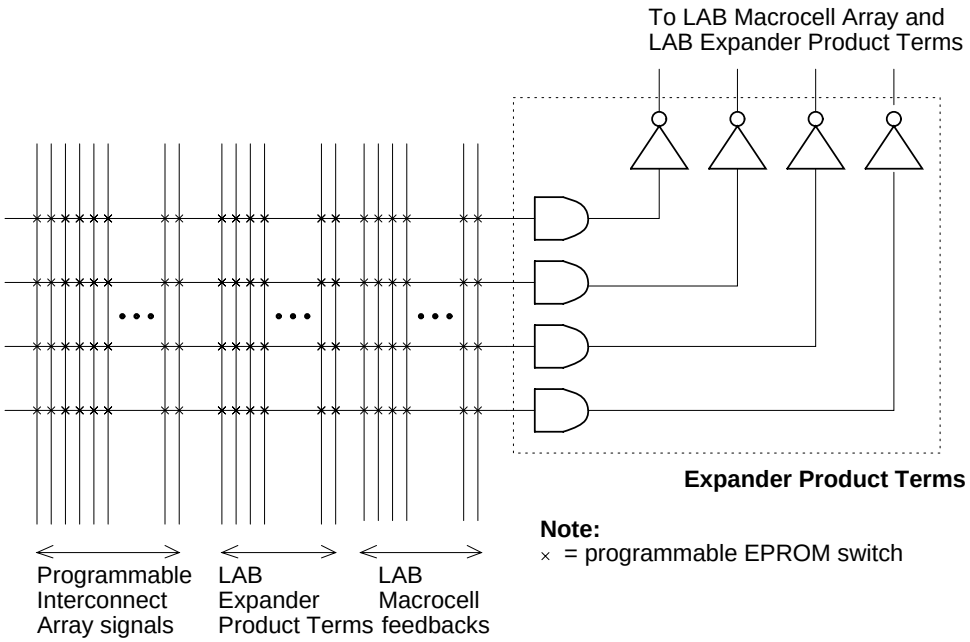

Figure 2.19 - *Altera Expander Product Terms*

used to connect the logic cell to the routing resources, which comprise wiring segments of various lengths: Local interconnect for short connections, Short Range interconnect for moderate-length connections, and Long Range interconnect for long connections.

2.3.4.2 Plus Logic FPGAs

The Plus Logic FPGA [Plus90] consists of two columns of four logic cells, called Functional Blocks (FBs), that can be fully interconnected by a Universal Interconnect Matrix (a full cross-bar switch). Compared to the three FPGA architectures that were described in detail at the first of this section, this device is most like an Altera FPGA, but the FBs represent more complex logic cells. Each FB comprises a wide AND plane that feeds an OR plane, like a PLA (programmable AND/programmable OR) device. The OR plane feed a third plane, which generates the nine (optionally registered) outputs of an FB. Each of these outputs corresponds to any function of two terms from the OR array and one output of any other FB. The programming technology used by Plus is EPROM.

3 A Detailed Router for Field-Programmable Gate Arrays

3.1 Introduction

This Chapter presents a new kind of detailed routing algorithm that has been designed specifically for FPGAs. The algorithm is unique in that it approaches this problem in a general way, allowing its use over a wide range of different FPGA routing architectures [Brow90] [Brow91]. This feature is used in Chapter 4, where the router is employed to investigate the effect of the flexibility of routing architectures.

Detailed routing for FPGAs can be more difficult than classic detailed routing [Aker72] [Souk81] [Loren89] because connections are made using wiring segments that are already in place and joins between segments are possible only at pre-determined places where routing switches exist. In some FPGA routing architectures the amount of connectivity available is low, which places exacting limitations on the number of routing choices for a connection. Wherever two or more connections pass through a common routing channel, there may be competition for the routing resources in that channel. In FPGAs that have limited connectivity, resolving such competitions is essential in order to achieve 100 percent routing completion. The algorithm described here, called the Coarse Graph Expansion (CGE) detailed router for FPGAs, addresses the issue of scarce routing resources by considering the side effects that the routing of one connection has on another, and also has the ability to optimize the routing delays of time-critical connections.

CGE has been used to obtain excellent routing results for several industrial circuits implemented in FPGAs with various routing architectures. The results show that CGE is able to route relatively large FPGAs in very close to the minimum number of tracks as

determined by global routing, and it can successfully optimize the routing delays of time-critical connections.

This chapter is organized as follows: Section 3.2 motivates the development of an FPGA-specific router, Section 3.3 presents the model used for the FPGA, Section 3.4 defines the detailed routing problem, Section 3.5 describes the CGE routing algorithm, Section 3.6 presents the results from tests of the router, and Section 3.7 gives concluding remarks.

3.2 Motivation

A key problem in the detailed routing of FPGAs is that routing choices made for one connection may unnecessarily block another. Consider Figure 3.1, which shows three views of the same section of an FPGA. Each view gives the routing options for one of connections A, B, and C. In the figure, a routing switch is shown as an X, a wiring segment as a dotted line, and a possible route as a solid line. Now, assume that a router first completes connection A. If the wiring segment numbered 3 is chosen for A, then one of connections B and C cannot be routed because they both rely on the same single remaining option, namely the wiring segment numbered 1. The correct solution is for the router to choose the wiring segment numbered 2 for connection A, in which case both B and C are also routable. Note that in a regular VLSI channel with full customization of mask layers, this scenario is not a problem because any of segments 1, 2, or 3 could be used for any of connections A, B, or C. Although this is a simple example, it illustrates the essence of the problems that occur because of limited routing options in FPGAs.

Common approaches used for detailed routing in other types of devices are not suitable for FPGAs. Maze routers [Lee61] are ineffective because, as shown in Chapter 2, they are inherently sequential and so, when routing one connection, they cannot consider

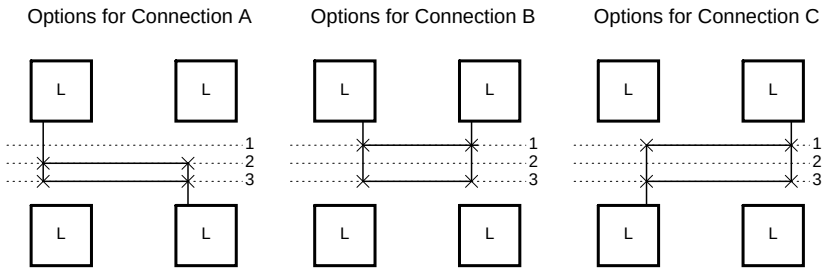

Figure 3.1 - *Routing Conflicts*

the side-effects on other connections. Channel routers [Hash71] are not appropriate because the detailed routing problem in FPGAs cannot generally be subdivided into independent channels. Note that a channel routing algorithm is used in [Green90] for Actel-like FPGAs [ElGa88]. This is possible for these types of FPGAs because the logic cells are arranged in rows separated by routing channels and the routing switches are such that each vertical wiring segment (from a logic cell pin or from another channel) can be connected to any horizontal wiring segment that it crosses in a channel. This routing flexibility cannot be assumed, in general, for an FPGA.

3.3 The FPGA Model

Since the primary purpose of this algorithm is to provide a means of investigating FPGA routing architectures, an appropriate model must be defined for the FPGA. The model that has been chosen has a two-dimensional array of logic cells interconnected by vertical and horizontal routing channels, similar to [Cart86]. Note that this model is much more flexible than the FPGA presented in [Cart86] because it allows the amount of routing resources to be changed over a wide range. The model comprises three major parts: the logic cells (L), Connection blocks (C), and Switch (S) blocks, as shown in Figure 3.2. The logic cells house the combinational and sequential logic that form the functionality of a circuit. In general, a logic cell has a number of pins that may each connect

to the four adjacent C blocks. The FPGA's I/O cells appear as logic cells that are on the periphery of the chip.

The C blocks are rectangular switch boxes with connection points on all four sides, and are used to connect the logic cell pins to the routing channels, via programmable switches. Depending on the topology of the C block, each logic cell pin may be switchable to either all or some fraction of the wiring segments that pass through the C block. The fewer wiring segments connectable in the C blocks, the harder the FPGA is to route. Connections along a routing channel may also pass straight through a C block, but in a typical routing architecture no switch would be involved for such connections.

The S blocks are also rectangular switch boxes. They are used to connect wiring segments in one channel segment to those in another. Depending on the topology, each wiring segment on one side of an S block may be switchable to either all or some fraction

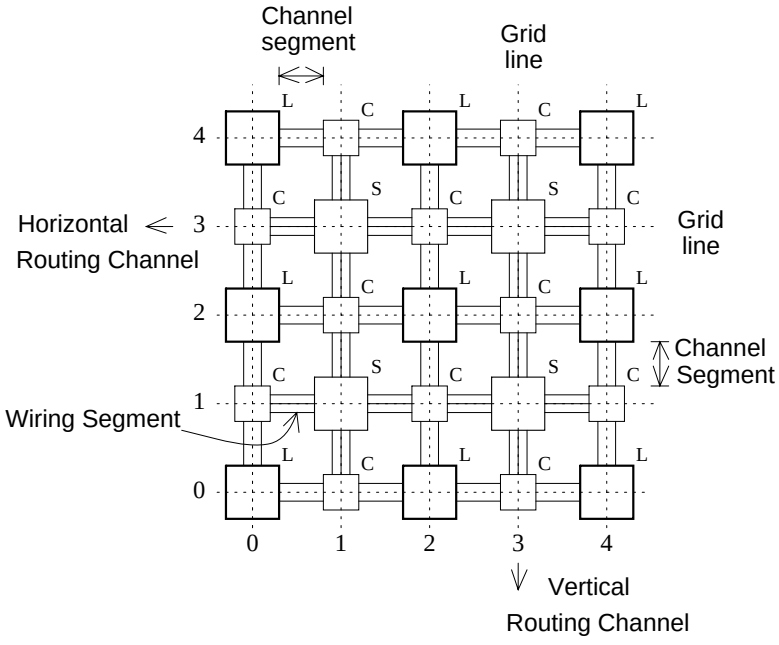

Figure 3.2 - *The FPGA Model*

of the wiring segments on each other side of the S block. Again, the fewer wiring segments that can be switched to, the harder the FPGA is to route. A connection that passes through an S block may do so through a switch or it may be hard-wired. A connection will have a lower routing delay if it uses hardwired wiring segments than if it passes through switches.

In Figure 3.2, each logic cell has two pins that appear on all four of its sides, and there are three *tracks* in each routing channel. The figure also defines several terms, such as *channel segment*, *wiring segment*, and *routing channel*. The two-dimensional grid that is overlayed on the FPGA is used in this chapter as a means of describing the connections to be routed.

3.4 General Approach and Problem Definition

FPGA routing is a complex combinatorial problem. The general approach taken here is the usual two-stage method of global routing followed by detailed routing. This allows the separation of two distinct problems: balancing the densities of all routing channels, and assigning specific wiring segments for each connection. The global router used is an adaptation of the LocusRoute global routing algorithm for standard cells, that was described in Chapter 2. The global router divides multi-point nets into two-point connections and routes them in minimum distance paths. Its main goal is to distribute the connections among the channels so that the channel densities are balanced.

The global router defines a coarse route for each connection by assigning it a sequence of channel segments. Figure 3.3a shows a representation of a typical global route for one connection. It gives a sequence of channel segments that the global router might choose to connect some pin of a logic cell at grid location 2,2 to another at 4,4. The global route is called a *coarse graph*, $G(V,A)$, where the logic cell at 2,2 is referred

to as the root of the graph and the logic cell at 4,4 is called the leaf. The vertices, *V*, and edges, *A*, of *G*(*V*,*A*) are identified by the grid of Figure 3.2. Since the global router splits all nets into two-point connections, the coarse graphs always have a fan-out of one.

After global routing the problem is transformed to the following: for each two-point connection, the detailed router must choose specific wiring segments to implement the channel segments assigned during global routing. As this requires complete information about the FPGA routing architecture, CGE uses the details of the logic cells, C blocks, and S blocks, as described in the following sections.

3.5 The CGE Detailed Router Algorithm

The basic algorithm is split into two phases. In the first phase, it records a number of alternatives for the detailed route of each coarse graph, and then in the second phase, viewing all the alternatives at once, it makes specific choices for each connection. The decisions made in phase 2 are driven by a cost function that is based on the alternatives enumerated in phase 1. Multiple iterations of the two phases are used to allow the algorithm to conserve memory and run-time while converging to its final result, as discussed in Section 3.5.3.

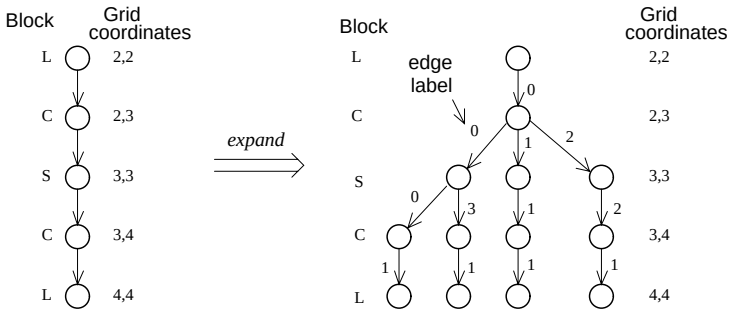

Figure 3.3a. Coarse graph, *G*

Figure 3.3b. Expanded graph, *D*

Figure 3.3 - *A Typical Coarse Graph and its Expanded Graph*

3.5.1 Phase 1: The Expansion of the Coarse Graphs

During phase 1, CGE *expands* each coarse graph and records a subset of the possible ways that the connection can be implemented. For each $G(V,A)$, the expansion phase produces an *expanded* graph, called $D(N,E)$. N are the vertices of D and E are its edges, with each edge referring to a specific wiring segment in the FPGA. The edges are labelled with a number that refers to the corresponding wiring segment.

In the expansion algorithm, the procedures that define the connection topology of the C and S blocks are treated as *black-box* functions. The black-box function for a C block is denoted as $f_c([d_1,d_2,l],d_3)$ and for an S block as $f_s([d_1,d_2,l],d_3)$. The parameters in square brackets define an edge that connects vertex d_1 to vertex d_2, using a wiring segment labelled l. Such an edge is later referred to as e, where $e = (d_1,d_2,l)$. The parameter d_3 is the successor vertex of d_2 in G. The task of the function call can be stated as: "If the wiring segment numbered l is used to connect vertex d_1 to d_2, what are the wiring segments that can be used to reach d_3 from d_2?" The function call returns the set of edges that answer this question. As explained in Section 3.5.4, this black-box approach provides independence from any specific FPGA routing architecture. The result of a graph expansion is illustrated in Figure 3.3b, which shows a possible expanded graph for the coarse graph of Figure 3.3a. An expanded graph is produced by examining the routing switches and wiring segments along the path described by the coarse graph, and recording the alternative detailed routes in the expanded graph. In algorithmic form, the graph expansion process for each coarse graph operates as follows:

Create D and give it the same root as G. Make the immediate successor to the root of D the same as for the root of G.
While traversing D breadth first, enumerate the paths originating at each added vertex according to:
Expand a C vertex in D by calling $Z = f_c(e_C,n)$. e_C is the edge in D that connects to C from its predecessor. n is the required successor vertex

of *C* (in *G*) and *Z* is the set of edges returned by $f_c(\,)$. The call to $f_c(\,)$ adds Z to *D*.

Expand an *S* vertex in *D* by calling $Z = f_s(e_S, n)$. e_S is the edge in *D* that connects to *S* from its predecessor. *n* is the required successor vertex of *S* (in *G*) and *Z* is the set of edges returned by $f_s(\,)$. The call to $f_s(\,)$ adds Z to *D*.

Endwhile

3.5.2 Phase 2: Connection Formation

After expansion, each $D(N,E)$ may contain a number of alternative paths. CGE places all the paths from all the expanded graphs into a single path list. Based on a cost function, the router then selects paths from the list; each selected path defines the detailed route of its corresponding connection. Phase 2 proceeds as follows (as explained later in this section, the terms c_f cost and c_t cost are functions that represents the relative cost of selecting a specific detailed route (path) for a connection, and an *essential* path indicates a connection that should be routed immediately because it has only one remaining option):

Put all the paths in the expanded graphs into the path-list

While the path-list is not empty
- **If** there are paths in the path-list that are known to be essential
  - **Select** the essential path that has the lowest c_f cost.
- **Else if** there are paths in the path-list that correspond to time-critical connections
  - **Select** the critical path with the lowest c_t cost.
- **Else**
  - **Select** the path with the lowest c_f cost
- **Mark** the graph corresponding to the selected path as routed - remove all paths in this graph from the path-list.
- **Find** all paths that would conflict with the selected path and remove them from the path list (see Note). If a connection loses all of its alternative paths, re-expand its coarse graph - if this results in no new paths, the connection is deemed unroutable (see Section 3.5.3.1 for a discussion relating to failed connections).
- **Update** the cost of all affected paths.

Endwhile

Note: When a wiring segment is chosen for a particular connection, it and any other wir-

ing segments in the FPGA that are hardwired to it must be eliminated as possible choices for connections that are in other nets. This requires a function analogous to $f_c(\,)$ and $f_s(\,)$ that understands the connectivity of a particular FPGA configuration. CGE calls this routine *update* (e) - the parameter *e* is an edge in the selected path and *update* (e) returns the set of edges that are hardwired to *e*.

3.5.2.1 Cost Function Design

Because the cost function allows it to consider all the paths at once, CGE can be said to route the connections 'in parallel'. Each edge in the expanded graphs has a two-part cost: $c_f\,(e)$ accounts for the competition between different nets for the same wiring segments, and $c_t\,(e)$ is a number that reflects the routing delay associated with the wiring segment. Each path has a cost that is simply the sum of the costs of its edges. CGE selects paths based on the c_t cost only if the path corresponds to a time-critical connection. Otherwise, paths are selected according to their c_f cost.

The c_f cost has two goals:

1. To select a path that has a relatively small negative effect on the remaining connections, in terms of routability. The cost deters the selection of paths that contain wiring segments that are in great demand. The reason for using wiring segment demand was illustrated in Figure 3.1, where connection A should be routed with wiring segment number 2, because wiring segment number 3 is in greater demand.
2. It is used to identify a path that is *essential* for a connection. A path is called essential when it represents the only remaining option in the FPGA for a connection, because previous path selections have consumed all other alternatives.

The importance of essential wiring segments is illustrated by the example in Figure 3.4. If the router were to complete connection D first, then wiring segment number 1 or 2 would be equal candidates according to their demand, since they both appear in one other graph. However, wiring segment number 1 is essential for the completion of connection E and to ensure the correct assignment of the essential wiring segment, connection E should be routed first.

To determine whether an edge, *e*, is in great demand the router could simply count the number of occurrences of *e* that are in expanded graphs of other nets. However, some occurrences of *e* are less likely to be used than others because there may be alternatives (edges in parallel with *e*). Thus, the c_f cost of an edge *e* that has *j* other occurrences (e_1, e_2, ..., e_j) is defined as

$$c_f(e) = \sum_j \frac{1}{alt(e_j)},$$

where $alt(e_j)$ is the number of edges in parallel with e_j.

