Modeling Routing Demand for Early-Stage FPGA Architecture Development

by

Wei Mark Fang

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science
Graduate Department of Electrical and Computer Engineering
University of Toronto

Abstract

Modeling Routing Demand for Early-Stage FPGA Architecture Development

Wei Mark Fang

Master of Applied Science

Graduate Department of Electrical and Computer Engineering

University of Toronto

2007

There is increasing interest in developing Field Programmable Gate Arrays (FPGA) architectures with new logic blocks containing specific functionality, such as special-purpose computational blocks and processors. In the development of new FPGAs, a key part in the evaluation of different logic block ideas and routing architectures is to determine their impact on the need for routing wires. This demand is typically determined through laborious and compute-intensive experimentation which is not possible in the early stages of architecture development. This gives rise to the need for FPGA routing demand models that could be employed at this stage.

This thesis presents an FPGA interconnect model that predicts routing demand, to guide the architect in the early stages of architecture development. The goal is to produce a model that is as simple as possible but still usefully accurate so that knowledge, understanding and intuition can be transmitted by the model. The inputs to the model are a characterization of the logic block architecture and a set of well-known routing architecture parameters. The output is the number of routing tracks per channel required for that logic block and routing architecture, in an island-style FPGA.

Acknowledgements

First I would like to thank my supervisor, Professor Jonathan Rose, for his guidance, support and encouragement throughout the course of my research. I wish to express my sincere gratitude to him for teaching me a great deal and guiding my development not only as a researcher but as a person.

I am grateful for funding from the Natural Sciences and Engineering Research Council (NSERC) through a CGS scholarship.

I would like to give special thanks to Ian Kuon and Peter Jamieson for helping me in countless ways during my degree and making it so much more fun.

I would like to thank all of my friends and colleagues in the Computer Group, particularly in LP392, including: Navid, Blair, Peter Y., Tomasz, Franjo, Sean, Imad, Lesley, Nahi, Wei, Jason and Jeff.

Last but certainly not least, I would like to thank my parents for their unwavering support, encouragement, and guidance. This thesis is as much an achievement of theirs as it is mine.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, there has been increasing interest in developing new Field Programmable Gate Arrays (FPGA) architectures. In particular there is interest in developing FPGA architectures with new logic blocks containing more specific functionality including special-purpose computational blocks, memories and even processors [3, 5, 44, 47, 63]. For each new logic block architecture, a key part of its evaluation is to determine its need for routing wires, and its impact on the need for extra routing wires when inflexibility is introduced to the routing architecture. This routing demand is typically measured through laborious and compute-intensive experimentation, requiring the synthesis of benchmark circuits into candidate architectures. This is difficult to do in the early stages of architecture development, however, as there is no complete architecture to synthesize circuits into. Furthermore, the effort required to create prototype tools and run time-consuming experiments for nascent architectures is too great to do for every new logic block and routing architecture idea. Therefore there is a need for an FPGA routing demand model that is simple to use, and provides understanding and intuition on FPGA routing and its interaction with logic block architecture, to guide the architect in the early stages of architecture development.

However, there is a lack of models for this purpose. Most of previous interconnect models in literature do not consider the inflexibility of FPGA routing architecture. The few models that are FPGA-specific consist of complex equations, from which it is difficult to draw intuition on FPGA routing to guide an architect in early-stage architecture development.

This research presents an FPGA interconnect model that predicts routing demand, to guide an FPGA architect in early-stage architecture development. The inputs to the model are a characterization of the logic block architecture and a set of well-known routing architecture parameters. The output of the model is the routing demand/track count (often referred to as W, the number of routing tracks per channel) required for that logic block and routing architecture, in an island-style FPGA. The model's inputs and output are visually shown in Figure 1.1.