Because of the summing process in $c_f(e)$, the more graphs *e* occurs in, the higher will be its cost. This reflects the fact that *e* is an edge that is in high demand and urges CGE to avoid using *e* when there are other choices. Note that an edge that only appears in its own graph will have a c_f of 0. For the special case when $alt(e_j)$ is 0, e_j is an edge

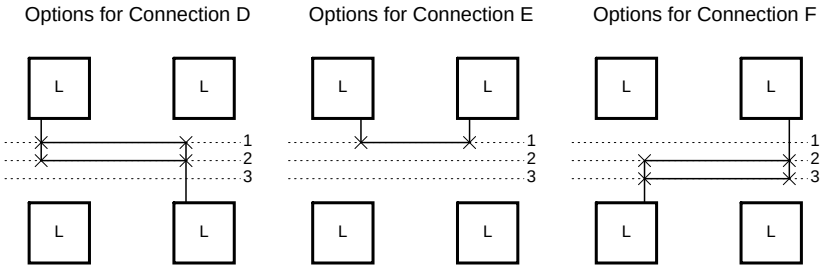

Figure 3.4 - *An Essential Wiring Segment*

that is essential to the associated connection because there are no alternatives. In this case, any path in the graph that uses e_j is identified as *essential*. When the calculation of a cost reveals that a path is essential, CGE gives that path the highest priority for routing.

3.5.3 Controlling Complexity

Although the above description of graph expansion implies that all possible paths in an FPGA are recorded during expansion, this is not practical because the number of paths can be very large in some architectures. For example, consider the connection of two pins on two different L blocks. Assume that each pin can connect to F_c of the wiring segments in the channel segments adjacent to each logic cell, and that the logic cells are separated by n Switch blocks. If each wiring segment that enters one side of a Switch block can connect to F_s wiring segments on the other three sides, then there are an average of $F_c\left(\frac{F_s}{3}\right)^n$ different paths from the first pin to the last logic cell, and assuming W tracks in each routing channel, there are an average of $\frac{F_c^2}{W}\left(\frac{F_s}{3}\right)^n$ possible ways to form the connection. Since typical values of F_s are three or greater, as shown in Chapter 4, and the number of connections is large, a heuristic is employed to reduce the number of paths in the expanded graphs. Some of the paths are pruned as each graph is expanded. The pruning procedure is parameterized so that the number of paths is controlled and yet the expanded graphs still contain as many alternatives as possible. Maximizing the number of alternatives is important in the context of resolving routing conflicts. The pruning procedure is part of the graph expansion process that is described in Section 3.5.1. The general flow follows (the criteria used for pruning is given at the end of this section):

Expand two levels
Prune; keep at most *K* vertices at this level, and assign each a unique group number. Discard the other vertices and the paths they terminate.
Expand two more levels. Assign each added vertex the group number of its predecessor.
While the leaf level has not been reached.
    Prune; keep at most *k* vertices with each group number at this level. Discard the other vertices and the paths they terminate.
    Expand two more levels. Assign each added vertex the group number of its predecessor.
Endwhile

The graphs are pruned every two levels because that is where fanout occurs (after the first C block and after every S block). The parameter *K* controls the starting widths of the graphs and can take values from one to F_c (the number of wiring segments connected to each logic cell pin). Beyond the maximum value of *K*, parameter *k* allows the expanded graphs to further increase in width. The concept of *group* numbers isolates each of the original *K* paths, which maximizes the number of alternatives at each level of the final expanded graph. The actual values used for *K* and *k* are discussed in the next section. The effect of the pruning algorithm is illustrated in Figure 3.5. The left half of the figure shows a fully expanded graph from an example circuit, while the corresponding pruned graph is on the right. Also shown are each graph's edges in the FPGA.

The choice to prune a vertex is based on the wiring segment that corresponds to its incoming edge, as follows. For the special case of time-critical connections, the wiring segments with the least delay are favored. For other connections, the wiring segments that have thus far been included in the most other expanded graphs will be discarded. This helps the c_f cost function discover the wiring segments that are in the least demand. Note that this introduces an order-dependence in the routing algorithm because the paths that are pruned from each expanded graph depend on the order in which the coarse graphs are expanded.

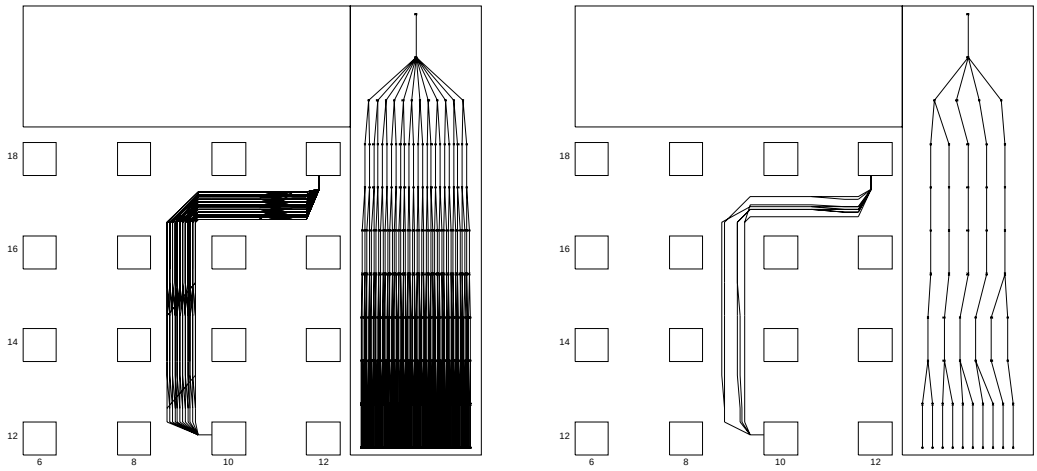

Figure 3.5 - *The Effect of Pruning*

Note that when paths are discarded because of pruning, they are not necessarily abandoned permanently by the router. In phase 2, as CGE chooses connections, if routing conflicts consume all the alternatives for some graph, CGE re-invokes the graph expansion process to obtain a new set of paths if some exist.

3.5.3.1 Iterative Improvements

This section explains how iterations of the two phases of CGE are used to conserve memory and run-time. The iterative approach is linked to the pruning parameters of the graph expansion phase. Setting the pruning parameters to large values allows the router to do a better job of resolving routing conflicts because it sees many alternatives for each connection. On the other hand, with large pruning parameters more memory and longer run-time are required by the algorithm. The key to this routing quality versus memory and time trade-off is the realization that most connections in an FPGA are relatively easy to route and only a small percentage of the connections pose real difficulties. This is because, in a typical routing problem, there are only a few channel segments whose densities are very close to the total number of wires in a routing channel. To exploit this

property, the router starts with small pruning parameters and then increases them through successive iterations, but only for the parts of the FPGA that are difficult to route.

For the first iteration the pruning parameters are set to relatively small values, and the entire FPGA is routed. If routing conflicts leave some connections unrouted, then another iteration is required. The procedure is to erase all the routing of any connection that overlaps any part of a failed connection, and then to attempt to route those channel segments again using larger pruning parameters. Only connections that touch some segment of a channel in which a failed connection occurred are re-routed in the next iteration. Iterations are continued until all connections are routed or until further improvements are not forthcoming. Note that at this point it would be desirable to try different global routes for connections that are left unrouted after all iterations, but no such failure-recovery mechanism is currently implemented. This iterative approach is a minor variation of classic rip-up and re-route schemes where individual connections would be removed and re-routed to try to resolve routing conflicts. The technique employed here allows the algorithm's cost function to solve the routing problem, but conserve memory and time where the problem is not difficult and expend them only where it is required.

The specific values used for the pruning parameters in each iteration affect the total number of iterations required, but do not appreciably affect the quality of the final result. This indicates a robustness in the algorithm because the quality of the routing does not depend on the specific values chosen for the program's parameters. For the results that are presented in Section 3.6, K and k are set to two for the first iteration. K is increased by one for each iteration until it reaches F_c, after which k is increased by one for each subsequent iteration.

3.5.4 Independence of CGE from FPGA Routing Architectures

CGE achieves the ability to route arbitrary FPGA routing architectures by isolating the parts of the code that are architecture-specific. This is illustrated in Figure 3.6, which shows the overall flow of the algorithm. The code that is dependent on the routing architecture is enclosed in circles. As shown, the separate code includes the f_c (), f_s (), and *update* () routines. Any architecture that fits the general model described in Section 3.3 can be routed by changing these isolated routines. This generality is the key that allows the router's use, in Chapter 4, as a research tool for studying routing architecture flexibility. Figure 3.6 also shows the organization of the phases of CGE and the feedback path used over multiple iterations.

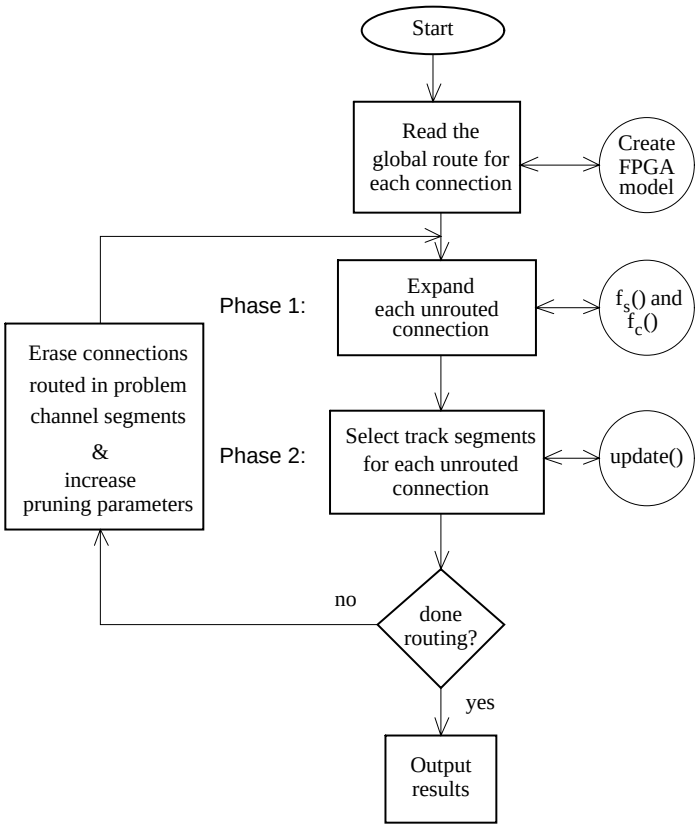

Figure 3.6 - *The Organization of CGE*

3.6 Results

CGE has been used to route several industrial circuits implemented as FPGAs. The routing results shown in this section are based on five circuits from four sources: Bell-Northern Research, Zymos, and two different designers at the University of Toronto. Table 3.1 gives the name, size (number of two-point connections and logic cells), source and the function of each circuit. For these results, the logic cell used is the result of a previous study [Rose89] [Rose90c], and the S and C blocks will be described in the next sub-section. Results are presented for a routing architecture similar to a commercial FPGA.

3.6.1 FPGA Routing Structures

Since the routability of an FPGA is determined by the topology and flexibility of its S and C blocks, those used in the tests of the algorithm are presented here. The general nature of the S block is illustrated in Figure 3.7a. Its flexibility is set by the parameter F_s, which defines the total number of connections offered to each wiring segment that enters the S block. For the example shown in Figure 3.7a, the wiring segment at the top left of the S block can connect to six other wiring segments, and so F_s is 6. Although not shown, the other wiring segments are similarly connected.

Circuit	#Blocks	#Conn	Source	Type
BUSC	109	392	UTD1	Bus Cntl
DMA	224	771	UTD2	DMA Cntl
BNRE	362	1257	BNR	Logic/Data
DFSM	401	1422	UTD1	State Mach.
Z03	586	2135	Zymos	8-bit Mult

Table 3.1 - *Experimental Circuits*

Figure 3.7b illustrates the test C block. The tracks pass uninterrupted through it and are connected to logic cell pins via a set of switches. The flexibility of the C block, F_c, is defined as the number of tracks that each logic cell pin can connect to. For the example shown in the figure, each logic cell pin can connect to 2 vertical tracks, and so F_c is 2.

3.6.2 Routing Results

The familiar yardstick of channel density is used as a measure of the quality of the detailed router. The 'Channel density' column in Table 3.2 shows the maximum channel density over all channels for each circuit. This represents a lower bound on the number of tracks per routing channel that is needed for each example. The real track requirements will depend on the flexibility of the routing architecture because there are interactions between one channel segment and another that are not accounted for in channel density measurements. The maximum flexibility has $F_s = 3W$ and $F_c = W$, where there are W tracks per channel. For the results in Table 3.2 the FPGA parameters are based on the Xilinx 3000 series [Xili89] FPGAs ($F_s = 6$, $F_c = 0.6W$). Table 3.2 gives the minimum number of tracks per channel that CGE needs in order to route 100 percent of the connections. The values for W are slightly greater than the global router minimum, which are excellent results considering the low flexibility of the FPGA routing architecture. Note

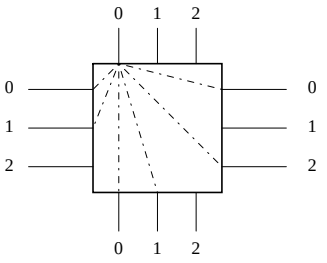

Figure 3.7a. The S block.

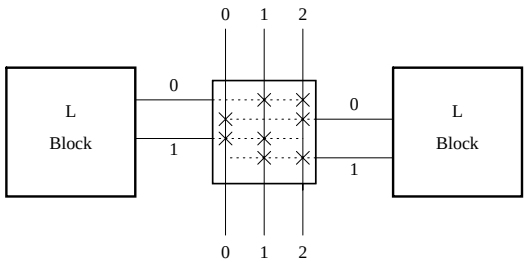

Figure 3.7b. The C block.

Figure 3.7 - *Definitions of S and C Block Flexibility*

that, although not shown, if F_c is increased to 0.8W, CGE achieves the absolute minimum number of tracks for all the circuits.

For comparison purposes, the same problems have also been routed using CGE with its c_f cost facility disabled. In this mode CGE has no ability to resolve routing conflicts and is thus a sequential router, similar to a maze router. At first glance, this may seem to be an unrealistic comparison because some maze routers are guided by cost functions that aid in finding good routes for connections. However, the 'maze' router used here has, in effect, access to the cost function that was used to solve the global routing, which is based on balancing the densities of all routing channels. Notwithstanding, this is a constrained 'maze' router because it is confined to remain within the global route of each connection, and the comparisons are valid only in that context. The rightmost column in Table 3.2 gives the number of tracks that the 'maze' router requires to achieve 100 percent routing. These results demonstrate that the 'maze' router needs an average of 60 percent more tracks than CGE. This shows that resolving routing conflicts is important and that CGE addresses this issue well. Figure 3.8 presents the detailed routing for circuit BUSC, with the FPGA parameters in Table 3.2; the logic cells are shown as solid boxes, whereas the S and C blocks are dashed boxes.

Circuit	Channel density	W required by CGE	W for 'maze'
BUSC	9	10	15
DMA	10	10	15
BNRE	11	12	20
DFSM	10	10	18
Z03	11	13	18

Table 3.2 - *CGE Minimum W for 100 % routing ($F_c = 0.6W$, $F_s = 6$)*

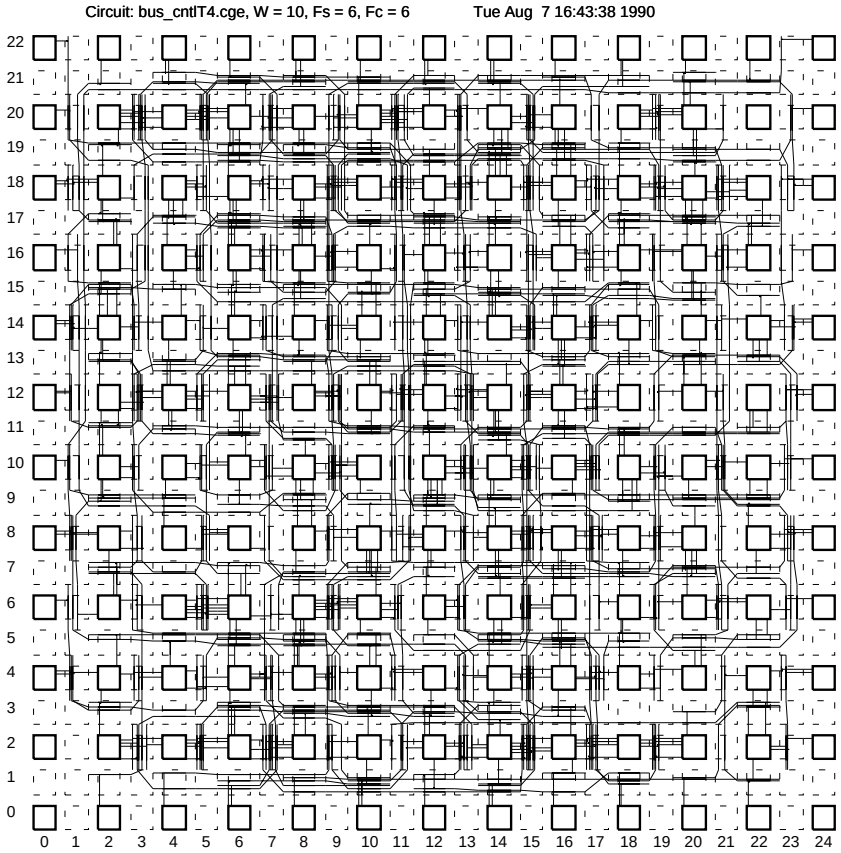

Figure 3.8 - *The Detailed Routing of Circuit BUSC*

3.6.3 Routing Delay Optimization for Critical Nets

Table 3.3 illustrates CGE's ability to optimize critical connections. For this experiment, several connections in circuit BNRE were marked critical. Then, CGE was used to route the circuit twice; once with CGE's critical net processing turned off, and once with it turned on. To facilitate this experiment, the FPGA was defined to have 18 tracks per channel, with four tracks hardwired for the entire length of each channel. Connections that use the hardwired tracks have lower routing delays because they pass through fewer switches (transistors). As Table 3.3 shows, a significant reduction in the number of switches in the critical paths was achieved.

Note that a better approach to routing delay optimization would set specific timing requirements that should be met for each critical *path* in a circuit. This exercise is left as future work.

Name of net	# of switches without critical processing	# of switches with critical processing
#143	15	5
#144	14	4
#220	10	3
#280	15	2
#351	15	4

Table 3.3 - *Critical Connection Routing Delay Optimization*

3.6.4 Memory Requirements and Speed of CGE

For the examples used here CGE needs between 1.5 and 7.5 MBytes of memory. As shown in Table 3.4, experimental measurements show that CGE is a linear-time algorithm, requiring from 25 to 215 SUN 3/60 CPU seconds for the smallest to the largest of the example circuits. This run-time behavior is due to the pruning procedure, which limits the number of routing alternatives that the algorithm considers for each connection.

Circuit	#Conn	Sun 3/60 CPU sec.	msec per connection
BUSC	392	25	63
DMA	771	59	76
BNRE	1257	122	97
DFSM	1422	103	72
Z03	2135	215	99

Table 3.4 - *CGE Run-time*

3.7 Conclusions and Future Work

This chapter has described a new kind of detailed routing algorithm that is designed specifically for Field-Programmable Gate Arrays. The algorithm is able to consider the side-effects that routing decisions made for one connection may have on another, and thus resolve routing conflicts and achieve a high quality result. The algorithm can be used over a wide range of FPGA routing architectures. It can route relatively large FPGAs in very close to the absolute minimum number of tracks as determined by global routing, and is capable of optimizing the routing delays of time-critical connections.

Future research should improve upon the treatment of time-critical nets. The router could be given specific timing requirements for each critical path in a circuit and should ensure that the routing delays of the nets in these paths do not exceed these times.