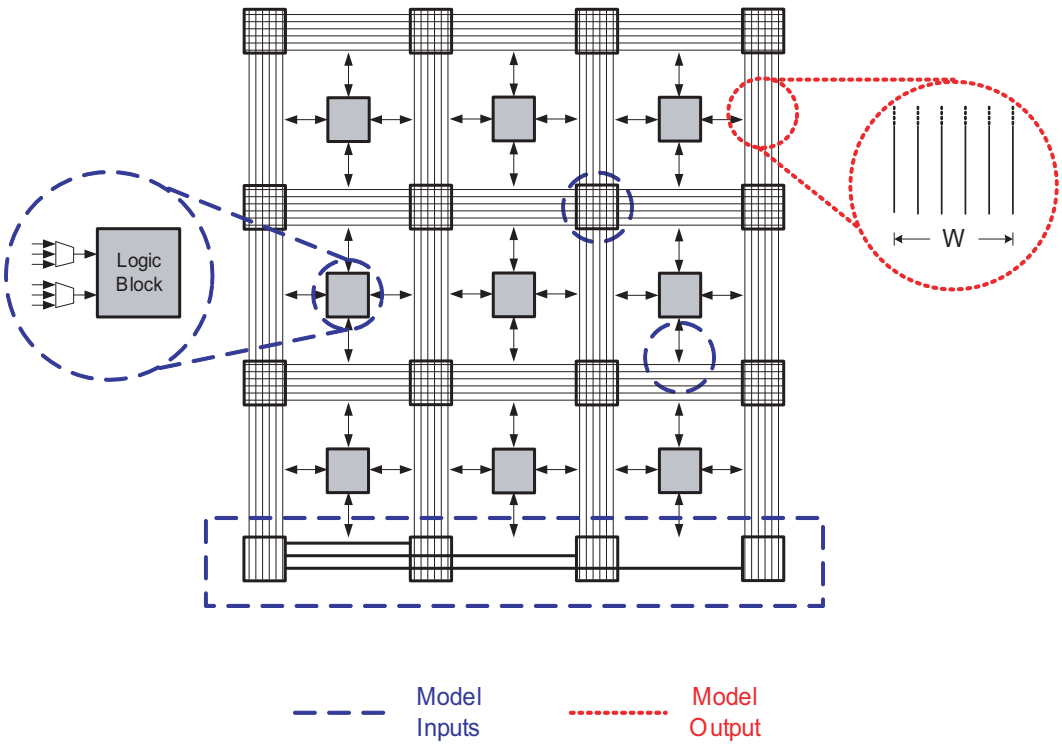


Figure 1.1: FPGA routing demand model inputs and output

The specific goal is to produce such an FPGA interconnect model that is as simple as possible, while still usefully accurate, and provides understanding and intuition on FPGA routing and its interaction with logic block architecture. The model is intended to guide an architect in evaluating new logic block architectures for its impact on routing demand and routing architecture exploration, in early-stage architecture development, in the absence of empirical methods and benchmark circuits. While this model is intended to replace empirical methods

in the early stages of architecture development, for providing quick preliminary feedback, the final architecture decisions should still be based on detailed experimentation.

The modeling approach is an empirical one, in combination with derivations from intuitive observations. We begin with a simple and intuitive model for the required channel width of a routing architecture of extreme overpopulation of connectivity – the fully flexible FPGA. We then incrementally reduce the flexibility of the modeled routing architecture, by reducing flexibility in routing architecture parameters. At the end of this gradual process, we arrive at the final routing demand model.

The logic block architectures used in this empirical approach are clusters of lookup table based logic blocks. While this does not test the model for new logic blocks such as special-purpose computational blocks, memories and processor cores, it has the advantage that the quality of the results can be compared to experimentally measured data. Furthermore in Chapter 5 we discuss how to apply the model for these new logic blocks by estimating parameters characterizing the logic block architecture.

1.1 Organization

This thesis is organized as follows. Chapter 2 provides the background on FPGA architecture terminology and tool flow. It also reviews a large body of routing models in literature, some of which this work draws from.

Chapter 3 presents a detailed description of the modeling methodology.

Chapter 4 discusses the creation of the model as it evolves into a complete equation predicting track count, as a function of logic block and routing architecture parameters.

Chapter 5 demonstrates the application of the model in early-stage architecture development. This includes providing feedback on evaluating routing demand of logic block architectures, exploring routing architecture space without tools and circuits, and estimating track count for a commercial FPGA architecture.

Finally, Chapter 6 concludes this work and suggests potential avenues for future research.

Chapter 2

Background and Previous Work

This chapter begins by reviewing FPGA architecture terminology, including the logic block architecture and routing architecture that is our goal to model. Next an outline the CAD flow required to implement a circuit onto an FPGA architecture is given. Then we give an overview of general ASIC interconnect models from classical to contemporary. This chapter concludes with a review of FPGA specific interconnect models which are based on the general ASIC models.

2.1 FPGA Architecture Terminology

This section reviews the most common architecture in research for SRAM-based FPGAs, the island-style FPGA architecture. This architecture forms the basis for the present work.

The island-style FPGA architecture uses a symmetric structure in which logic blocks (LB) are laid out in an array of islands, as shown in Figure 2.1. The logic blocks are surrounded by routing channels, where routing wires lie. At the periphery of the array are IO blocks that connect the FPGA's logic blocks to off-chip devices.

We will describe the details of the logic block in the following subsection.

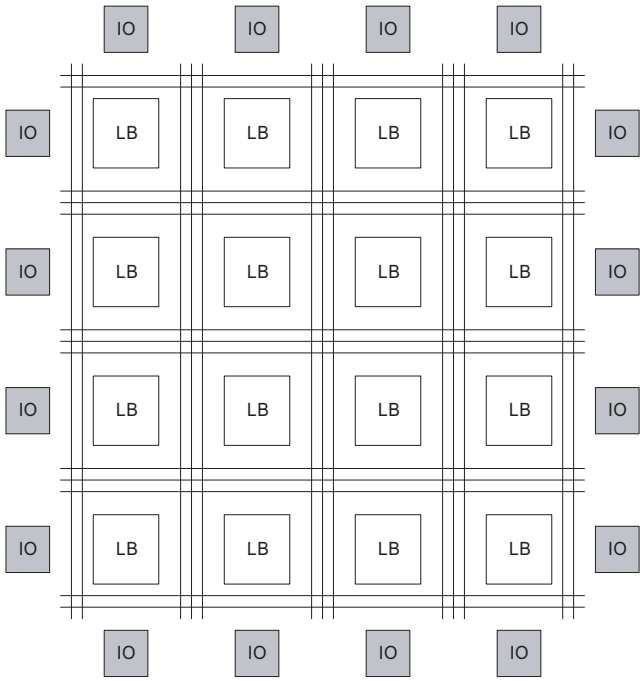


Figure 2.1: The island-style FPGA architecture

2.1.1 The Logic Block Architecture

The most common logic block architecture is made up of a group/cluster of basic logic elements (BLEs) [9]. This is called a cluster-based logic block, or simply a logic cluster as shown in Figure 2.2 (b). The number of inputs to the logic cluster is I, and the number of outputs equals the number of BLEs in the logic cluster, the cluster size N. Each BLE consists of a K-input lookup table (LUT) and a register. A two-input multiplexer is used to provide either a registered or unregistered BLE output as shown in Figure 2.2 (a). Each logic cluster is "fully connected" meaning that all I inputs and N BLE outputs can connect to each of the K inputs on every LUT. This connectivity is implemented using a multiplexer on each of the BLE input as shown in Figure 2.2 (b).

Previous research [1] has shown that the number of input pins to the logic cluster I should be set according to Equation 2.1.1.1 to permit complete use of BLEs in the cluster. In this work we adopt this architectural choice.

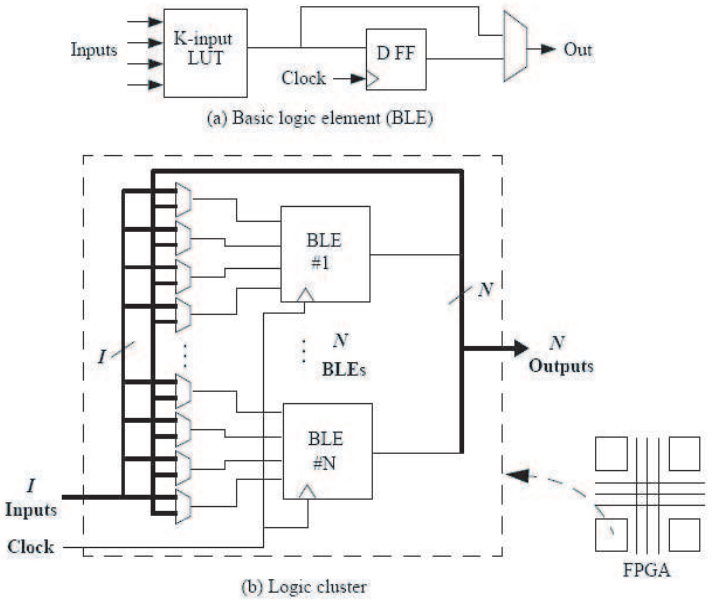


Figure 2.2: Structure of (a) basic logic element (BLE) and (b) logic cluster [9]

$$I = \frac{K}{2}(N+1) \tag{2.1.1.1}$$

We assume the input and output pins of the logic cluster are evenly distributed among the four sides of the logic cluster in the manner shown in Figure 2.3, which shows an example logic cluster ($N = 4$, $K = 4$ and $I = 10$).

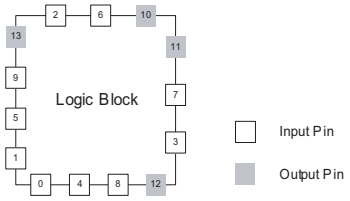


Figure 2.3: Location of input and output pins on the logic cluster

In commercial FPGAs more complex logic cluster architectures are used. For example the Altera Stratix II adopts a logic cluster of size 8, each basic logic element being the Adaptive Logic Module (ALM) shown in Figure 2.4. The ALM contains a fracturable LUT [42] with

additional carry-chain circuitry. While this commercial logic block is far more complex than our logic block, it has the same basic hierarchical structure. We believe the logic block architecture we consider is representative of commercial and future FPGA logic block architectures.