2.3.4.3 Advanced Micro Devices (AMD) FPGAs

The AMD FPGA [AMD90], based on EEPROM technology, can be considered to be an array of PAL (programmable AND/fixed OR) devices that are interconnected by a switch matrix, much like an Altera FPGA. Each PAL block consists of an AND plane that feeds a large block, called a Logic Allocator (LA). The LA allocates a variable number of product terms (p-terms) from the AND plane to individual Macrocells, where each Macrocell provides an OR function of its p-terms, optionally registered. The Macrocell outputs are fed back to the other PAL blocks via the switch matrix.

2.3.4.4 Quicklogic FPGAs

The Quicklogic FPGA [Quick91] consists of a two-dimensional array of like cells called pASIC Logic Cells (pLCs). Each pLC comprises four two-input AND gates feeding two two-input multiplexers, which feed a third multiplexer. The two first-stage multiplexer's select lines are driven by a single six-input AND gate, and the second-stage multiplexer's select line is driven by another six-input AND gate. The second-stage multiplexer provides the cell output, which can be optionally registered by a D flip-flop. The pLCs are interconnected by horizontal and vertical routing channels that provide full connectivity - every horizontal routing track can be connected to every vertical track and every track that passes a logic cell can connect to all the pins on that cell. Programmed connections are formed in Quicklogic FPGAs using a unique anti-fuse. Compared to the Actel anti-fuse, these devices boast an extremely low on-resistance (50 ohms) and unprogrammed capacitance (one femtofarad). Compared with other FPGAs, the Quicklogic devices are most like those from Actel.

4 The Flexibility of Field-Programmable Gate Array Routing Architectures

4.1 Introduction

This chapter uses an experimental approach to investigate the effect of the flexibility of an FPGA's routing architecture on its routability. *Flexibility* is a measure of the connectivity provided by a routing architecture and is a function of the total number of routing switches and wires. A loose definition of *routability* is the percentage of the connections in a given circuit that can be successfully routed. The experiments consist of routing a set of circuits in an FPGA that is modelled in such a way that the number of routing switches and wires can be changed. Routability is measured over a range of flexibility, using the routing algorithm described in Chapter 3.

Recall that a discussion in Chapter 1 showed that the choice of a good routing architecture involves a tradeoff among flexibility, logic density, and speed performance. High flexibility results in an FPGA that has good routability, but routing switches will be wasted if the flexibility is higher than is necessary for 100 percent routability. Also, high flexibility results in lower logic density and speed performance because each routing switch consumes significant area and has appreciable resistance and capacitance. Low flexibility, on the other hand, permits higher logic density and speed performance, but if flexibility is too low then 100 percent routability of circuits may not be possible.

The routing architectures that will studied in this chapter fit the FPGA model that was introduced in Chapter 3. Figure 4.1 reproduces the model, for ease of reference. The FPGA shown in the figure has three tracks per routing channel and two pins on each side of a logic cell (L). Recall that the connection (C) blocks contain the routing

switches that are used to connect the pins of the logic cells to the routing channels, and the switch (S) blocks at the intersections of horizontal and vertical routing channels provide the routing switches that can connect wiring segments in one channel segment to those in another. The flexibility of the routing architecture can be altered by changing the connectivity in the C blocks, the S blocks or the number of tracks in each routing channel [Rose90b] [Rose91].

The specific questions concerning FPGA routing architectures that are answered in this chapter include:

What is the effect of the flexibility of the C blocks on routability? Section 4.5.1 shows that a high flexibility in the C blocks is desirable for good routability.

What is the effect of the flexibility of the S blocks on routability? Section 4.5.2 shows that good routability can be achieved with a relatively low flexibility in the S blocks.

How do the S block and C block flexibilities interact? Section 4.5.3 shows that a high flexibility in either of the S or C blocks can compensate to some extent for a relatively low flexibility in the other.

What is the effect of the flexibilities of the C and S blocks on the number of tracks per routing channel required to achieve 100 percent routability? Section 4.5.4 shows that very close to the theoretical minimum number of tracks per channel (this is defined by the maximum channel density over all channels in the FPGA) can be achieved for surprisingly low flexibilities.

What is the effect of the flexibilities of the C and S blocks on the total number of routing switches required in an FPGA to achieve 100 percent routability? Section 4.5.5 shows that there is a range of C and S block flexibilities that results in a minimum number of switches.

This chapter is organized as follows: Section 4.2 presents assumptions that are made about the architecture of the FPGA, Section 4.3 describes the experimental procedure, the limitations of this work are discussed in Section 4.4, Section 4.5 presents the experimental results and explanations, and concluding remarks appear in Section 4.6.

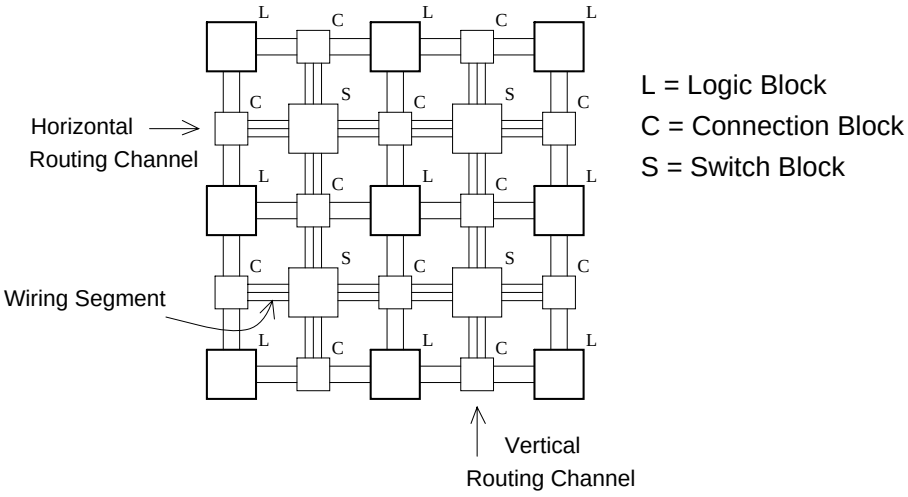

Figure 4.1 - *The Model of the FPGA Routing Architecture*

4.2 FPGA Architectural Assumptions

The architecture of an FPGA consists of its I/O cells, logic cells, and routing architecture. The I/O cells are assumed, as in Chapter 3, to be logic cells on the periphery of the chip. The other assumptions that are made in this chapter concerning the logic cells and the routing architecture are given in the following subsections.

4.2.1 The Logic Cell

The logic cell that is used here is the result of a previous study [Rose89][Rose90c] and is illustrated in Figure 4.2. It has a four input look-up table, a D flip-flop, and a tri-state output. In the previous study, this cell achieved the minimum total area over a set of circuits when compared to other cells that had differing numbers of inputs, including and excluding a D flip-flop. The logic cell has a total of 7 logical pins (4 inputs, 1 output, 1 clock, 1 tri-state) but they may not physically appear on all sides of the cell. The number of physical occurrences of each pin is an important architectural parameter, as explained below.

The number of logic cell sides on which each logical pin physically appears is called *T*. To illustrate this concept, the cases $T = 4$ and $T = 1$ are shown in Figure 4.3, where the logic cell pins are numbered from 0 to 6. As an example, the figure also shows the connection of pin 0 on one logic cell to pin 6 on another. The particular choice of *T* affects the routing problem in a number of ways. Selecting a low value of *T* implies that there will be fewer routing switches, which means the switches will use less area and add less capacitance to the tracks, but as shown in Figure 4.3, connections may be longer since it may be necessary to route to a certain side. This increases the channel densities and causes the connections to pass through more routing switches. Conversely, choosing a higher value of *T* allows shorter connections and minimizes the channel densities, but if *T* is higher than necessary, switches will be wasted.

For the experiments shown here, the value used for *T* is 2. This was chosen for area considerations only and was determined by performing the global routing of several circuits for each value of *T* and measuring the number of tracks per channel (maximum channel density) required in each case. The results are shown in Table 4.1, which gives the average maximum channel density of the circuits, for each value of *T*. The table shows a significant decrease in track count from the $T = 1$ to $T = 2$ case but diminishing

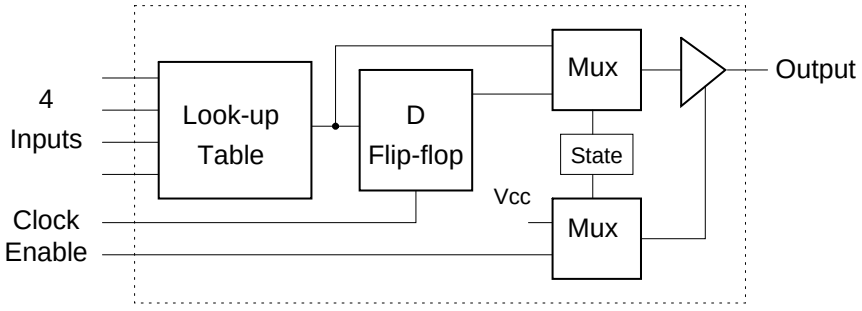

Figure 4.2 - *The Logic Cell*

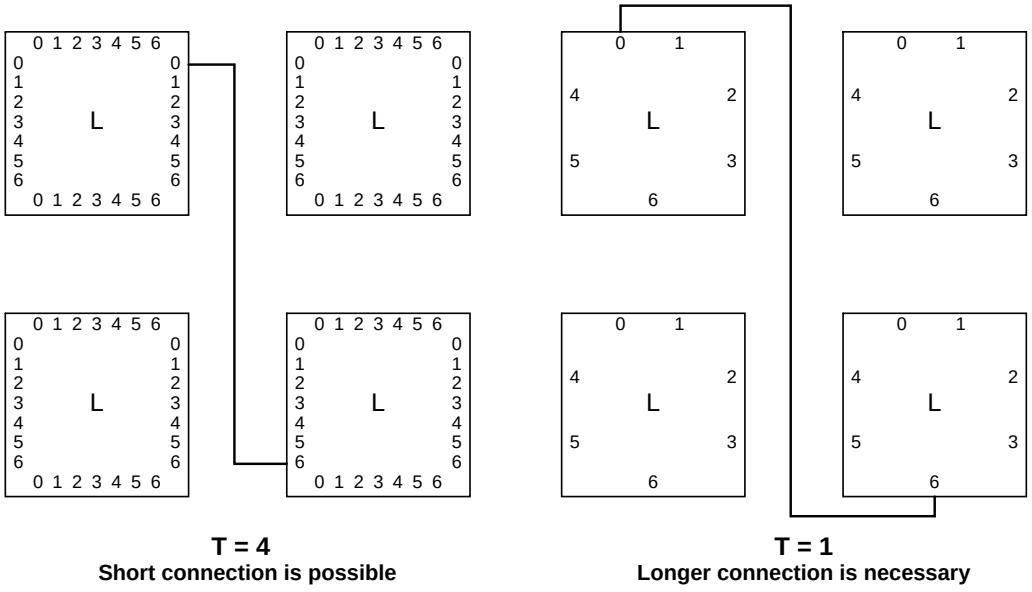

Figure 4.3 - *Example, and Effect on Connection Length, of T*

returns for higher values of *T*. Note that the routing tools used for these experiments did not make use of the functional equivalence of the logic cell inputs (the inputs to a look-up table are functionally equivalent), and if they had it may have been possible to choose a value of $T = 1$ without an increase in the number of tracks.

T	Avg. Maximum Channel Density
1	15
2	12
3	11
4	11

Table 4.1 - *The Effect of T on Channel Density*

The following two sections provide a detailed discussion of the C and S blocks used in these experiments. Some of this information also appeared in Chapter 3, but is repeated here for continuity.

4.2.2 The Connection Block

The connection block used is illustrated in Figure 4.4, where a routing switch is indicated by an X. The channel wires (drawn vertically in the figure) pass uninterrupted through the C block and have the option of connecting to the logic cell pins through the switches. The flexibility of the C block is represented by the variable F_c, which defines the number of tracks that each logic cell pin can connect to. For the example shown in the figure, each logic cell pin can connect to 2 vertical tracks, and so F_c is 2. In this chapter, no assumption is necessary about the implementation of routing switches, except that the switches are assumed to be bi-directional.

4.2.2.1 Connection Block Topology

The *topology* of the connection block (the pattern of the switches) can have a significant effect on routability, particularly when F_c is low. To illustrate this consider Figure 4.5 which shows two different C block topologies and one connection (from pin A to pin B) that must be routed. In this example each logic cell has three pins on a side and there are four tracks per routing channel. By examining the locations of the routing switches, it is clear that it is not possible to route the connection with Topology 1, while

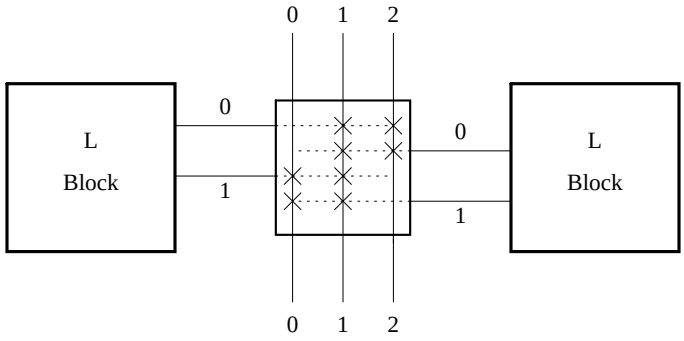

Figure 4.4 - *The Connection Block*

it is possible to do so using Topology 2. Topology 2 works because it has a common wire that can be reached by both pin A and pin B. This example illustrates the fact that a C block topology must provide common wires for every pair of pins that may need to be connected. At the same time, however, it is easy to recognize that it is desirable for the routing switches in the C block to be spread evenly among the tracks, so that there is a reasonable opportunity for each track to be used. A good C block topology should achieve a balance of these tradeoffs. Given W tracks per channel, the design of a C block is straight-forward if F_c is close to W, but for lower values of F_c the C block should be carefully designed. The issue is most acute if $F_c \leq 0.5W$ because at this point some pairs of pins may not have any common wires if the C block is poorly designed.

The topology of the C block that is used for the results presented in this chapter is illustrated by Figure 4.6. In the figure, there are 10 tracks per routing channel, seven pins per logic cell, and F_c is 6. The design of this topology is based on statistics, from a set of

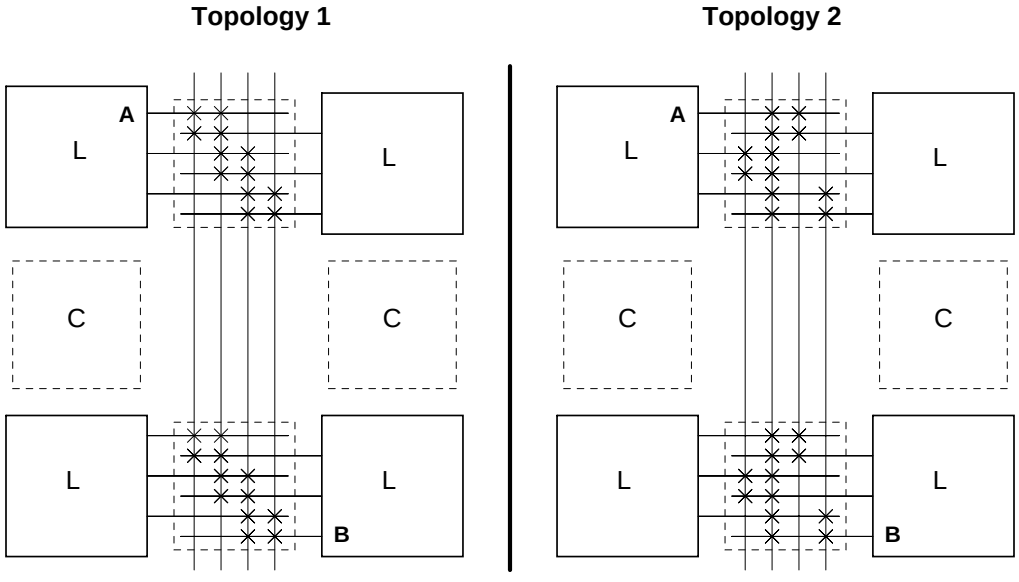

Figure 4.5 - *Two Connection Block Topologies*

circuits, that show how frequently each pair of pins is connected. For pins that are often connected the topology tends to provide common tracks, whereas for pins that are seldom connected different tracks are used. For example, the statistics say that pin 0 (a logic cell input) is often connected to pin 6 (an output), so these two pins share six tracks, whereas pin 0 is seldom connected to pin 5 (an input), so this pair shares only three tracks. This type of analysis is possible because logic cell inputs tend to be connected to outputs, and vice-versa. In this way, the topology provides as much overlap as practical for each pair of logic cell pins, while also balancing the distribution of the switches among the channel wires.

4.2.3 The Switch Block

The general nature of the switch block used is illustrated in Figure 4.7. Its flexibility, F_s, defines 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 left of the S block can be switched to six other wiring segments, and so F_s is 6. Although not shown, the other wiring segments are similarly connected.

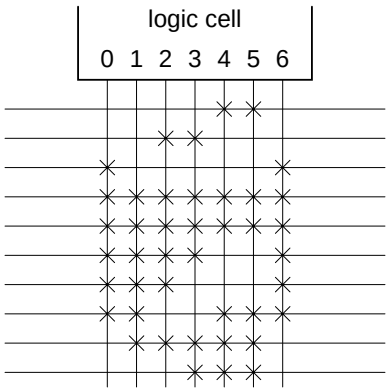

Figure 4.6 - *The C Block Topology Used for this Research*

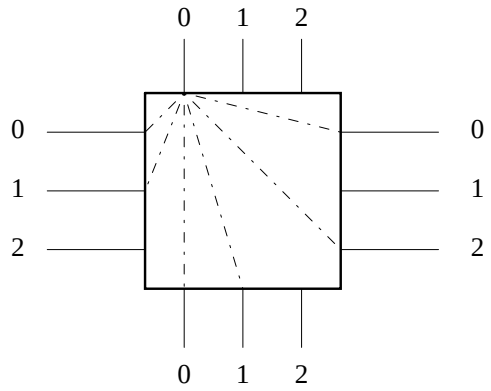

Figure 4.7 - *The Switch Block*

4.2.3.1 Switch Block Topology

The topology of the S blocks can be very important since it is possible to choose two different topologies with the same flexibility measure (F_s) that result in very different routabilities. This is particularly important if the flexibility is low. As an illustration, consider the two different topologies shown in Figure 4.8. In both topologies, the switch block has the same flexibility measure, $F_s = 2$. Assume that a global router has specified that a wiring segment at A must be connected to another at B by traveling through the two switch blocks shown. By examining the routing switches, it is easy to see that it is not possible to reach B from A with topology 1, while it is with topology 2. The reason that topology 2 is successful can be explained as follows. Consider the two vertical wires in topology 2 that connect from A to the two horizontal wires on the right side of the S block. At the next S block, one of the horizontal wires can connect to the top of the block (to B) and one to the bottom. The key is that any turn that is taken at one S block does not prohibit any other turn at the next S block, and this is true for all possible sequences of turns. For the results that are presented in this chapter, topology 2 is used. For higher flexibilities, switches are added such that the basic pattern is preserved.

4.3 Experimental Procedure

This section describes the experimental procedure that is used to investigate FPGA routing architectures. Given a functional description of a circuit, the following steps implement the circuit in an FPGA.

(1) Perform the *technology mapping* [Keut87] of the original network into the FPGA logic cells. This step transforms the functional description of the network into a circuit that interconnects only logic cells of the type shown in Figure 4.2. The technology mapping was done using an early version of the algorithm described in [Fran90].

(2) Perform the placement of the netlist of logic cells. The logic cells are placed by the Altor placement program [Rose85], which is based on the min-cut placement algorithm [Breu77]. Altor makes the resulting two-dimensional array of logic cells as square as possible.