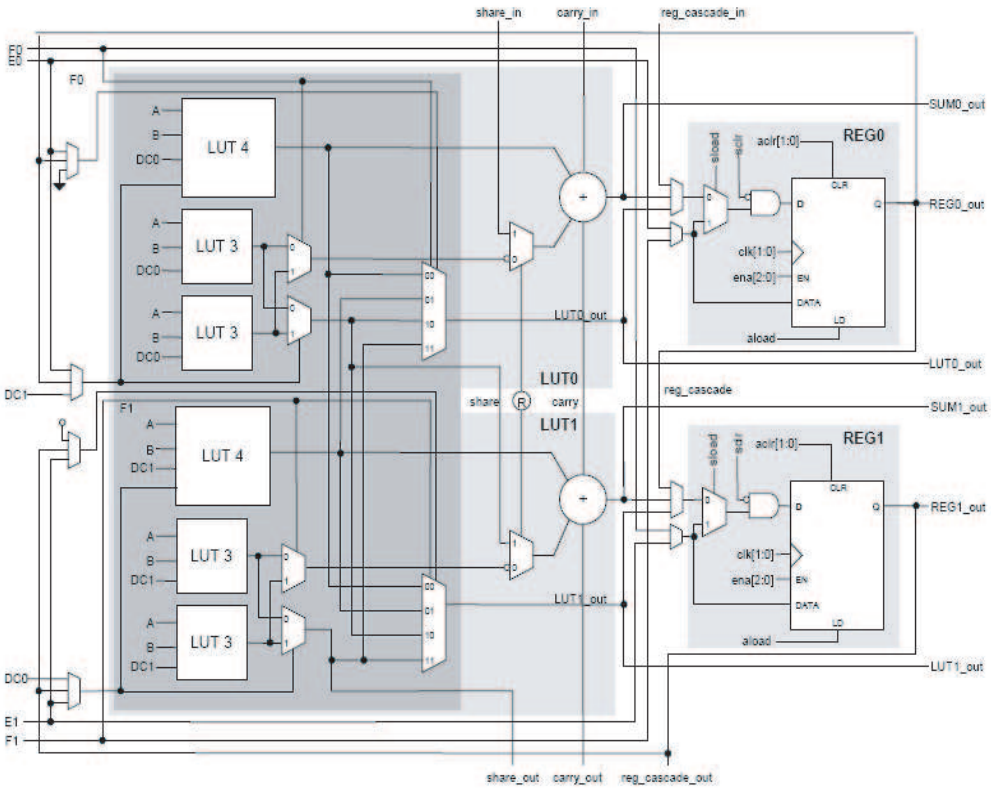


Figure 2.4: A commercial logic block architecture: the Altera Stratix II ALM [4]

The logic blocks implement the logic functionality of circuits and some local connections internal to the logic block. To fully implement the circuit logic blocks must be connected using the FPGA routing architecture discussed in the next subsection.

2.1.2 The Routing Architecture

In this subsection we will describe the routing architecture space considered in this work. Along the way we review a number of routing architecture parameters, many of which were introduced in [51] and have become the standard method for describing island-style FPGA

routing architecture. These parameters parameterize a large space of routing architectures, and we will use them as inputs to our model.

In the island-style FPGA architecture, between the rows and columns of logic blocks are horizontal and vertical routing channels. Each routing channel spans the length of the array. The portion of a channel adjacent to a logic block is a channel segment. In each channel segment there are **W** wire tracks, where routing wires lie. This is often referred to as the *channel width*, or *track count*. We assume that all channel segments have the same width as Betz et al. in [9] showed there is little advantage to making them different.

In this work we also assume that the detailed routing architecture employs the single-driver approach, studied by Lemieux et al. in [38], where each wire can only be driven by a single source, chosen by a multiplexer followed by a buffer at the beginning of the wire as shown in Figure 2.5. This is now the common standard in industry. Altera FPGAs use this wire design and refer to it as *direct drive* wires [41]. Xilinx FPGAs use the same approach and refer to it as *unidirectional* wires [64].

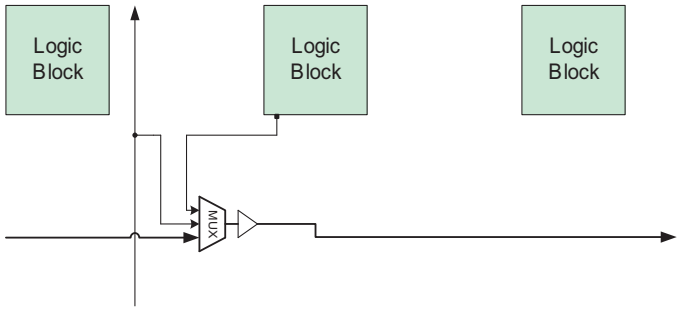


Figure 2.5: Example of FPGA single-driver interconnect

The **length** of a wire **L** is the number of logic blocks it spans. A wire that spans multiple logic blocks is referred to as a *starting wire* in the location containing its multiplexer. The example wire shown in Figure 2.5 has length 2. Betz et al. in [9] studied such routing architectures for bidirectional wires that are pass transistor and buffer switched. He suggested a mix of medium length wires length 4 and length 8 for best area delay.

In a commercial routing architecture, such as that of Altera's Stratix II FPGA [42], 89% of the tracks run length 4 wires (208 horizontally and 128 vertically), and the rest consists of

long wires of length 24 (24 horizontally) and 16 (16 vertically). Another commercial routing architecture, that of Xilinx's Virtex 4 FPGA [65], has 65% length 6 wires (120) and 22% length 2 wires (40) and 13% chip length long wires (24), both horizontally and vertically.

For the scope of this work, we will assume all wires in the detailed routing architecture have the same length. Commercial routing architectures have a mix of wire lengths to achieve optimal performance, whereas our goal is to study the routability of different wire lengths. Length will be varied from 1 to 8 to model a large routing architecture space.

Between channel segments, at the intersection of the horizontal and vertical channels, are **switch blocks** [51]. In an switch block, labeled as S in Figure 2.6, routing switches are used to connect wires for turning corners or extending farther down a channel. The flexibility of the switch block, **Fs**, is defined to be the total number of starting wires adjacent to the switch block each incoming wire can connect using routing switches. Routing switches are implemented as multiplexer that choose the driver of a wire. The switch block in Figure 2.6 (a) uses a dashed line to indicate a routing switch between wires.

A **connection block** [51] conceptually contains the connection between the logic block and the routing wires in the channel segments. It is separated into the input connection block and the output connection block. The input connection block is a set of routing switches, implemented by multiplexers, selecting routing wires from the adjacent channel segment to connect to the logic block input pins. The flexibility of the input connection block, $\mathbf{Fc_{in}}$, is defined to be the total number of wires that can connect to each input pin through routing switches. These routing switches are depicted in Figure 2.6 (b) as circles.

The flexibility of the output connection block, Fc_{out}, is defined to be the total number of starting wires in local channel segments that each output pin can connect via routing switches. The connection block in Figure 2.6 (b) uses a dashed line to indicate a routing switch between output pin and wire. The output pin routing switch is implemented by the same multiplexer that chooses the driver of a wire as the Fs routing switch, hence Figure 2.6 shows them both as dashed lines going into these multiplexers.

The input and output pins on a logic block can be either **logically equivalent** or not. Logical equivalence between a set of pins means they can be swapped for one another without

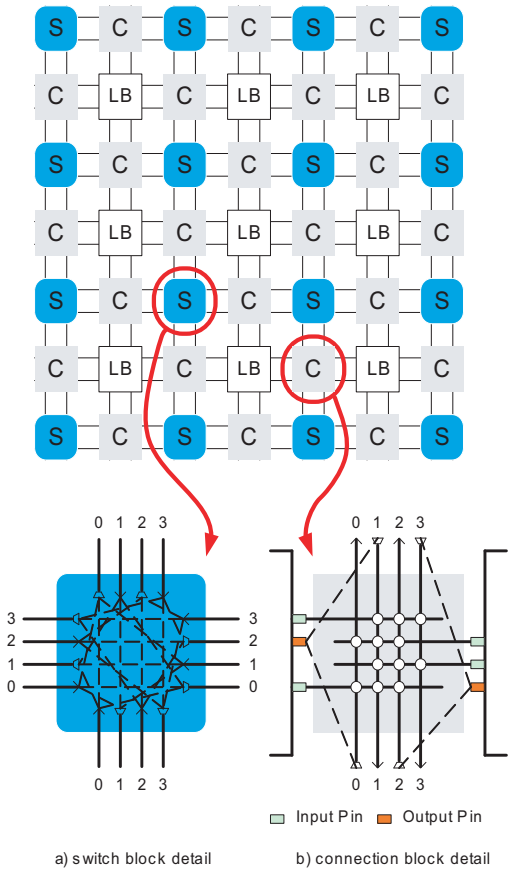


Figure 2.6: Island-style FPGA with single-driver routing architecture [38]

change to the functionality of the logic, such as the inputs of an AND gate. In Section 2.1.1 it was assumed logic clusters are "fully connected", and this means the I input pins of the cluster are logically equivalent. This gives routing flexibility since pins are evenly distributed among the four sides of the logic block as in Figure 2.3 enabling programmable connections to wires in multiple channel segments when the pins are logically equivalent. This architecture feature is represented using the binary parameter **Eqv**. We consider the input pins and output pins to either all be logically equivalent ($Eqv = 1$), or not ($Eqv = 0$).

In sum there are five routing architecture/flexibility parameters we consider: Fs, Fc_{in}, Fc_{out}, L, and Eqv. We will use these routing flexibility parameters as inputs to the routing model.

2.2 FPGA CAD Flow

This section reviews the Computer Aided Design (CAD) steps of implementing a circuit onto an FPGA architecture described in the last section. This is an important part of our modeling methodology, which will be described in Chapter 3.