(3) Perform the global routing of the logic cell interconnections. This step finds a path through the routing channels for each pair of logic cell pins that are to be connected.

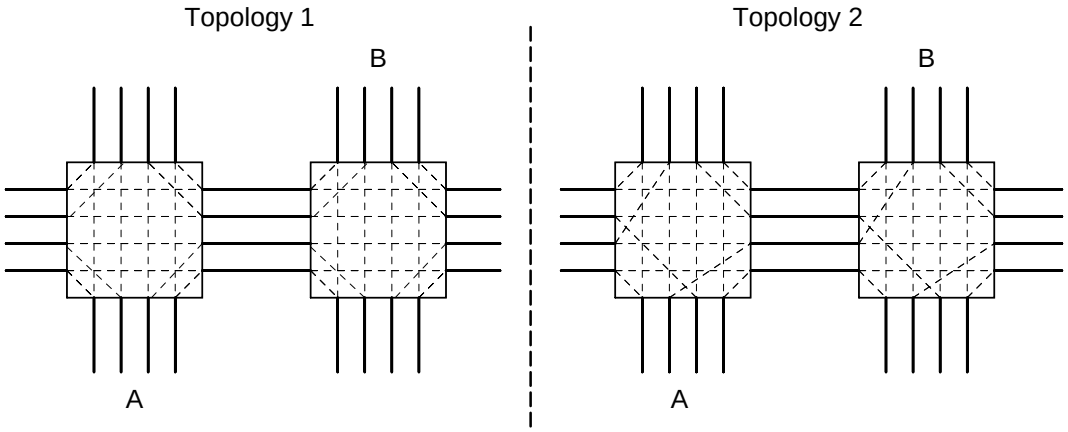

Figure 4.8 - *Two Switch Block Topologies*

Since each connection is assigned to specific channels this determines the *maximum channel density* of the circuit, which is defined as the maximum number of connections that pass through any channel segment. This sets the theoretical minimum number of tracks per channel (for the particular global router used) that is needed to route the circuit. The global router that is used here is based on the LocusRoute standard cell global routing algorithm [Rose90a], that is described in Chapter 2.

(4) Perform the detailed routing of each connection, using the path assigned by the global router. The CGE detailed router, described in Chapter 3, is used for this purpose, and yields two kinds of results. If a specific W (number of tracks per channel) is given as input, CGE determines the percentage of connections that can be successfully routed for specific values of F_s and F_c. Alternatively, if the desired output is the number of tracks per routing channel required to route 100% of connections for a specific F_s and F_c, then CGE is invoked repeatedly, with an increasing number of tracks, until complete routing is achieved.

The salient point in this procedure is that the global router is used only once for each circuit, and this determines the densities of all of the routing channels. The number of tracks required per channel to route each circuit then depends on the flexibility of the routing architecture. Thus, to investigate the effect of flexibility on routability, step (4) is performed over a range of values of F_c, F_s, and W.

4.4 Limitations of this Work

This section discusses the effects of the architectural assumptions and the experimental procedure on the accuracy of the results that are presented later in this chapter.

The models that have been used for the C and S blocks are based on balanced topologies, in that each L block pin can be connected to exactly F_c tracks and each wiring

segment that enters an S block can connect to exactly F_s others. Also, every wiring track must use a routing switch to pass through an S block - i.e. all the tracks comprise short wiring segments only. Although it is also interesting to consider other classes of architectures, the assumptions made here allow interesting and useful results to be generated with experiments that have simple parameters.

The experimental procedure described in Section 4.3 limits each connection to a single global route. A better approach would be one that provides a feedback mechanism that allows the detailed router to request a different global route for connections that fail. Finally, the accuracy of the routability results that are presented in this chapter depends on the quality of the routing CAD tools, which includes both the global and detailed routers.

4.5 Experimental Results

The experimental results that are presented here are based on the five circuits that were described in Table 3.1. This section first investigates the effect of the flexibilities of the C and S blocks on the routability of these circuits and shows the tradeoffs that exist between these two blocks. Following this, the effect of different values of F_c and F_s on the number of tracks required per channel is shown. Finally, the effect of the C and S block flexibilities on the total number of switches required in an FPGA is measured.

4.5.1 Effect of Connection Block Flexibility on Routability

Figure 4.9 is a plot of the percentage of successfully routed connections versus connection block flexibility, F_c, for the circuit BNRE. Each curve in the figure corresponds to a different value of switch block flexibility, F_s. The lowest curve in the figure corresponds to the case $F_s = 2$ and the highest curve to $F_s = 10$. The number of tracks, W, is set to 14, which is two greater than W_g, the minimum possible number of tracks as

indicated by the global router. The value of $W = W_g + 2$ was chosen to give the detailed router a reasonable chance of success. Using a higher or lower value of W would shift the curves slightly upward or downward, respectively. Figure 4.9 indicates that the routing completion rate is very low for small values of F_c and only achieves 100% when F_c is at least one-half of W. The figure also shows that increasing the switch block flexibility improves the completion rate at a given F_c, but to get near 100% the value of F_c must always be high (above 7 for this circuit).

Table 4.2 summarizes the results for the other circuits. It gives the minimum values of F_c and $\frac{F_c}{W}$ required to achieve 100% routing completion for each circuit, for nine

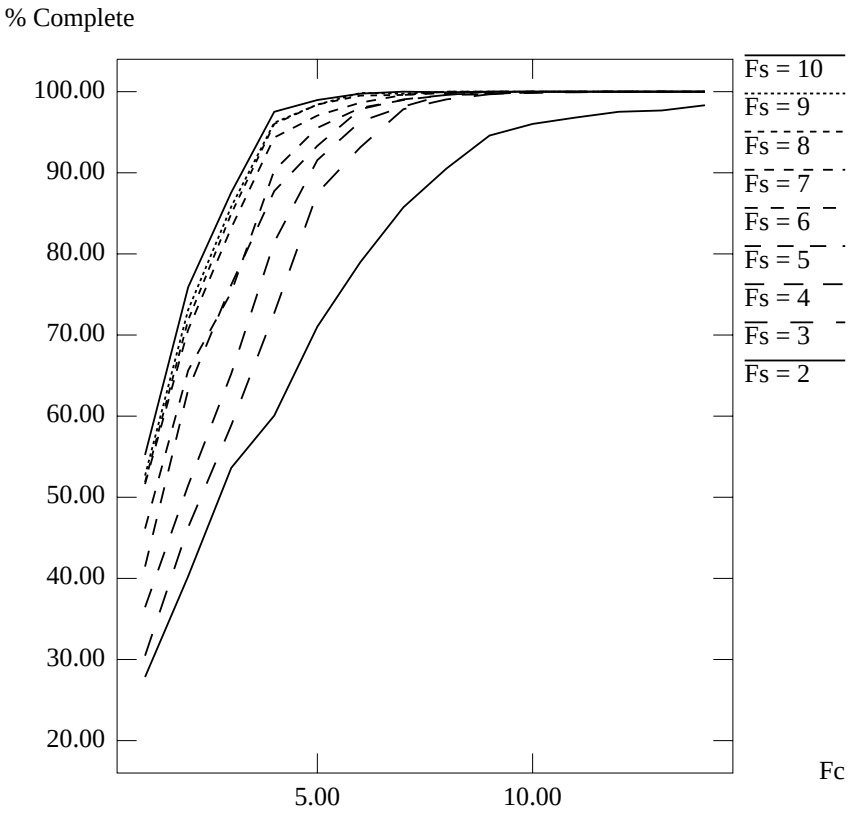

Figure 4.9 - *Percent Routing Completion vs. Fc, Circuit BNRE*

values of F_s. W is fixed at $W_g + 2$, in all cases, to give a reasonable chance for success. The value "nr" in the table indicates that 100% routing was not achieved.

The key observation from the data of Table 4.2 is that there appear to be minimum values of F_c and $\frac{F_c}{W}$ below which circuits are not routable. However, since this data is based on a fixed value of $W = W_g + 2$, it is interesting to investigate whether F_c or $\frac{F_c}{W}$ can be reduced if W is not fixed. To study this, a similar experiment was conducted in which W was allowed to vary to a maximum of $3 \times W_g$. Again, the experiments measure the minimum possible values of F_c and $\frac{F_c}{W}$ for which 100 percent routing can be achieved, for a range of values of F_s. The results are shown in Figure 4.10, which for conciseness gives the average results for the five circuits. The left curve in the figure shows that $\frac{F_c}{W}$ can be substantially reduced by allowing W to vary, but the curve to the right shows that F_c still reaches about the same minimum value.

To see why there exists a minimum value of F_c below which circuits are not routable, consider the following discussion concerning C block topology. Assume that a C block must connect n logic cell pins to a set of tracks, and that some pin, p_i, must be able to connect to all of pins p_j, $1 \le j \le n$. Some connections between these pin pairs will occur within one C block and others will involve two different C blocks. To simplify the analysis, assume that $F_s \le 3$, so that no jogging is allowed among the tracks. As the discussion in Section 4.2.2.1 showed, the C block topology must provide at least one common track that connects to both p_i and each p_j. To accomplish this, the design of the topology may:

Circuit	W	Fs	100% Fc	Fc/W
BUSC	11	2	nr	nr
BUSC	11	3	9	0.82
BUSC	11	4	7	0.64
BUSC	11	5	7	0.64
BUSC	11	6	6	0.54
BUSC	11	7	6	0.54
BUSC	11	8	5	0.45
BUSC	11	9	6	0.54
BUSC	11	10	5	0.45
DMA	12	2	nr	nr
DMA	12	3	8	0.67
DMA	12	4	7	0.58
DMA	12	5	7	0.58
DMA	12	6	7	0.58
DMA	12	7	7	0.58
DMA	12	8	5	0.42
DMA	12	9	5	0.42
DMA	12	10	5	0.42
BNRE	14	2	nr	nr
BNRE	14	3	12	0.86
BNRE	14	4	11	0.79
BNRE	14	5	10	0.71
BNRE	14	6	9	0.64
BNRE	14	7	10	0.71
BNRE	14	8	8	0.57
BNRE	14	9	8	0.57
BNRE	14	10	7	0.50
DFSM	13	2	nr	nr
DFSM	13	3	9	0.69
DFSM	13	4	9	0.69
DFSM	13	5	9	0.69
DFSM	13	6	8	0.62
DFSM	13	7	8	0.62
DFSM	13	8	7	0.54
DFSM	13	9	7	0.54
DFSM	13	10	7	0.54
Z03	13	2	nr	nr
Z03	13	3	10	0.77
Z03	13	4	9	0.69
Z03	13	5	9	0.69
Z03	13	6	9	0.69
Z03	13	7	7	0.54
Z03	13	8	7	0.54
Z03	13	9	7	0.54
Z03	13	10	7	0.54

Table 4.2 - *Minimum F_c Required for 100% Completion*

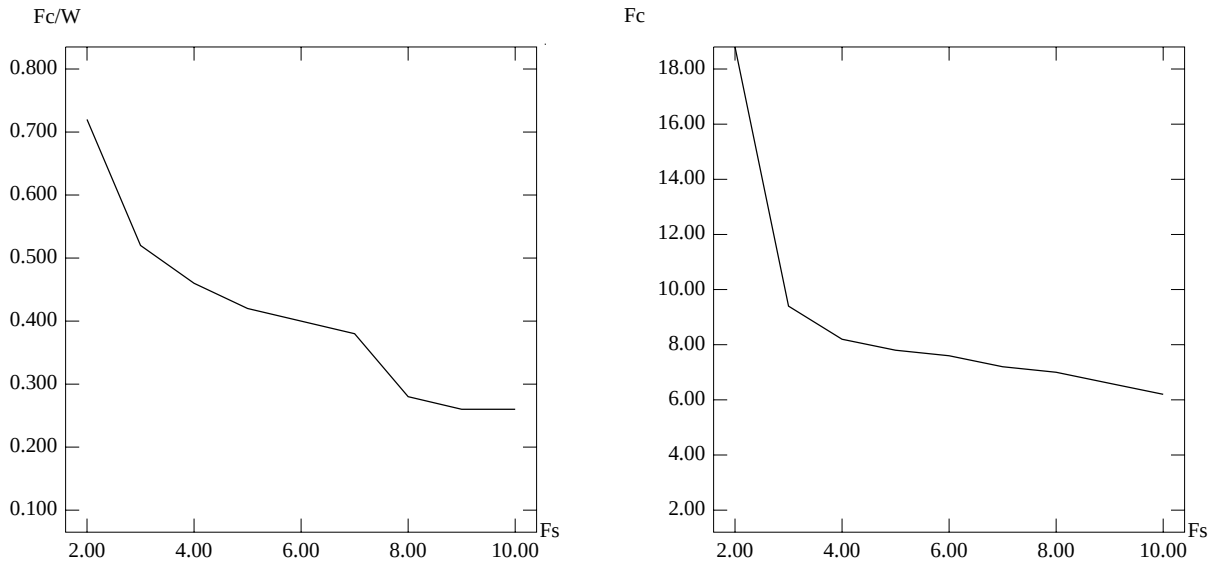

Figure 4.10 - $\frac{F_c}{W}$ *vs.* F_s *and* F_c *vs.* F_s*, With a Variable W*

(1) Attach two switches to each of *n* different tracks, such that each track connects one p_j to p_i. In terms of Section 4.2.2.1, this corresponds to spreading the switches evenly across the tracks.

(2) Attach *n* switches to any one track, such that p_i can connect to any p_j on that track.

(3) Use a combination of options (1) and (2).

Option (1) leads directly to a minimum value for F_c because it entails attaching *n* switches to p_i. The effect of option (2) is more subtle, as discussed below. Consider Figure 4.11, which shows a C block topology in which each pin connects to exactly the same F_c tracks as every other pin. The figure shows that, with this topology, when one pin is connected to a track, one choice of track is eliminated for every other pin. In this scenario, it follows that the minimum possible value of F_c is determined by the maximum number of pins that are connected at any C block.

A more realistic C block, such as the one that was shown in Figure 4.6, is based on option (3). This means that a combination of the effects of options (1) and (2) determines a minimum value for F_c. The key to this discussion is that any realistic C block must provide connections between a number of different pairs of pins and this leads directly to a minimum possible value for F_c.

Note that the minimum value of F_c can be reduced slightly by increasing F_s to be above three, because this increases the connectivity between pairs of pins by allowing jogging from one track to another. However, this only affects connections that involve two different C blocks, and since some connection's pins are both within one C block, an absolute minimum value exists for F_c.

4.5.2 Effect of Switch Block Flexibility on Routability

Figure 4.12 is a plot of the percentage routing completion versus switch block flexibility, F_s. Each curve in the figure corresponds to a different value of F_c, with the lowest

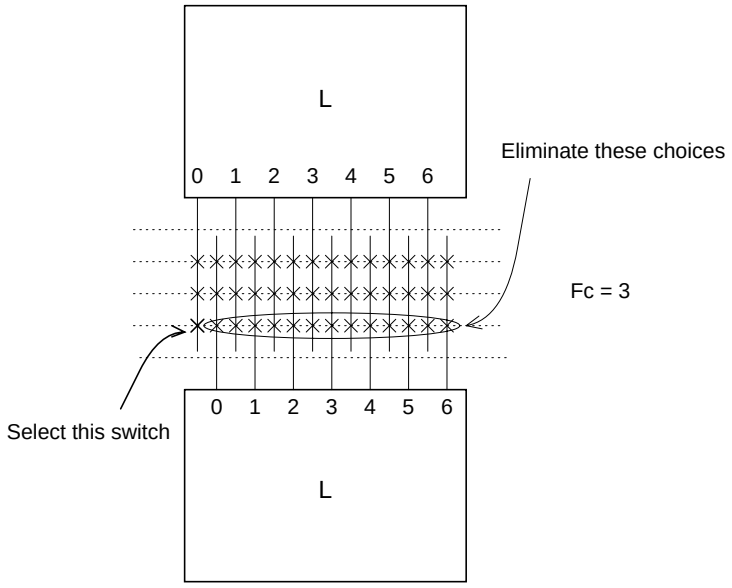

Figure 4.11 - *Connecting One Pin Eliminates One Choice for Every Other*

curve representing $F_c = 1$ and the highest curve corresponding to $F_c = W$. This plot is for the circuit BNRE, with W set to 14. The figure shows that if F_c is high enough, then very low values of F_s can achieve 100% routability. These F_s values are low in comparison with the maximum possible value of F_s, which is $3 \times W$. For the results in Figure 4.12 this maximum is 42, whereas 100% routing completion is often achieved for $F_s = 3$. This makes intuitive sense because even for $F_s = 3$ every track that passes through a particular C block is guaranteed to connect to one other track at every other C block. To further quantify the effect of F_s on routability, consider the connection of two logic cell pins that are separated by n S blocks. The number of tracks connectable at the first logic cell pin is F_c and the number of paths available to reach the connection block adjacent to the second logic cell is

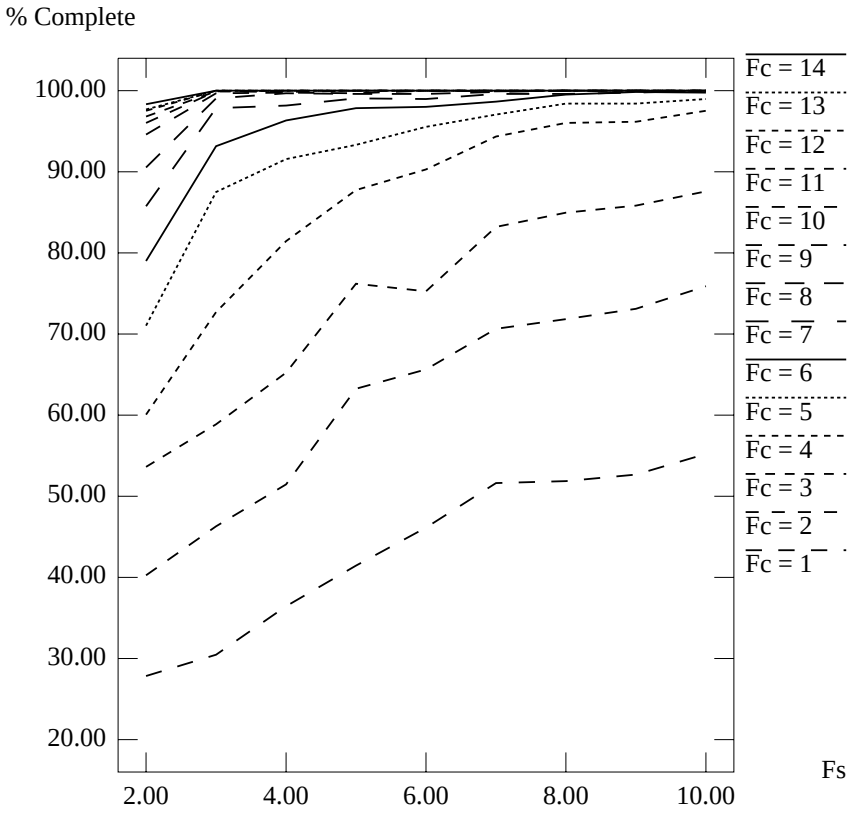

Figure 4.12 - *Percent Routing Completion vs. Fs, Circuit BNRE*

$$\#\,Paths\;=\;F_c \times \left(\frac{F_s}{3}\right)^n$$

Using the average value of n of about 3 for typical circuits, if $F_s = 3$ and $F_c = 10$, then there are 10 paths available. If F_s is increased to 6, there are 80 paths available. Thus a small increase in F_s greatly increases the number of paths, and hence the routability.