The process of implementing a circuit onto an FPGA architecture can be divided into five steps, namely: synthesis, technology mapping, packing, placement and routing. The final output is a detailed placement and routing solution. Figure 2.7 shows a flowchart of the typical FPGA CAD flow. The following subsections will describe the algorithms that are typically used in each step of the CAD flow.

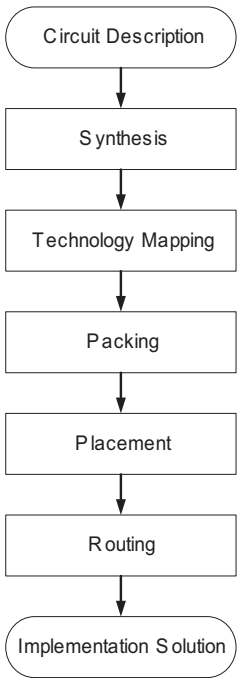


Figure 2.7: FPGA CAD flow

2.2.1 Synthesis and Technology Mapping

In the synthesis step of the CAD flow, a circuit is translated from a hardware description language (HDL) (such as VHDL or Verilog), into a gate-level representation. The gate-level representation consists of a network of Boolean logic gates and flip-flops. Often logic optimiza-

tions, independent of the implementation technology, is performed at this step.

Once the circuit is in a network of boolean gates, technology mapping is performed on it to convert it into a network of library gates specific to the implementation technology. For FPGAs these are often K-input lookup tables (LUT).

There are a number of FPGA synthesis and technology mapping algorithms [6, 18, 20] optimizing for a number of objectives including delay, area and power. The SIS/Flowmap/FlowPack [19] algorithms are the most widely used for synthesis and technology mapping to FPGAs. The Flowmap algorithm is able to find a logical-delay-optimal (depth-optimal) solution in polynomial time, and FlowPack further improves the solution through area reduction.

For this work, the SIS/Flowmap/FlowPack tools are used for the synthesis and technology mapping of circuits into netlists of K-input LUTs and flip-flops.

2.2.2 Packing

After synthesis and technology mapping the circuit must be transformed into a netlist of logic blocks (clusters) by grouping together related LUTs and flip-flops. This step is known as packing.

Packing algorithms can be categorized into three general approaches, namely top-down ([30, 32]), depth-optimal ([22, 46]) and bottom-up ([25, 43]). In a top-down approach a circuit is recursively partitioned into fixed size clusters. Depth-optimal solutions minimizes circuit delay at the expense of area by duplicating logic. Bottom-up approaches are most suitable for FPGAs because they have fast run times and yield good circuit performance.

An example bottom-up packing approach is the VPack [9] clustering algorithm. VPack builds clusters sequentially one at a time. It first perform a grouping of LUTs and flip-flops into BLEs, then group related BLEs into logic blocks/clusters. The first step involves a simple pattern matching. The second step, the clustering, starts by selecting a seed BLE for a potential cluster and grows the cluster by greedily adding other BLEs that are attracted to it. The attraction function is based on the number of shared nets between BLEs. The cluster is complete when the number of BLEs in it equals the cluster size N of the FPGA architecture, or when other constraints such as input usage arise. This procedure is repeated until all BLEs are

clustered.

For this work, VPack is used for the packing of circuits into netlist of logic blocks, which can then be placed in the FPGA as described in the next subsection.

2.2.3 Placement

The placement step in the FPGA CAD flow places the netlist of logic blocks (clusters) of the circuit on to the fixed locations in the FPGA. An example placement problem is depicted in Figure 2.8 where one must place two LUTs into 3×3 array.

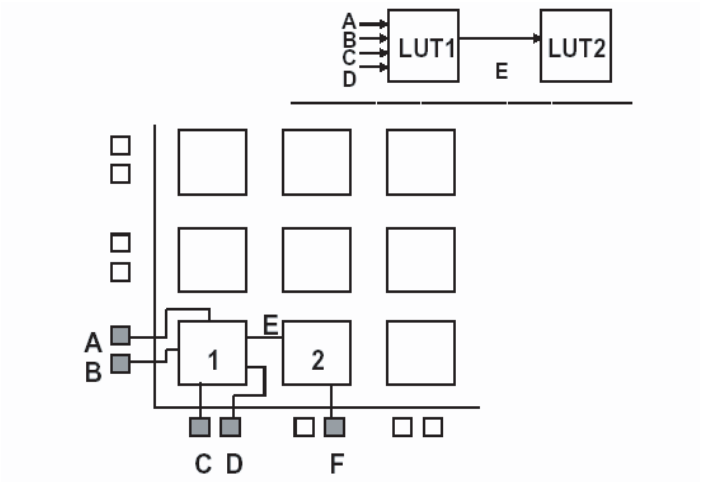


Figure 2.8: An example placement problem

The FPGA placement problem takes as input a netlist of logic blocks and outputs their positions in the FPGA, optimized for three metrics. These three metrics are: total wirelength, routability (balance wiring density across the FPGA), and performance. There are three classes of placement approaches: min-cut partition ([31]), analytical ([12, 26, 34]), and simulated annealing ([9, 54]). Simulated annealing is the most common approach since it is easy to add new optimization goals.