4.5.3 Tradeoffs in the Flexibilities of the S and C Blocks

Figures 4.9 and 4.12 can be combined in three-dimensions to show that a tradeoff exists between the flexibilities of the S blocks and the C blocks. This is illustrated by the three-dimensional surface plot in Figure 4.13. The plot shows, for example, that a decrease along the F_c axis can be compensated for by an increase along the F_s axis, and vise-versa. This can also be seen in Figure 4.14, which is a plot of the minimum value of F_c (averaged over the five circuits) for which 100 percent routing can be attained for a range of values of F_s. In this plot, W is constant for each circuit, at two higher than W_g. Note that there is no data point for the case $F_s = 2$ in the figure because the circuits are not routable for that S block flexibility with $W = W_g + 2$. The slope of the curve in Figure 4.14 will flatten for higher values of F_s since there exists a minimum value of F_c for each circuit, as was discussed in Section 4.5.1.

Note that Figure 4.14 alone does not provide enough information to choose the best values of F_c and F_s because it is based on the fixed value of W. The required value of W for different values of F_c and F_s is the subject of the following section.

4.5.4 Track Count Requirements

This section investigates the effect of the S and C block flexibilities on routability by measuring the number of tracks per channel, W, required to route 100 percent of a circuit's connections for specific values of F_s and F_c. For these experiments, the detailed

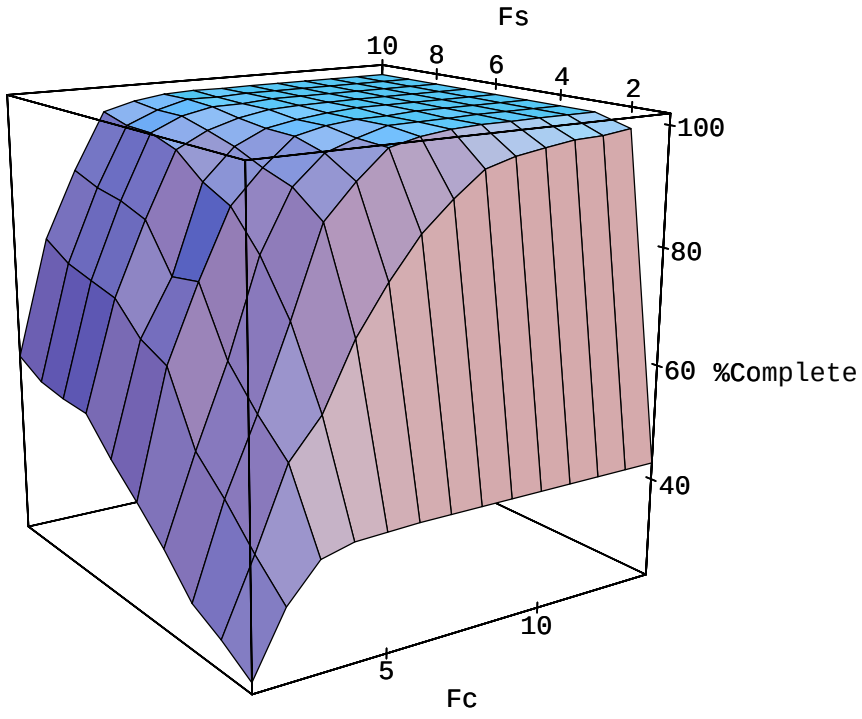

Figure 4.13 - *C Block and S Block Tradeoff*

router was invoked repeatedly over a range of values of W, from W_g (the maximum channel density of the circuit) to a maximum of $3 \times W_g$, until 100 percent routing was achieved. Each routing architecture is assessed by comparing how close the required W is to W_g. For these experiments the flexibility of the C block is expressed as the ratio of F_c over W - as W is changed, the ratio is kept constant. This provides a convenient way to coordinate changes in W with the number of routing switches in a C block. This is automatic for an S block, since F_s applies to each track.

Table 4.3 shows the number of tracks required to achieve 100 percent routing completion for circuit BNRE over a set of FPGA routing architectures. Each entry in the table corresponds to a different pair of (F_s , $\frac{F_c}{W}$) values. The value "nr" in the table

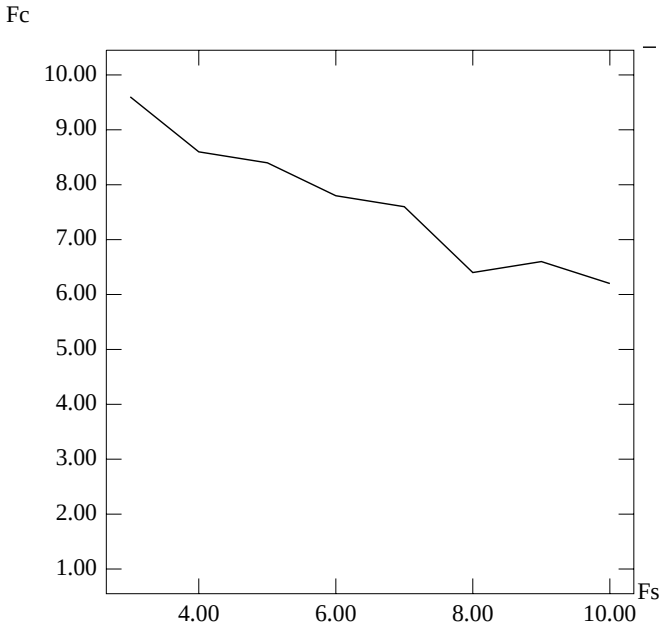

Figure 4.14 - *F_c for 100% Routing vs. F_s, With $W = W_g + 2$*

means that 100 percent routing was not achieved. Since W_g for this circuit is 12, the table shows that even with low flexibilities it is possible to achieve 100 percent routing using very near to the minimum possible number of tracks.

Table 4.4 summarizes the results for all the circuits. Each entry in this table is the average over all the circuits of the number of tracks that are required *in excess* of the minimum (W_g) to route 100 percent of the connections. These results show that with very low flexibilities it is possible to achieve a number of tracks only slightly greater than the minimum. In particular, for $F_s \geq 3$ and $\frac{F_c}{W} \geq 0.6$, excluding the case ($F_s = 3$, $\frac{F_c}{W} = 0.6$), the number of tracks required in excess of the minimum is no greater than three.

Fs	Fc/W							
	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
2	nr	nr	nr	nr	nr	23	23	17
3	nr	nr	nr	18	14	13	13	13
4	nr	nr	18	14	13	13	13	13
5	nr	nr	18	14	13	12	12	12
6	nr	21	16	14	14	12	13	13
7	nr	19	17	14	12	12	12	12
8	nr	19	15	13	12	12	12	12
9	23	17	13	13	12	12	12	12
10	19	16	13	13	12	12	12	12

Table 4.3 - *Track Count Requirements for BNRE (Minimum = 12)*

4.5.5 Architectural Choices

The choice of a particular FPGA routing architecture must be based on the cost of its implementation in terms of area and delay. Although these depend on the technology used for the routing switches, it is possible to make general comments because, regardless of their implementation, routing switches consume area and cause time delays. It is instructive to examine the total number of routing switches required by different routing architectures. The number of switches in a C block and an S block depends on the number of logic cell pins (P), the number of sides that each pin appears on (T), and on the parameters F_c, F_s, and W, according to the following equations:

$$\#\ Switches\ in\ Connection\ Block = \frac{1}{2} \times T \times P \times F_c$$

$$\#\ Switches\ in\ Switch\ Block = 2 \times F_s \times W$$

For the results presented here P is 7, T is 2, and the last three parameters are read from the equivalent of Table 4.3 for each circuit. As the flexibility of the routing structures

Fs	Fc/W 0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
2	nr	nr	nr	nr	nr	11.2	10.8	9.0
3	nr	nr	nr	4.6	2.4	1.2	0.8	0.8
4	nr	nr	6.2	3.0	1.6	0.6	0.6	0.6
5	nr	nr	4.2	2.4	1.0	0.4	0.2	0.2
6	nr	7.6	3.8	1.8	0.8	0.4	0.4	0.4
7	nr	5.2	3.4	1.4	0.2	0.2	0.2	0.2
8	nr	4.4	1.4	0.6	0.2	0.0	0.4	0.2
9	10.0	4.2	1.0	0.4	0.2	0.2	0.0	0.0
10	8.0	3.2	1.4	0.6	0.2	0.0	0.0	0.0

Table 4.4 - *Average Excess Track Count Requirements over all Circuits*

increases (F_s and F_c) the number of switches per track increases, but the number of tracks per channel may decrease, as shown in Table 4.3. Hence there should be an architecture that exhibits a minimum total number of switches. Table 4.5 gives the number of switches *per tile* required for circuit BNRE for each FPGA routing architecture. A tile is the section of the FPGA that would be replicated across the entire chip, and includes the logic cell, two connection blocks and one switch block.

As Table 4.5 shows, flexibilities of $F_s = 3$ and $\frac{F_c}{W} = 0.7$ achieve a minimum number of switches for this circuit, at 221. Note that several neighboring architectures have similar switch counts. For all of the test circuits the minimum number of switches was between 172 and 223, and occurred when the architecture's parameters were in the range $3 \le F_s \le 4$ and $0.7 \le \frac{F_c}{W} \le 0.8$.

	Fc/W							
Fs	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
2	nr	nr	nr	nr	nr	349	381	306
3	nr	nr	nr	259	221*	223	241	260
4	nr	nr	270	229	231	249	267	286
5	nr	nr	306	257	257	254	271	288
6	nr	369	304	285	305	278	319	338
7	nr	372	357	313	285	302	319	336
8	nr	410	345	317	309	326	343	360
9	510	401	325	343	333	350	367	384
10	459	409	351	369	357	374	391	408

Table 4.5 - *Average Number of Switches per Tile for Each Architecture*

4.6 Conclusions

This chapter has explored the relationships between the flexibility of routing architectures and routability in FPGAs. The principal conclusions are that connection blocks should have high flexibility to achieve high percentage routing completion, but that a relatively low flexibility is sufficient in the switch blocks. Furthermore, with low flexibilities the number of tracks per channel required is very close to the minimum. Finally, it has been shown that routing architectures with these properties yield the lowest total number of routing switches.

5 A Stochastic Model to Predict the Routability of Field-Programmable Gate Array Routing Architectures

5.1 Introduction

In this chapter, a study of FPGA routing architectures using a stochastic modelling approach is presented. The purpose of the model is twofold: to confirm the experimental results that were presented in Chapter 4, and to develop the foundations of a theory that facilitates further study of FPGA routing architectures, without the time-consuming demands of experimental work. In the stochastic model, circuits are represented by parameters that specify the total number of connections to be routed and the length of each connection. The model probabilistically 'routes' a circuit in an FPGA that has a certain size and a given routing architecture. Using probability theory, the model predicts the effect of a routing architecture's flexibility on its routability. Flexibility and routability were defined in Section 4.1. It is shown that the theoretical results are comparable to the corresponding experimental results from Chapter 4.

FPGAs are characterized in the stochastic model using the same basic assumptions that appear in Chapters 3 and 4. For ease of reference, some information discussed in those chapters is repeated here. An example FPGA is illustrated in Figure 5.1, which shows a square array of logic cells, with N cells per side. Note that the grid coordinates in Figure 5.1 number only the logic cells, which differs from figures shown in earlier chapters. This change has been made because this labelling scheme is more convenient for the stochastic model. No assumptions are necessary about the internal details of the logic cells, except that each cell has some number of pins that are connected to the routing channels by routing switches. Routing switches are contained within C blocks and S blocks. It is assumed that tracks are hard-wired straight through the C blocks, without

passing through routing switches, but that a routing switch is always involved to pass through an S block. This means that all tracks are composed of wiring segments that span the length of one logic cell. The switches in the C blocks are used to connect the logic cell pins to the wiring segments, and the S block switches provide connections from one wiring segment to another. The flexibility of a C block is defined by F_c, which specifies the number of tracks that each logic cell pin can connect to. The flexibility of an S block is given by F_s, which defines the number of other wiring segments that a wiring segment entering an S block can be connected to.

The routing of FPGAs is modelled assuming that a two-stage routing approach of global routing followed by detailed routing is used (see Section 2.2.2). The global routing stage is represented in the model by making assumptions, which are described in the next section, concerning the solution that a global router would produce. Assuming that a connection is assigned a single global route, the stochastic model uses probability

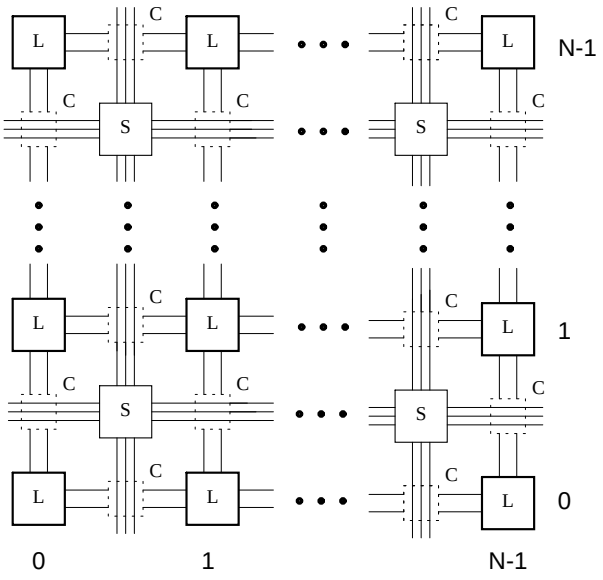

Figure 5.1 - *An $N \times N$ FPGA*

theory to represent the detailed routing of an FPGA as a random process. The probability of successfully performing the detailed routing of each connection is calculated and used to predict the routability of the FPGA.

This chapter is organized as follows. Section 5.2 provides an overview of the stochastic model, Section 5.3 describes previous research that is used to predict channel densities, Section 5.4 derives analytic expressions for calculating the probability that a connection can be successfully routed, the theoretical predictions of routability are given in Section 5.5, and Section 5.6 provides concluding remarks.

5.2 Overview of the Stochastic Model

In the stochastic model, it is assumed that a circuit with a total of C_T two-point connections is to be routed in an FPGA with $N \times N$ logic cells. The length of each connection is drawn from a probability distribution, P_L. It will later be necessary to choose a specific distribution for P_L. In Section 5.4.4, it is assumed that P_L is geometric, with mean length $\bar{R}$. This assumption is taken from earlier work on the stochastic modelling of two-dimensional arrays of connected cells [Heller84] [ElGa81], and has the following physical interpretation in an FPGA: at each C block along the path of a connection, the connection will terminate (at a logic cell) with probability $\frac{1}{\bar{R}}$ and will continue (to the next C block) with probability $1 - \frac{1}{\bar{R}}$.

The C_T connections are individually referred to as $C_1, C_2, ..., C_{C_T}$ and the statistical event that each connection is successfully routed is called $R_{C_1}, R_{C_2}, ..., R_{C_{C_T}}$. The key to the stochastic model is the calculation of the probabilities of $R_{C_1}, R_{C_2}, ..., R_{C_{C_T}}$. Recall that routability is defined as the percentage of the connections in a circuit that can be suc-

cessfully routed. In terms of $R_{C_1}, R_{C_2}, ..., R_{C_{C_T}}$ this corresponds to the ratio of the expected number of successfully routed connections to the total number of connections, C_T. Thus, routability is the average probability of completing a connection and can be calculated in the stochastic model according to

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

where $P(R_{C_i})$ is the probability of successfully routing C_i.

5.2.1 Model of Global Routing and Detailed Routing

In order to use a key research result by El Gamal [ElGa81] to predict the densities of the routing channels in an FPGA, the following assumption is made concerning the way in which a global routing algorithm would assign the connections in a circuit to the routing channels. It is assumed that each connection is assigned a single path through the routing channels in such a way that the number of connections per routing channel is Poisson distributed. Section 5.3 justifies this assumption and illustrates its use.

In the stochastic model, the detailed routing of an FPGA is represented as a random process. Based on the assumption that a connection is assigned a single path through the routing channels, the probability of successfully performing the detailed routing of the connection is calculated using combinatorial analysis. The probability expressions account for the number of tracks per routing channel, the flexibilities of the C and S blocks, and the side-effects that the routing of one connection has on others.

Recall, from earlier discussions in this dissertation, that a key issue in the detailed routing of FPGAs is how the routing of one connection may affect other connections. To compute the value of each $P(R_{C_i})$, it is necessary for the stochastic model to account for these effects. To accomplish this, the model 'routes' the connections in a serial manner

and predicts the effect that each successfully routed connection has on the densities of the routing channels. By this mechanism, the probability of routing each subsequent connection is influenced because there are more connections in a channel to compete with. The next section shows how El Gamal's results can be used to calculate channel densities and following this, the probability formulas for $P(R_{C_i})$ are derived.

5.3 Previous Research for Predicting Channel Densities

In [ElGa81], a stochastic model is developed to predict the wiring requirements of Master Slice Integrated Circuits that have a two-dimensional array of identical cells, with horizontal and vertical routing channels between the rows and columns of cells. The model divides the channels into segments that span the length or width of one cell and it is assumed that all interconnections start at one cell and travel a minimum distance through the channel segments to another cell. It is further assumed that the number of connections per cell can be drawn independently from a Poisson distribution, with parameter λ, where λ is defined as the ratio of the total number of connections in a circuit to the total number of cells in the array. The average connection length, in number of cells traversed, is called $\bar{R}$. The paper also makes assumptions about the trajectories of connections, but these assumptions are not necessary for the results that are quoted here.

[ElGa81] shows that under the above assumptions, in an array that has $N \times N$ routing channels, the densities of the channel segments will be Poisson distributed, with the average density given by $\frac{\lambda \bar{R}}{2}$. This result holds as long as $\bar{R} < \infty$, independent of N, and provides a convenient method of predicting channel densities.

5.3.1 Predicting Channel Densities in FPGAs

Although the results in [ElGa81] were developed for Master Slice circuits, they can also be applied to the FPGAs considered here, since both types of devices are based on a two-dimensional array of identical cells. The definitions of the routing channels differ, but these differences can be ignored since the tracks consist of short segments that span only one logic cell in both cases.

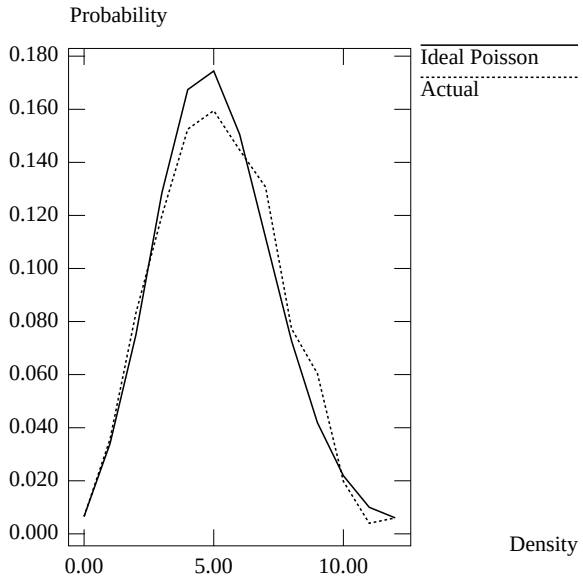

Figure 5.2 - *Predicted versus Actual Channel Densities*

Having made these assumptions, it is convenient to predict channel densities in FPGAs using El Gamal's result. The accuracy of the predictions can be checked by comparing the ideal Poisson distribution with mean $\frac{\lambda \overline{R}}{2}$ to the distribution of channel densities in real FPGA circuits. Such comparisons were conducted for the example circuits from Chapter 4. A typical result is shown in Figure 5.2, which gives one curve for the ideal Poisson distribution and another curve for the measured distribution of channel densities. As the figure shows, the actual channel densities are surprisingly close to the Poisson predictions.

It is interesting to discuss a physical interpretation of the Poisson distribution in this context. Assume that an FPGA has W tracks in each routing channel and consider a specific channel segment. For each of the W tracks, define p_i as the probability of the statistical event that the track would be occupied if a circuit were routed in the FPGA. If $W = 1$, there will be a probability, p_1, that the track will be occupied by some connection. If $W = 2$, then there will be a probability, p_2, that each of the two tracks will be occupied by some connection and $p_2 < p_1$. Extending this to the general case, if $W = n$, then each track will be occupied with probability p_n, and $p_n < p_{n-1} < \cdots < p_2 < p_1$. Furthermore, as $n \rightarrow \infty$, $p_n \rightarrow 0$. Since p_n is small in the limiting case, the event that a track is used is a rare event and the number of these events (density) can be approximated by the Poisson distribution.

In FPGAs in which the tracks consist of segments that span multiple logic cells, El Gamal's result is probably not an accurate approximation of channel densities. In such cases, a different method of calculating densities would be needed. For this reason, the probability expressions that are developed in the following section are derived in a general way that does not hinge upon any particular distribution for the channel densities. However, assuming a Poisson distribution for the channels does allow some expressions to be simplified, an example of which is given in Section 5.4.1.

5.4 Calculating the Probability of Successfully Routing a Connection

This section derives analytic expressions for calculating the probability of successfully performing the detailed routing of a single connection in an FPGA. As an example of a connection, consider Figure 5.3. The figure shows a connection, C_i, that starts at logic cell (x_1, y_1) and travels through routing channels to logic cell (x_2, y_2). The length of C_i is defined in terms of logic cell hops (to be consistent with [ElGa81]), as

$LC_i = |x_1 - x_2| + |y_1 - y_2|$. Also, the number of S blocks that C_i passes through is given by $LC_i - 1$. To define the probability, $P(R_{C_i})$, of successfully routing C_i, assume that $LC_i = n+1$. The statistical event that corresponds to this assumption is written L_{n+1}. Also define the following events:

- X_1 - the event that the logic cell pin associated with C_i at (x_1, y_1) can connect to at least one track at the first C block. Note that there are, by definition, F_c tracks that can connect to the logic cell pin, but any number of them may already be used by other connections that have been previously 'routed'.
- $S_1, S_2, \cdots, S_n$ - the events that C_i can successfully reach at least one track on the outgoing side of the first, second, up to the n^{th} S block. There are $LC_i - 1$ such events for C_i.
- X_2 - the event that at least one of the tracks that are available to C_i at the last C block can be connected to the appropriate logic cell pin at (x_2, y_2).
- R_{C_i} - the event that C_i can be successfully routed.

Since C_i is successfully routed only if all of the events $X_1, S_1, S_2, ..., S_n, X_2$ occur, then

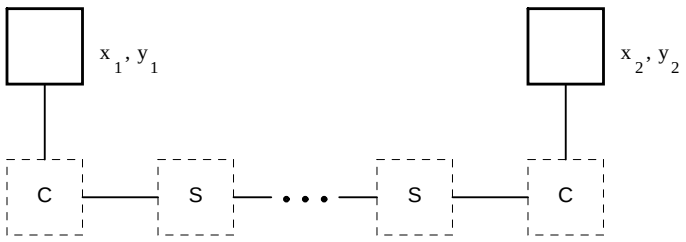

Figure 5.3 - *A Typical Connection*

$$R_{C_i} \mid L_{n+1} = X_1 \cap S_1 \cap S_2 \cdots \cap S_n \cap X_2$$

and the probability of successfully routing C_i is given by

$$\begin{aligned} P(R_{C_i} \mid L_{n+1}) &= P(X_1 \cap S_1 \cap S_2 \cdots \cap S_n \cap X_2) \\ &= P(X_1)P(S_1 \mid X_1)P(S_2 \mid S_1 \cap X_1) \cdots P(S_n \mid S_{n-1} \cap \cdots \cap S_1 \cap X_1) \\ &\cdot P(X_2 \mid S_n \cap \cdots S_1 \cap X_1). \end{aligned} \quad 5.1$$

Since the events $X_1, S_1, S_2, ..., S_n,$ and X_2 are not independent, it is necessary to find formulas for each of the terms in Equation 5.1. This is accomplished in the following sections by using combinatorial analysis that accounts for the flexibilities of the C and S blocks (F_c and F_s), the number of tracks per routing channel (W), and the densities of the routing channels. As discussed in Section 5.3, channel density is approximated by the Poisson distribution with parameter $\lambda_g = \frac{\lambda \bar{R}}{2}$, where λ is the number of connections per logic cell and $\bar{R}$ is the average connection length. Appropriate values for λ and $\bar{R}$ are discussed in Section 5.5.

5.4.1 The Logic Cell to C Block Event

The event X_1 can be depicted pictorially as shown in Figure 5.4. The figure shows a routing channel with W tracks and a logic cell pin that can connect to F_c of the tracks, via routing switches (shown by an X). The figure also shows a set of D tracks, drawn as dashed lines, that are already occupied by previously routed connections. In the figure, $W = 10$, $F_c = 5$, and $D = 5$. The event X_1 can then be viewed as a random process in which a logic cell pin can connect to any of the unused tracks where there are switches. To derive a formula for $P(X_1)$, it is convenient to define the event NONE as the oppo-

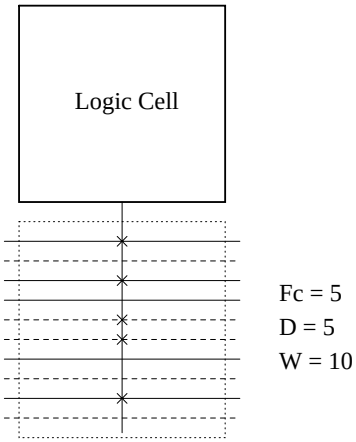

Figure 5.4 - *The Event X_1*

site of X_1 - i.e. $P(X_1) = 1 - P(NONE)$. The event *NONE* occurs when all F_c tracks are within the set of D used tracks. As a first step to evaluating $P(NONE)$, assume that $D = d$ and define the corresponding event D_d. Assuming that the F_c switches can appear on any of the W tracks, the probability of NONE conditional on D_d is given by the ratio of the number of ways in which all of the F_c switches can lie within the d occupied tracks to the number of ways in which the F_c switches can appear on any of the W tracks. Using combinatorial analysis, this can be expressed as

$$P(NONE \mid D_d) = \frac{{}_dC_{F_c}}{{}_WC_{F_c}}, \qquad 5.2$$

where ${}_dC_{F_c}$ means the combinations of d things taken F_c at a time. As a check, note that $P(NONE \mid D_d)$ is 0 if $d < F_c$ and 1 if $d = W$. Next, consider the events $D_0, D_1, ..., D_W$ corresponding to the possible values of D. Since the occurrence of NONE implies exactly one of $D_0, D_1, ..., D_W$, then

$$NONE = (NONE \cap D_0) \cup (NONE \cap D_1) \cup \cdots \cup (NONE \cap D_W)$$

and since $D_0, D_1, ..., D_W$ are mutually exclusive

$$P(NONE) = P(NONE \cap D_0) + P(NONE \cap D_1) + \cdots + P(NONE \cap D_W).$$

Using the relation $P(X \cap Y) = P(X)P(Y \mid X)$,

$$P(NONE) = P(D_0)P(NONE \mid D_0) + P(D_1)P(NONE \mid D_1) + \cdots + P(D_W)P(NONE \mid D_W).$$

The terms $P(D_d)$ are given by the Poisson distribution with parameter λ_g, written $p(\lambda_g, d)$, so that

$$P(NONE) = \sum_{d=0}^{W} p(\lambda_g, d) \cdot P(NONE \mid D_d)$$

and, substituting equation 5.2,

$$P(NONE) = \sum_{d=0}^{W} p(\lambda_g, d) \cdot \frac{{}_dC_{F_c}}{{}_WC_{F_c}}.$$

Finally,

$$P(X_1) = 1 - P(NONE) = 1 - \sum_{d=0}^{W} p(\lambda_g, d) \cdot \frac{{}_dC_{F_c}}{{}_WC_{F_c}} \qquad 5.3$$

Note that Equation 5.3 is an approximation because the Poisson distribution has an infinite tail, whereas the summation has an upper limit of *W*. This means that there is a non-zero probability of channel densities above *W*, but for practical values of *W* this error is very small and can be ignored. This same statement also applies to other equations that appear later in this chapter.

Equation 5.3 has been developed in a way that does not depend upon the channel densities being Poisson distributed. This approach is taken because the channel densities in some FPGAs, such as those having tracks with segments that span multiple logic cells, may have distributions that are not Poisson. The stochastic model can still be used for such FPGAs, simply by replacing $p(\lambda_g, d)$ with an appropriate distribution. It is

interesting to note, however, that the properties of the Poisson distribution allow expressions like Equation 5.3 to be simplified, as described below.

Equation 5.3 can be simplified by noting that a Poisson distribution is infinitely divisible. In the case of event X_1, this means that rather than considering a Poisson process over W tracks, with mean λ_g, it is sufficient to deal with a smaller Poisson process over F_c tracks, with mean $\lambda_g \frac{F_c}{W}$. $P(NONE)$ is then given by $p(\lambda_g \frac{F_c}{W}, F_c)$, and Equation 5.3 can be expressed as

$$P(X_1) = 1 - P(NONE) = 1 - p(\lambda_g \frac{F_c}{W}, F_c).$$

Similar simplifications can be made for other expressions shown later in this section, but they are easily developed and so are not shown.

Equation 5.3 calculates $P(X_1)$ based on the relationship between the event X_1 and the event *NONE*. An alternative is to calculate $P(X_1)$ directly by defining $A_1^{X_1}, A_2^{X_1}, ..., A_{F_c}^{X_1}$ as the events that X_1 occurs with exactly 1, 2, ..., F_c available tracks. Using this approach,

$$X_1 = A_1^{X_1} \cup A_2^{X_1} \cup \cdots \cup A_{F_c}^{X_1}$$

and since $A_1^{X_1}, A_2^{X_1}, ..., A_{F_c}^{X_1}$ are mutually exclusive,

$$P(X_1) = P(A_1^{X_1}) + P(A_2^{X_1}) + \cdots + P(A_{F_c}^{X_1}).$$

Although $P(X_1)$ can be calculated using Equation 5.3, each of $P(A_a^{X_1})$ will be required in the next section, so they are derived here. Consider the general case of X_1 occurring with exactly (a) available tracks, and the corresponding event $A_a^{X_1}$. Assuming a specific number of occupied tracks, $D = d$, the conditional probability $P(A_a^{X_1} \mid D_d)$ can be expressed using combinatorial analysis as

$$P(A_a^{X_1} \mid D_d) = \frac{{}_dC_{(F_c-a)} \cdot {}_{(W-d)}C_a}{{}_WC_{(F_c-a)} \cdot {}_{(W-(F_c-a))}C_a} \cdot {}_{F_c}C_a \ .$$

In words, this is the number of ways that a set of $(F_c - a)$ tracks can be within (d) used tracks times the number of ways of choosing a set of (a) tracks from (F_c) tracks, all divided by the number of ways that two distinguishable sets of (a) and $(F_c - a)$ tracks can be within W tracks. Since the occurrence of $A_a^{X_1}$ implies exactly one of $D_0, D_1, ..., D_W$, following the steps shown for equation 5.3,

$$\begin{aligned} P(A_a^{X_1}) &= \sum_{d=0}^{W} p(\lambda_g, d) \cdot P(A_a^{X_1} \mid D_d) \\ &= \sum_{d=0}^{W} p(\lambda_g, d) \cdot \frac{{}_dC_{(F_c-a)} \cdot {}_{(W-d)}C_a}{{}_WC_{(F_c-a)} \cdot {}_{(W-(F_c-a))}C_a} \cdot {}_{F_c}C_a \end{aligned} \qquad 5.4$$

As a check, it is easily verified that $P(NONE)$ can be obtained using equation 5.4 by setting $a = 0$, which must be true since $P(NONE) \equiv P(A_0^{X_1})$. Finally,

$$\begin{aligned} P(X_1) &= \sum_{a=1}^{F_c} P(A_a^{X_1}) \\ &= \sum_{a=1}^{F_c} \sum_{d=0}^{W} p(\lambda_g, d) \cdot \frac{{}_dC_{(F_c-a)} \cdot {}_{(W-d)}C_a}{{}_WC_{(F_c-a)} \cdot {}_{(W-(F_c-a))}C_a} \cdot {}_{F_c}C_a \ . \end{aligned}$$

5.4.2 The S Block Events

All of the events that are associated with S blocks can be treated in a uniform way. This section first derives probability formulas for $S_1 \mid X_1$ and then shows how the resulting expressions can be applied to subsequent S blocks.

5.4.2.1 The First S Block Event, for $F_s = 3$

Since $P(S_1 \mid X_1)$ will be affected by the flexibility of the S block, it is convenient to assume a specific value of F_s. In the following derivation, the case $F_s = 3$ is assumed.

This is the easiest case to handle because it means that each wiring segment that enters an S block can connect to exactly one wiring segment on each other side. Also, the derivation need not be concerned with whether a connection turns or passes straight through an S block since the effect is the same in both cases.

The event $S_1 \mid X_1$ is depicted in Figure 5.5, which shows an S block and a routing channel that has W tracks. The figure shows a set of A^{X_1} tracks, drawn as bold lines, that are available at the incoming side of the S block and a set of D tracks, drawn as dashed lines on the outgoing side of the S block, that are already used by other connections. In the figure, $D = 4$, $W = 10$, and $A^{X_1} = 3$. Note that setting A^{X_1} to three corresponds to the event $A_3^{X_1}$, from Section 5.4.1. Figure 5.5 uses dotted lines to indicate S block switches and shows that each track on the incoming side of the S block can be connected to one other track on the outgoing side. The event can then be considered to be a random process in which each of the A^{X_1} incoming tracks can connect to one track on the outgoing side of the S block, as long as that outgoing track is not among the D used tracks. In other words, given that there are A^{X_1} tracks that are available on the incoming side of the S block, it is necessary to find the probability that one or more of these tracks are also available on the outgoing side.

The event $S_1 \mid X_1$ can occur with one or more available outgoing tracks. To calculate $P(S_1 \mid X_1)$, define $A_1^{S_1}, ..., A_{F_c}^{S_1}$ as the events that $S_1 \mid X_1$ occurs with exactly 1, 2, ..., F_c available tracks on the outgoing side. Since

$$S_1 \mid X_1 = A_1^{S_1} \cup \cdots \cup A_{F_c}^{S_1}$$

and $A_1^{S_1}, ..., A_{F_c}^{S_1}$ are mutually exclusive

$$P(S_1 \mid X_1) = P(A_1^{S_1}) + \cdots + P(A_{F_c}^{S_1}) . \qquad 5.5$$

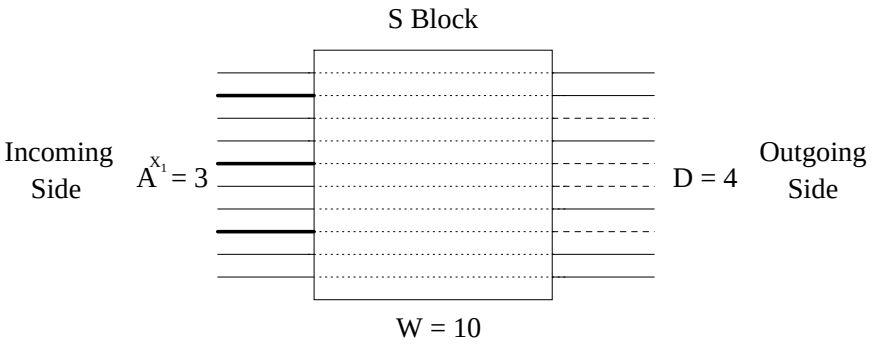

Figure 5.5 - *The Event S_1*

Solving for each term in this summation requires several steps. Consider the general case where $S_1 \mid X_1$ occurs with exactly k available outgoing tracks. The corresponding event is written $A_k^{S_1}$. The probability of $A_k^{S_1}$ will depend on the number of tracks available on the incoming side, given by A^{X_1}, and on the value of D. Assume a specific value of $A^{X_1} = a$. Since X_1 is known to have occurred, this corresponds to assuming that X_1 occurred with exactly (a) available tracks. The appropriate statistical event for this assumption is then written as $A_a^{X_1} \mid X_1$. Also, assume that $D = d$ and define the event D_d. A conditional probability for $A_k^{S_1}$ can then be expressed using combinatorial analysis as

$$P((A_k^{S_1} \mid (A_a^{X_1} \mid X_1)) \mid D_d) = \frac{{}_dC_{(a-k)} \cdot {}_{(W-d)}C_k}{{}_WC_{(a-k)} \cdot {}_{W-(a-k)}C_k} \cdot {}_aC_k . \qquad 5.6$$

Equation 5.6 expresses the ratio of the number of ways in which exactly (k) of the (a) available incoming tracks can end up on unoccupied tracks on the outgoing side to the number of ways in which (k) and $(a-k)$ tracks can appear on any of the W tracks. To expand Equation 5.6, following the steps outlined in the previous section, consider the events $D_0, D_1, ..., D_W$ corresponding to the possible values of D. Since the occurrence

of $(A_k^{S_1} \mid (A_a^{X_1} \mid X_1))$ implies exactly one of $D_0, D_1, ..., D_W$, then

$$P(A_k^{S_1} \mid (A_a^{X_1} \mid X_1)) = \sum_{d=0}^{W} p(\lambda_g, d) \cdot \frac{{}_dC_{(a-k)} \cdot {}_{(W-d)}C_k}{{}_WC_{(a-k)} \cdot {}_{W-(a-k)}C_k} \cdot {}_aC_k \,.$$

Next, consider the events $(A_1^{X_1} \mid X_1)$, ..., $(A_{F_c}^{X_1} \mid X_1)$ corresponding to the possible values of A^{X_1}. The occurrence of $A_k^{S_1}$ implies exactly one of $(A_1^{X_1} \mid X_1)$, ..., $(A_{F_c}^{X_1} \mid X_1)$, so that

$$P(A_k^{S_1}) = \sum_{a=1}^{F_c} P(A_a^{X_1} \mid X_1) \cdot \sum_{d=0}^{W} p(\lambda_g, d) \cdot \frac{{}_dC_{(a-k)} \cdot {}_{(W-d)}C_k}{{}_WC_{(a-k)} \cdot {}_{W-(a-k)}C_k} \cdot {}_aC_k \,. \qquad 5.7$$

As stated above, the terms $P(A_a^{X_1} \mid X_1)$ express the probability that, given the occurrence of event X_1, X_1 occurred with exactly (a) available tracks. Each of $P(A_a^{X_1} \mid X_1)$ is defined by Bayes' rule [Fell52], according to

$$P(A_a^{X_1} \mid X_1) = \frac{P(A_a^{X_1})}{\sum_{j=1}^{F_c} P(A_j^{X_1})}, \qquad 5.8$$

where $P(A_1^{X_1})$, ..., $P(A_{F_c}^{X_1})$ are given by Equation 5.4. Substituting 5.7 and 5.8 into 5.5,

$$P(S_1 \mid X_1) = \sum_{k=1}^{F_c} P(A_k^{S_1})$$

$$= \sum_{k=1}^{F_c} \sum_{a=1}^{F_c} \frac{P(A_a^{X_1})}{\left(\sum_{j=1}^{F_c} P(A_j^{X_1}) \right)} \cdot \sum_{d=0}^{W} p(\lambda_g, d) \cdot \frac{{}_dC_{(a-k)} \cdot {}_{(W-d)}C_k}{{}_WC_{(a-k)} \cdot {}_{W-(a-k)}C_k} \cdot {}_aC_k \,. \qquad 5.9$$