The Simulated Annealing algorithm starts with a random placement, and then performs a number of pair-wise random logic block swaps. At any placement stage the quality of the placement is measured by a cost function (which is to be minimized). The algorithm always accepts a swap that reduces the cost function and accepts with a probability a swap that

increases the cost function (a bad swap). The choice to accept some bad swaps, allows simulated annealing to escape cost function local minima; this is called "hill-climbing". The probability of accepting a bad swap is $e^{-\Delta C/T}$, where ΔC is the amount increase in cost function, and T is temperature. The temperature is set to be high at the beginning of the algorithm to accept nearly all swaps. Gradually temperature decreases and the probability of accepting a cost-increasing swap decreases. At low temperatures only cost-reducing swaps are accepted. The algorithm ends when a stopping criterion is met. In research, the most commonly used simulated annealing placer for FPGAs is the VPR placer [9], which we use in this work.

2.2.4 Routing

The final stage in the FPGA CAD flow is routing. It is the step which connects the placed logic blocks using routing wires and routing switches. The routing problem takes as inputs the placement solution with a netlist of circuit connections, and a detailed routing architecture. The output is a detailed mapping solution of nets to pins and wires and which routing switches should be turned on.

Routers can be classified into two categories: global-detailed two-step routers ([13, 37]) and combined single-step routers ([9, 45]). Global-detailed routers first assign nets to channels in a first global routing step then assign nets to specific pins and wires in a second detailed routing step. Global-detailed routers are not suitable for FPGAs because the limited flexibility of FPGA routing architecture makes the second step difficult under global routing constraints determined in the first step, and it has been shown that a single-step router is superior [9].

The combined single-step router searches for a final routing solution directly. The most widely used single-step router in research is the VPR router [9]. It models any arbitrary routing architecture by representing it by a routing resource graph, a directed acyclic graph in which nodes are routing resources (pins and wires) and directed edges are routing switches. An example routing resource graph is depicted in Figure 2.9. The VPR router's ability to model any routing architecture is suitable for this work, as we wish to model a large space of routing architectures.

With a complete routing resource graph, the VPR router routes the circuit's nets on the

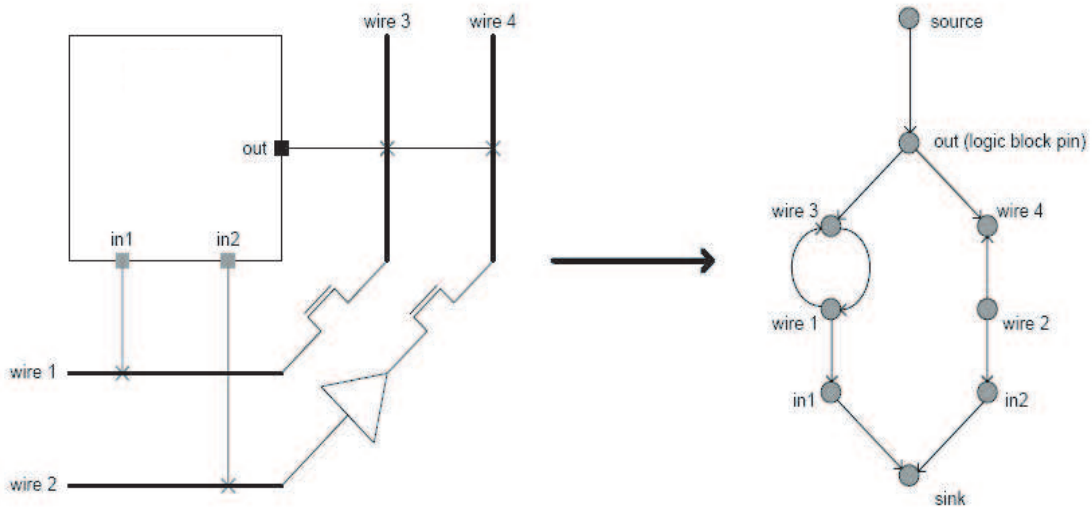


Figure 2.9: Modeling of FPGA routing architecture as a routing resource graph [9]

routing resource graph using an advanced Pathfinder [45] routing algorithm. It is an iterative routing algorithm which allows nets to use the same routing resources, causing congestion. At the end of a iteration if there is congestion, all nets are ripped up and a new iteration is performed. However, in successive routing iterations the cost of using routing resources that were overused in previous iterations is penalized. The penalty gradually increases with the number of iterations and increases dramatically for routing resources overused by many nets. This approach ensures important nets acquire routing resources that were overused in a previous iteration, while less important nets move off and use other routing resources resolving congestion.