5.4.2.2 The First S Block Event, for Any Value of F_s

Equation 5.9 assumes a specific value of S block flexibility, $F_s = 3$. This section shows how Equation 5.9 can be generalized for other values of F_s. In Equation 5.6, a one-to-one correspondence was assumed between the subscript (a) in $A_a^{X_1} \mid X_1$, on the left hand side of the equation, and the variable (a), on the right hand side. This relation holds for $F_s = 3$ but does not necessarily apply for other values of F_s. For example, if

$F_s = 6$, a more appropriate variable for the right hand side of the equation is $2a$. In general, the subscript (a) should be scaled by some factor, α, and Equation 5.6 becomes

$$P((A_k^{S_1} \mid (A_a^{X_1} \mid X_1)) \mid D_d) = \frac{{}_dC_{(\alpha a-k)} \cdot {}_{(W-d)}C_k}{{}_WC_{(\alpha a-k)} \cdot {}_{W-(\alpha a-k)}C_k} \cdot {}_{\alpha a}C_k \cdot \qquad 5.10$$

Clearly, α depends on the value of F_s, but α may also depend on whether a connection passes straight through a particular S block, or turns. Define Z_1 as the event that a connection passes straight through an S block, and Z_2 as the event that it turns. Also, define α_1 and α_2 as the values of α corresponding to Z_1 and Z_2. Since $S_1 \mid X_1$ implies one of Z_1 and Z_2, then

$$P(S_1 \mid X_1) = P(Z_1) \cdot P((S_1 \mid X_1) \mid Z_1) + P(Z_2) \cdot P((S_1 \mid X_1) \mid Z_2)$$

and using Equation 5.9 and 5.10,

$$P(S_1 \mid X_1) =$$

$$P(Z_1) \cdot \sum_{k=1}^{W} \sum_{a=1}^{F_c} \frac{P(A_a^{X_1})}{\left(\sum_{j=1}^{F_c} P(A_j^{X_1}) \right)} \cdot \sum_{d=0}^{W} p(\lambda_g, d) \cdot \frac{{}_dC_{(\alpha_1 a-k)} \cdot {}_{(W-d)}C_k}{{}_WC_{(\alpha_1 a-k)} \cdot {}_{W-(\alpha_1 a-k)}C_k} \cdot {}_{\alpha_1 a}C_k +$$

$$P(Z_2) \cdot \sum_{k=1}^{W} \sum_{a=1}^{F_c} \frac{P(A_a^{X_1})}{\left(\sum_{j=1}^{F_c} P(A_j^{X_1}) \right)} \cdot \sum_{d=0}^{W} p(\lambda_g, d) \cdot \frac{{}_dC_{(\alpha_2 a-k)} \cdot {}_{(W-d)}C_k}{{}_WC_{(\alpha_2 a-k)} \cdot {}_{W-(\alpha_2 a-k)}C_k} \cdot {}_{\alpha_2 a}C_k \qquad 5.11$$

Appropriate values for $P(Z_1)$ (note that $P(Z_2) = 1 - P(Z_1)$), α_1, and α_2 are discussed in Section 5.5. Note that the (k) summation in Equation 5.11 has an upper limit of W, whereas the corresponding upper limit in Equation 5.9 is F_c. This change is required since it may be possible to connect to all W tracks in a channel for values of F_s that are greater than three.

5.4.2.3 The Remaining S Block Events

Thus far, this section has dealt specifically with the event $S_1 \mid X_1$, but the derived expressions are applicable to any of the other S block events, with two changes. First, for the m^{th} S block event, $(S_m \mid S_{m-1} \cap \cdots \cap S_1 \cap X_1)$, all summations must reach an upper limit of W. Second, the probabilities $P(A_1^{X_1})$, ..., $P(A_{F_c}^{X_1})$ in Equation 5.11 are replaced by $P(A_1^{S_{m-1}})$, ..., $P(A_W^{S_{m-1}})$, which are defined by Equation 5.12, with $m = m-1$. Applying these changes, Equation 5.7 becomes

$$P(A_k^{S_m}) = \sum_{a=1}^{W} P(A_a^{S_{m-1}} \mid S_{m-1} \cap \cdots \cap S_1 \cap X_1) \cdot \sum_{d=0}^{W} p(\lambda_g, d) \cdot \frac{{}_dC_{(a-k)} \cdot {}_{(W-d)}C_k}{{}_WC_{(a-k)} \cdot {}_{W-(a-k)}C_k} \cdot {}_aC_k \qquad 5.12$$

and Equation 5.11 becomes

$$P(S_m \mid S_{m-1} \cap \cdots \cap S_1 \cap X_1) =$$

$$P(Z_1) \cdot \sum_{k=1}^{W} \sum_{a=1}^{W} \frac{P(A_a^{S_{m-1}})}{\left(\sum_{j=1}^{W} P(A_j^{S_{m-1}}) \right)} \cdot \sum_{d=0}^{W} p(\lambda_g, d) \cdot \frac{{}_dC_{(\alpha_1 a-k)} \cdot {}_{(W-d)}C_k}{{}_WC_{(\alpha_1 a-k)} \cdot {}_{W-(\alpha_1 a-k)}C_k} \cdot {}_{\alpha_1 a}C_k +$$

$$P(Z_2) \cdot \sum_{k=1}^{W} \sum_{a=1}^{W} \frac{P(A_a^{S_{m-1}})}{\left(\sum_{j=1}^{W} P(A_j^{S_{m-1}}) \right)} \cdot \sum_{d=0}^{W} p(\lambda_g, d) \cdot \frac{{}_dC_{(\alpha_2 a-k)} \cdot {}_{(W-d)}C_k}{{}_WC_{(\alpha_2 a-k)} \cdot {}_{W-(\alpha_2 a-k)}C_k} \cdot {}_{\alpha_2 a}C_k \qquad 5.13$$

5.4.3 The C Block to Logic Cell Event

The event X_2 is depicted by Figure 5.6, which shows a set of $A^{S_n} = 4$ tracks, drawn as bold lines, that are available at a C block (this corresponds to the event $A_4^{S_n}$ in Section 5.4.2) and a set of $F_c = 5$ tracks that connect to the appropriate logic cell pin for the connection. The event X_2 can then be viewed as a random process in which the logic cell pin can be connected to any of the set of (A^{S_n}) tracks where there are switches. Stated differently, given that one or more tracks were available at the outgoing side of the last S block, it is necessary to determine the probability that one or more of these tracks con-

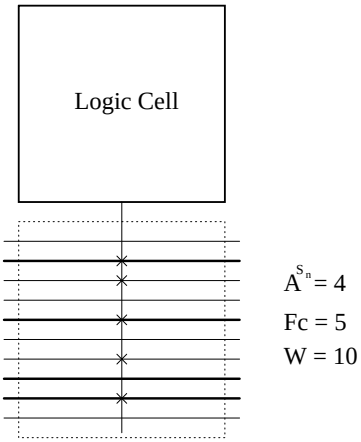

Figure 5.6 - *The Event X_2*

nects to the appropriate logic cell pin. To simplify the notation, the expression $S_n \cap \cdots S_1 \cap X_1$ will be substituted for by *SX*. To calculate the probability of $X_2 \mid SX$, define the opposite event $NONE \mid SX$, where $P(X_2 \mid SX) = 1 - P(NONE \mid SX)$. To find $P(NONE \mid SX)$, assume a specific value of $A^{S_n} = a$ and define the corresponding event $A_a^{S_n}$. A conditional probability for $NONE \mid SX$ can then be defined by

$$P((NONE \mid SX) \mid A_a^{S_n}) = \frac{(W - F_c)C_a}{WC_a} . \qquad 5.14$$

Equation 5.14 assumes that each of the F_c switches for the the logic cell pin associated with event X_2 is equally likely to be on any of the *W* tracks. This may not be realistic since a good C block topology would ensure that the tracks that are connectable to one pin would overlap the tracks connectable to others, as was discussed in Section 4.2.2.1. This assumption will have the effect of producing low predictions of routability for low values of F_c, which is discussed further in Section 5.5.

Consider the events $A_1^{S_n}, A_2^{S_n}, ..., A_W^{S_n}$ corresponding to the possible values of A^{S_n}. Since the occurrence of $NONE \mid SX$ implies exactly one of $A_1^{S_n}, A_2^{S_n}, ..., A_W^{S_n}$, it follows

that

$$P(NONE \mid SX) = \sum_{a=1}^{W} P(A_a^{S_n} \mid SX) \cdot P((NONE \mid SX) \mid A_a^{S_n}),$$

where each of $P(A_a^{S_n} \mid SX)$ is given by Bayes' rule, so that

$$P(X_2 \mid SX) = 1 - P(NONE \mid SX) = 1 - \sum_{a=1}^{W} \frac{P(A_a^{S_n})}{\left[\sum_{j=1}^{W} P(A_j^{S_n})\right]} \cdot \frac{(W-F_c)C_a}{{}_W C_a}. \qquad 5.15$$

Each of $P(A_1^{S_n}), ..., P(A_W^{S_n})$ can be calculated using Equation 5.12, with $m = n$. Note that for the case of a connection that has length one, there are no S block events, so that $A_a^{S_n}$ in Equation 5.15 are replaced by $A_a^{X_1}$. Each of $P(A_1^{X_1}), ..., P(A_{F_c}^{X_1})$ can be calculated using Equation 5.4.

5.4.4 The Probability of R_{C_i}

Equation 5.1 can now be solved using the formulas developed in this section to calculate $P(R_{C_i})$, for the given value of $LC_i = n+1$. Equation 5.1 is reproduced below, as Equation 5.16.

$$\begin{aligned} P(R_{C_i} \mid L_{n+1}) &= P(X_1 \cap S_1 \cap S_2 \cdots \cap S_n \cap X_2) \\ &= P(X_1) P(S_1 \mid X_1) P(S_2 \mid S_1 \cap X_1) \cdots P(S_n \mid S_{n-1} \cap \cdots \cap S_1 \cap X_1) \\ &\quad \cdot P(X_2 \mid S_n \cap \cdots S_1 \cap X_1). \end{aligned} \qquad 5.16$$

To make use of this result to calculate $P(R_{C_i})$, define $LC_i = l_{\max}$ as the maximum length of any connection and $L_{l_{\max}}$ as the corresponding event. Appropriate values for $l_{\max}$ are discussed in Section 5.5. Next, consider the events $L_1, ..., L_{l_{\max}}$ corresponding to the possible values of LC_i. Since the occurrence of R_{C_i} implies exactly one of $L_1, ..., L_{l_{\max}}$, then

$$P(R_{C_i}) = \sum_{l=0}^{l_{\max}} P(L_l) \cdot P(R_{C_i} \mid L_l), \qquad 5.17$$

where $P(L_l)$ are given by the probability distribution of connection length, referred to in Section 5.2 as P_L, and each $P(R_{C_i} \mid L_l)$ is defined by Equation 5.16. As mentioned in Section 5.2, P_L is assumed to be geometric, with mean $\bar{R}$. Thus, $P(L_l)$ is given by

$$P(L_l) = pq^{l-1},$$

where $p = \frac{1}{\bar{R}}$ and $q = 1 - p$. The following section shows how Equation 5.17 is evaluated to predict routability.

5.5 Using the Stochastic Model to Predict Routability

In order to make use of Equation 5.17, it is necessary to choose appropriate values for the various parameters that appear in the formulas developed in Section 5.4, as well as to evaluate the function λ_g, that is used to predict channel densities. This section first shows how λ_g is calculated and then gives appropriate values for each of the parameters. Following this, the routability predictions that are produced by the stochastic model are presented, with comparisons to the experimental results that were shown in Chapter 4.

As stated in Section 5.3, the parameter λ_g is defined by $\lambda_g = \frac{\lambda \bar{R}}{2}$, where $\bar{R}$ is the average connection length and λ is the ratio of the expected number of routed connections to the total number of logic cells. Given this definition, λ must be re-calculated after each connection is probabilistically 'routed' by the stochastic process. Thus, after $i-1$ connections have been 'routed', λ can be calculated as

$$\lambda = \frac{1}{N^2} \sum_{c=1}^{i-1} P(R_{C_c}). \qquad 5.18$$

It is necessary to assign values to the following parameters: N, W, $l_{\max}$, C_T, $\bar{R}$, $P(Z_1)$, α_1, α_2, F_s, and F_c. The first three of these depend on the size of the FPGA array and the next three are determined by the characteristics of the circuit to be routed. Since the routability predictions that are generated in this chapter are to be compared with the results from Chapter 4, the parameters will be taken from the FPGA circuits that were used in that chapter. The corresponding values are given in Table 4.1.

Circuit	N	W	$l_{\max}$	C_T	$\bar{R}$	$P(Z_1)$
BUSC	11	11	20	392	2.7	.71
DMA	15	12	28	771	2.8	.75
BNRE	20	14	38	1257	3.0	.75
DFSM	21	13	40	1422	2.85	.76
Z03	25	13	48	2135	3.15	.75

Table 4.1 - *Stochastic Model Parameters for Experimental Circuits*

	F_s									
	2	3	4	5	6	7	8	9	10	...
α_1	1.0	1.0	2.0	2.0	2.0	3.0	3.0	3.0	4.0	...
α_2	0.5	1.0	1.0	1.5	2.0	2.0	2.5	3.0	3.0	...

Table 4.2 - *Approximations to* α_1 *and* α_2

The parameters α_1 and α_2 can be approximated by making some assumptions concerning the topology of the S blocks. It is assumed here that the topology is similar to the one used in Chapter 4. This means that, as F_s is increased from its minimum value of 2, switches are added to the wiring segments in the order straight across, right turn, left turn, straight across, right turn, etc.. It is further assumed that the topology spreads the switches among the tracks such that every track can be switched to exactly F_s others.

Given these assumptions, appropriate values of α_1 and α_2 are shown in Table 4.2.

5.5.1 Routability Predictions

Recall, from Section 5.2, that routability is defined as

$$Routability = \frac{1}{C_T} \sum_{i=1}^{C_T} P(R_{C_i}), \qquad 5.19$$

This equation can now be evaluated using Equation 5.17, the formulas developed in Section 5.4, Equation 5.18, and Tables 4.1 and 4.2. A typical result is shown in Figure 5.7, which gives a plot of the expected percentage of successfully completed connections versus connection block flexibility, F_c, for parameters that correspond to the circuit BNRE.

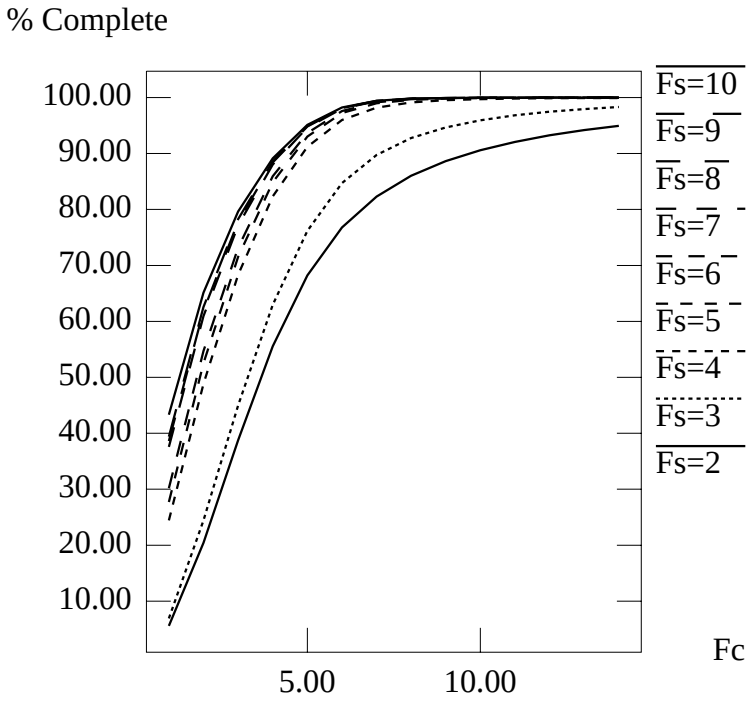

Figure 5.7 - *Routability Predictions vs. F_s, for Circuit BNRE*

Figure 5.7 is analogous to Figure 4.9. Each curve in the figure corresponds to a different value of S block flexibility, F_s. The lowest curve represents the case $F_s = 2$ and the highest curve corresponds to $F_s = 10$. The figure indicates that the routability is low for small values of F_c and only approaches 100% when F_c is at least one-half of W. The figure also shows that increasing the S block flexibility improves the completion rate at a given F_c, but to get near 100% the value of F_c must always be high (above 7 for this circuit). The reader will note that these are the same conclusions that were reached experimentally in Chapter 4.

Figure 5.8 is a plot of the expected percentage of successfully completed connections versus S block flexibility, F_s, also for the circuit BNRE. This figure is analogous to Figure 4.12. Each curve in the figure corresponds to a different value of F_c, with the lowest curve representing $F_c = 1$ and the highest curve corresponding to $F_c = W$. The curves show an increase in slope at F_s values of 4, 7, and 10. This occurs because switches are added straight across the S blocks for these values of F_s and, as Table 4.1 shows, connections pass straight through the S blocks more than 70 percent of the time. Figure 5.8 shows that if F_c is at least half of W, then very low values of F_s approach 100% routability. Again, this is the same conclusion reached in Chapter 4.

While the theoretical and experimental results lead to the same general conclusions, they are not identical. Figure 5.9 compares the routability results produced by the stochastic model with the experimental results. The dashed curve corresponds to the model, whereas the solid curve is produced experimentally. Both curves correspond to circuit BNRE, with $F_s = 6$. As Figure 5.9 indicates, the two results are quite similar. The fact that the theoretical curve is lower than the experimental curve for low values of F_c is due in part to Equation 5.14, which, as discussed in Section 5.4, does not accurately represent

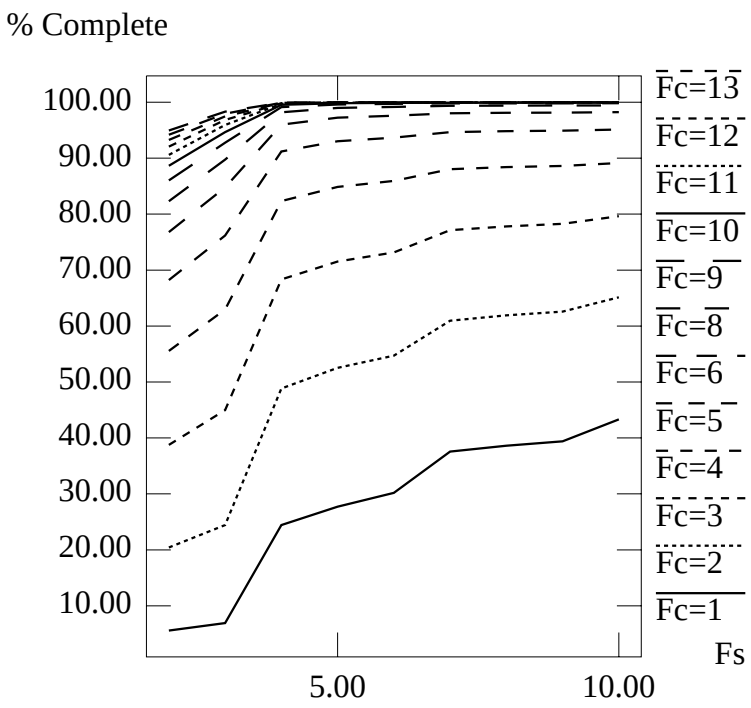

Figure 5.8 - *Routability Predictions vs. F_c, for Circuit BNRE*

good C block topologies. A summary of comparisons between theory and experiment for all of the circuits appears in Table 5.3. For each circuit, the table shows the difference between the theoretical and experimental routability results, for each value of F_s. Each entry gives the mean value (and standard deviation) of the difference between the experimental and theoretical routabilities, over the range of values of F_c from 1 to *W*. The values in the table are in percentages since those are the units of routability. Absolute values are used in the table to avoid a misleading average that could be caused by combining negative and positive differences. However, this is not really necessary since, as Figure 5.9 indicates, the theoretical predictions are almost always pessimistic. As Table 5.3 shows, the experimental measurements and theoretical predictions of routability are surprisingly close, especially for values of F_s greater than three.

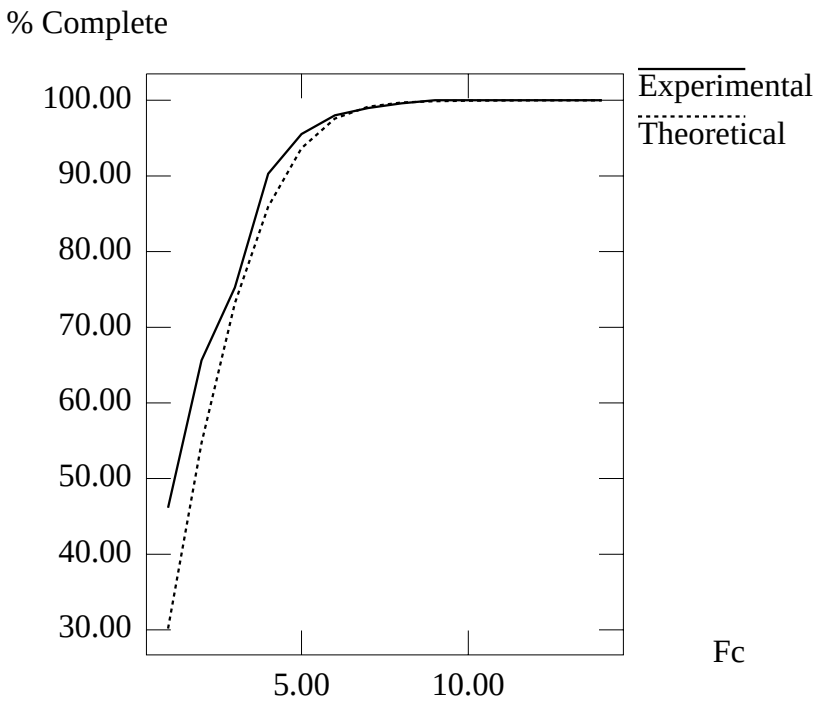

Figure 5.9 - *Comparison of Predictions and Experiments for $F_s = 6$*

	BUSC		DMA		BNRE		DFSM		Z03	
F_s	Mean	S.D.	Mean	S.D	Mean	S.D.	Mean	S.D.	Mean	S.D.
2	7.7	4.9	10.2	7.5	7.3	6.3	8.9	8.3	7.2	5.9
3	9.7	5.6	12.5	8.1	8.7	6.7	10.8	9.4	10.2	5.6
4	2.9	2.9	4.1	4.5	1.5	3.1	2.7	5.3	1.9	2.1
5	3.7	4.3	4.9	5.8	2.4	4.3	3.7	6.1	1.8	2.7
6	3.2	3.5	5.0	6.2	2.6	4.7	4.0	6.7	2.1	3.3
7	4.8	4.3	5.1	6.6	2.8	4.3	3.9	6.1	1.8	2.8
8	4.3	4.6	5.1	6.5	3.1	4.3	4.1	6.2	2.2	2.6
9	4.3	4.9	5.0	6.3	3.2	4.4	4.2	6.2	2.5	3.0
10	4.3	4.8	5.2	6.7	3.2	4.3	4.2	5.9	2.9	3.4

Table 5.3 - *Summary of Comparisons Between Theory and Experiment*

5.6 Conclusions

This chapter has developed a stochastic model that can be used to study the effect of the flexibility of an FPGA's routing architecture on its routability. It has been shown that the model can be used to reach the same conclusions that were generated in Chapter 4 using an experimental approach. In future work the model could be extended to handle the case where some of the routing switches in the S blocks are replaced by hard-wired connections. This would allow the modelling of routing architectures in which the tracks may be composed of segments of various lengths and would allow the stochastic model to be used to study such architectures.

6 Conclusions

6.1 Thesis Summary

The main focus of this thesis has been the study of FPGA routing architectures with regard to the tradeoff among routability, area and speed performance. The study's purpose has been to determine the level of flexibility of routing architectures that is high enough so that real circuits can be successfully routed, and yet low enough so that routing switches are not wasted. The research has been carried out using both experimental and theoretical approaches. In the experimental study, a new type of detailed routing algorithm, parameterized for different levels of flexibility, has been used to measure the effects of flexibility on the routability of circuits. Theoretical methods have also been applied to study this issue, using a stochastic modelling approach that predicts, based on combinatorial analysis, the effect of flexibility on routability. The results of the work in the thesis provide new insights into the design of FPGA routing algorithms and routing architectures.

6.2 Thesis Contributions

In Chapter 3, the Coarse Graph Expansion (CGE) detailed routing algorithm for FPGAs was described. It is the first published algorithm that approaches FPGA routing in a general way, and it can be used over a wide range of routing architectures. CGE employs a heuristic approach that bases routing decisions on a global cost function that accounts for contention for the routing resources that may exist among the connections in a circuit. The cost function also allows the optimization of routing delays for time-critical connections. CGE has been used to obtain excellent routing results for relatively large FPGA routing problems.

Chapter 4 gives the results of an experimental study of the effects on routability of the flexibility of FPGA routing architectures. This study is the first of its kind for FPGAs. The principal conclusions reached are that a high level of connectivity is desirable in the connection blocks that join the logic cell pins to the routing channels, but that a relatively low flexibility is sufficient in the switch blocks at the intersections of horizontal and vertical channels. Also, it is shown that, even with surprisingly low levels of flexibility, circuits can be routed with very close to the theoretical minimum number of tracks per channel. Finally, the results show the effect of the flexibility of the routing architecture on the total number of routing switches needed in an FPGA.

A theoretical foundation that can be used to study FPGA routing architectures is developed in Chapter 5. A stochastic model is presented that uses probability theory to predict the routability of circuits over a range of flexibility. The probability expressions are based on combinatorial analysis that accounts for both the flexibility of the routing architecture and the side-effects that the successful routing of one connection has on others. It is shown that the stochastic model can be used to reach the same conclusions that were found experimentally in Chapter 4.

6.3 Suggestions for Future Research

A future enhancement for the CGE detailed router could improve upon the usage of routing tracks comprising wiring segments that span multiple logic cells. Currently, special attention is paid to the segmentation-length of tracks only when routing time-critical connections. A better approach would be for the router to assign tracks to each connection such that the segmentation-length of the track matches, as closely as practical, the length of the connection. Both phases of the algorithm are affected by this issue. First, when paths are pruned from expanded graphs during graph expansion, those paths that

correspond to tracks that have appropriate segment lengths should be given preference. Second, during path selection, segment lengths should be considered when specific tracks are chosen for each connection. These modifications to the algorithm's implementation should be integrated with the current cost function so that a high quality of routing can still be obtained, but with improved usage of segmented tracks.

A topic of future research that may yield significant results is the development of a near-optimal detailed router for FPGAs. The following discussion describes how such a router could be realized by building upon the results of this thesis. The CGE algorithm follows a heuristic approach; a pruning procedure controls the complexity of the routing problem, and an approximate cost function is used to resolve routing conflicts. The heuristic approach is necessary for two reasons:

(1) The number of possible detailed routes for a connection is an exponential function of the length of the connection.

(2) The number of connections to be routed is large.

Using the routing architecture flexibility results in this thesis, issue (1) can be removed by constraining the flexibility of the switch blocks ($F_S \leq 3$). While issue (2) still remains, it too can be eliminated by using the CGE router. An interesting scheme would be one that uses CGE to complete the detailed routes of all but the most difficult connections in a circuit. Then, for channel segments that still contain unrouted connections, a final routing stage could be invoked to try to complete the routing task. Having greatly reduced the scope of the problem, it should be practical to implement the final routing stage using some sort of optimal assignment algorithm. Note that the overall solution is still only "near-optimal" since there might be some influence on the final routing stage of decisions made by the heuristic.

The routing architecture results that are presented in Chapter 4 assume routing tracks that have a segmentation-length of one. Future research should expand this study to include other segment lengths. There are three issues to be addressed:

(1) How many tracks of each segmentation-length should be included?

(2) What connectivity should be available among the various tracks?

(3) How should the segmented tracks be populated? For segmented tracks, this issue refers to whether or not there should be routing switches, in the connection and switch blocks, that connect to the middle sections of segments.

The stochastic model described in Chapter 5 also assumes that tracks have a segmentation-length of one. The theory should be extended to include other segment lengths so that the model could be used to study such architectures. Appropriate modifications would also be necessary for the method that is used to approximate channel density, since it is also based on non-segmented channels.

References

[Ahre90]

M. Ahrens, A. El Gamal, D. Galbraith, J. Greene, S. Kaptanoglu, K. Dharmarajan, L. Hutchings, S. Ku, P. McGibney, J. McGowan, A. Samie, K. Shaw, N. Stiawalt, T. Whitney, T. Wong, W. Wong and B. Wu, "An FPGA Family Optimized for High Densities and Reduced Routing Delay," *Proc. 1990 Custom Integrated Circuits Conference*, May 1990, pp. 31.5.1 - 31.5.4.

[Aker72]

S.B. Akers, "Routing," Chapter 6 of *Design Automation of Digital Systems; Theory and Techniques*, M.A. Breuer, Ed., NJ, Prentice-Hall, 1972.

[Alt90]

*The **Max**imalist Handbook*, Altera Corp., 1990.

[AMD90]

MACH 1 and MACH 2 Device Families Preliminary Data Sheets, 1990.

[Bray86]

R. Brayton, E. Detjens, S. Krishna, T. Ma, P. McGeer, L.Pei, N. Phillips, R. Rudell, R. Segal, A. Wang, R. Yung and A. Sangiovanni-Vincentelli, "Multiple-Level Logic Optimization System," *Proc. IEEE International Conference on Computer Aided Design*, pp. 356-359, Nov. 1986.

[Breu77]

M.A. Breuer, "Min-Cut Placement," *Journal of Design Automation and Fault Tolerant Computing*, pp. 343-362, Oct. 1977.

[Brow90]

S. Brown, J. Rose and Z.G. Vranesic, "A Detailed Router for Field-Programmable Gate Arrays", *Proc. IEEE International Conference on Computer Aided Design*, pp. 382-385, Nov. 1990.

[Brow91]

S. Brown, J. Rose and Z.G. Vranesic, "Routing in Field-Programmable Gate Arrays", to appear in *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, 1991.

[Cart86]

W. Carter, K. Duong, R. H. Freeman, H. Hsieh, J. Y. Ja, J. E. Mahoney, L. T. Ngo and S. L. Sze, "A User Programmable Reconfigurable Gate Array," *Proc. 1986 Custom Integrated Circuits Conference*, May 1986, pp. 233-235.

[Cong88]

J. Cong and B. Preas, "A New Algorithm for Standard Cell Global Routing," *Proc. IEEE International Conference on Computer Aided Design*, pp. 176-179, Nov. 1988.

[ElAy88]

K. El-Ayat, A. El Gamal and A. Mohsen, "A CMOS Electrically Configurable Gate Array," *Int'l Solid State Circuits Conf. Digest of Technical Papers*, Feb. 1988.

[ElGa81]

A. El Gamal, "Two-Dimensional Stochastic Model for Interconnections in Master Slice Integrated Circuits" , *IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems*, Vol. CAS-28, No. 2, February 1981.

[ElGa88]

A. El Gamal, J. Greene, J. Reyneri, E. Rogoyski, K. El-Ayat and A. Mohsen, "An Architecture for Electrically Configurable Gate Arrays," *Proc. 1988 Custom Integrated Circuits Conference*, May 1988, pp. 15.4.1 - 15.4.4.

[ElGa89]

A. El Gamal, J. Greene, J. Reyneri, E. Rogoyski, K. El-Ayat and A. Mohsen, "An Architecture for Electrically Configurable Gate Arrays," *IEEE Journal of Solid State Circuits* Vol. 24, No. 2, April 1988, pp. 394-398.

[Fell52]

W. Feller, *Introduction to Probability Theory and its Applications*, John Wiley and Sons, 1952.

[Fran90]

R.J. Francis, J. Rose and K. Chung, "Chortle: A Technology Mapping Program for Lookup Table-Based Field-Programmable Gate Arrays," *Proc. 27th Design Automation Conference*, June 1990, pp. 613-619.

[Fran91]

R. J. Francis, J. Rose, and Z. Vranesic, "Chortle-crf: Fast Technology Mapping for Lookup Table-Based FPGAs," *Proc. 28th Design Automation Conference, June 1991, pp. .*

[Green90]

J. Greene, V. Roychowdhury, S. Kaptanoglu, and A. El Gamal, "Segmented Channel Routing," *Proc. 27th Design Automation Conference*, pp. 567-572, June 1990.

[Greg86]

D. Gregory, K. Bartlett, A. de Geus and G. Hachtel, "Socrates: a system for automatically synthesizing and optimizing combinational logic," *Proc. 23rd Design Automation Conference*, June 1986, pp. 79-85.

[Gupt90]

A. Gupta, V. Aggarwal, R. Patel, P. Chalasani, D. Chu, P. Seeni, P. Liu, J. Wu and G. Kaat, "A User Configurable Gate Array Using CMOS-EPROM Technology," *Proc. 1990 Custom Integrated Circuits Conference*, May 1990, pp. 31.7.1 - 31.7.4.

[Hana72]

M. Hanan and J.M. Kurtzberg, "Placement Techniques," Chapter 4 of *Design Automation of Digital Systems; Theory and Techniques*, M.A. Breuer, Ed., NJ, Prentice-Hall, 1972.

[Hash71]

A. Hashimoto and J. Stevens, "Wire routing by optimizing channel assignment within large apertures," *Proc. 8th Design Automation Conference*, June 1971, pp. 155-163.

[Heller84]

W.R. Heller, C.G. Hsi and W.F. Mikhaill, "Wirability - Designing Wiring Space for Chips and Chip Packages," *IEEE Design and Test of Computers*, August 1984.

[Hsie88]

H. Hsieh, K. Duong, J. Ja, R. Kanazawa, L. Ngo, L. Tinkey, W. Carter and R. Freeman, "A Second Generation User-Programmable Gate Array," *Proc. 1987 Custom Integrated Circuits Conference*, May 1987, pp. 515 - 521.

[Hsie90]

H. Hsieh, W. Carter, J. Ja, E. Cheung, S. Schreifels, C. Erickson, P. Freidin, L. Tinkey and R. Kanazawa, "Third-Generation Architecture Boosts Speed and Density of Field-Programmable Gate Arrays" *Proc. 1990 Custom Integrated Circuits Conference*, May 1990, pp. 31.2.1 - 31.2.7.

[Kahr86]

M. Kahrs, "Matching a parts library in a silicon compiler," *Proc. IEEE International Conference on Computer Aided Design*, pp. 169-172, Nov. 1986.

[Kawa90]

K. Kawana, H. Keida, M. Sakamoto, K. Shibata and I. Moriyama, "An Efficient Logic Block Interconnect Architecture For User-Programmable Gate Array," *Proc. 1990 Custom Integrated Circuits Conference*, May 1990, pp. 31.3.1 - 31.3.4.

[Keut87]

K. Keutzer, "DAGON: Technology Binding and Local Optimization by DAG Matching," *Proc. 24th Design Automation Conference*, June 1987, pp. 341-347.

[Lee61]

C. Lee, "An algorithm for path connections and its applications," *IRE Transactions on Electronic Computers*, VEC-10, pp. 346-365, Sept. 1961.

[Lee88]

K. Lee and C. Sechen, "A New Global Router for Row-Based Layout," *Proc. IEEE International Conference on Computer Aided Design*, pp. 180-183, Nov. 1988.

[Loren89]

M.J. Lorenzetti and D.S. Baeder, Chapter 5 of "Physical Design Automation of VLSI Systems," B. Preas and M. Lorenzetti, Ed., Benjamin/Cummings, 1989.

[Marr89]

C. Marr, "Logic Array Beats Development Time Blues," *Electronic System Design Magazine*, Nov. 1989, pp. 38-42.

[Mail90]

F. Mailhot, Actel Corp., *Private Communication*, 1990.

[Ples89]

Plessey Semiconductor ERA60100 Advance Information, Nov. 1989.

[Plus90]

Plus Logic FPGA2020 Preliminary Data Sheet, 1990.

[Prim57]

R. Prim, "Shortest Connecting Networks and Some Generalizations," Bell System Technical Journal, Vol. 39, pp. 1389-1401, 1957.

[Rose85]

J. Rose, Z. Vranesic and W.M. Snelgrove, "ALTOR: An Automatic Standard Cell Layout Program," *Proc. Canadian Conference on VLSI*, Nov. 1985, pp. 168-173.

[Rose89]

J.S. Rose, R.J. Francis, P. Chow and D. Lewis, "The Effect of Logic Block Complexity on Area of Programmable Gate Arrays," *Proc. 1989 Custom Integrated Circuits Conference*, May 1989, pp. 5.3.1-5.3.5.

[Rose90a]

J. Rose, "Parallel Global Routing for Standard Cells," *IEEE Transactions on Computer Aided Design* Vol. 9, No. 10, pp. 1085-1095, Oct. 1990.

[Rose90b]

J. Rose and S. Brown, "The Effect of Switch Box Flexibility on Routability of Field Programmable Gate Arrays," *Proc. 1990 Custom Integrated Circuits Conference*, pp. 27.5.1-27.5.4, May 1990.

[Rose90c]

J.S. Rose, R.J. Francis, D. Lewis and P. Chow, "Architecture of Programmable Gate Arrays: The Effect of Logic Block Functionality on Area Efficiency," *IEEE Journal of Solid State Circuits*, Vol. 25, No 5, October 1990, pp. 1217-1225.

[Rose91]

J. Rose and S. Brown, "Flexibility of Interconnection Structures in Field-Programmable Gate Arrays", *IEEE Journal of Solid State Circuits*, Vol. 26 No. 3, pp. 277-282, March 1991.

[Sech87]

C. Sechen and K. Lee, "An Improved Simulated Annealing Algorithm for Row-Based Placement," *Proc. IEEE International Conference on Computer Aided Design*, Nov. 1987, pp. 478-481.

[Sing91]

S. Singh, J. Rose, D. Lewis, K. Chung and P. Chow, "Optimization of Field-Programmable Gate Array Logic Block Architecture for Speed," to appear in *Custom Integrated Circuits Conference*, 1991.

[Souk81]

J. Soukup, "Circuit Layout," *Proc. of the IEEE*, Vol. 69, No. 10, pp. 1281-1304, October 1981.

[Wong89]

S.C. Wong, H.C. So, J.H. Ou and J. Costello, "A 5000-Gate CMOS EPLD with Multiple Logic and Interconnect Arrays," *Proc. 1989 Custom Integrated Circuits Conference*, May 1989, pp. 5.8.1 - 5.8.4.

[Xili89]

The Programmable Gate Array Data Book, Xilinx Co., 1989.