The "importance" of nets is captured in the VPR router by a cost function that can optimize for routability or performance. The routability-driven VPR router uses a cost function that minimizes wirelength by encouraging each net to use the fewest number of routing resources possible (taking minimum path). In timing-driven mode VPR router uses a cost function that encourages timing critical nets to use fastest routing resources. In this work, VPR's router is used under routability-driven mode.

2.3 Previous Research on Interconnect Models

In this section we will review general interconnect models proposed for an ASIC implementation medium. These models compute the expected wirelength distribution and then predict the routing demand/wire spacing using the wirelength distribution. It is important to review these ASIC-centric models because FPGA specific interconnect models are built on top of them, including some aspects of our model.

2.3.1 Rent's Rule

One of the most widely explored interconnect model is Rent's Rule. Rent's Rule pertains to the organization of circuits, specifically the relationship between the number of external signal connections to a partition of logic.

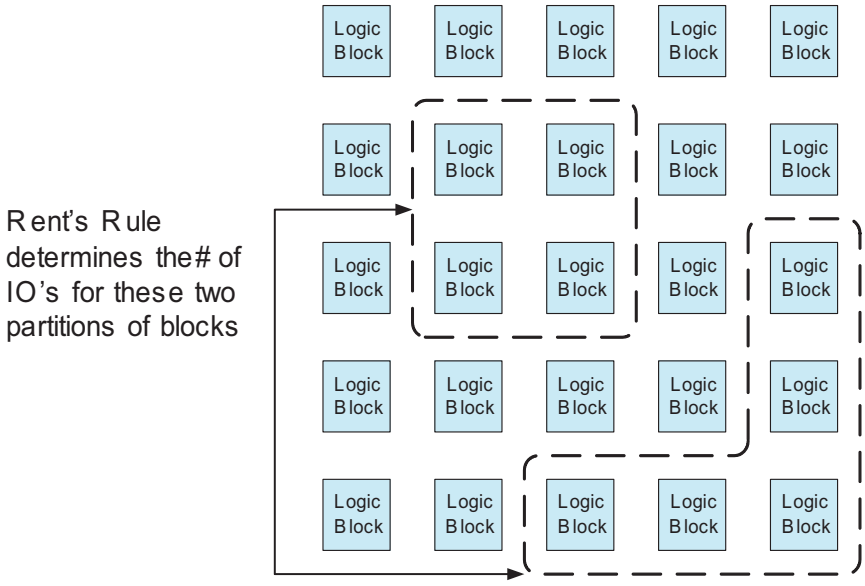


Figure 2.10: Visual interpretation of Rent's Rule

Rent's Rule was first reported in [36] as an empirical observation on that relationship. It states that for a partition the number of IO connections external to the partition is proportional to a power of the number of logic blocks of the partition. A visual interpretation is depicted in Figure 2.10. This power-law relationship is expressed by Equation 2.3.1.1

$$T = t \cdot C^p \tag{2.3.1.1}$$

where T is the number of terminals/external IO connections of the partition, C is the logic block count of the partition, t, known as *Rent constant*, is the average number of terminals on a single basic logic block, and p is the *Rent exponent* (a constant between 0 and 1). Typical values for p have been observed in the range from 0.5 for a highly serialized structure to 0.75 for a highly parallel structure.

The Rent exponent reflects circuit topology as well as the amount of optimization achieved in placement. To more precisely define that notion, Hagen et al. in [29] introduced the intrinsic Rent exponent, p^*, to characterize optimal circuit placement and measure the interconnect complexity of a circuit. A higher intrinsic Rent exponent value corresponds to higher topological complexity. The intrinsic Rent exponent is a lower bound on Rent exponent $p^* < p$, since it assumes optimal placement.

There are many applications of Rent's Rule. The primary application is in estimating wirelength distributions of integrated circuit chips. The earliest such applications are classic wirelength distribution models by Donath and Feuer independently, briefly described in the next subsection.

2.3.2 Donath's and Feuer's Models on Wirelength Distribution

Using Rent's Rule, Donath in [23] derived a model for an upper bound on the average 2-pin wirelength of a circuit/package, referred to as $\overline{R}$. He found that the upper bound of average wirelength $\overline{R}$ is a function of the circuit size C and the Rent exponent p, as

$$\begin{aligned} \overline{R} &\propto C^{p-\frac{1}{2}} && \text{for } p > \tfrac{1}{2} \\ \overline{R} &\propto logC && \text{for } p = \tfrac{1}{2} \\ \overline{R} &\propto F(p) && \text{for } p < \tfrac{1}{2} \end{aligned} \tag{2.3.2.1}$$

where $F(p)$ is a function of p but independent of C.

This model only predicts a theoretical upper bound on a single average. Later, Donath extended his findings into a model for wirelength distribution in [24], which can be used to predict wiring space/routing demand in a uniform gate array routing. He found the distribution

of wirelengths to be defined by Equation 2.3.2.2

$$\begin{aligned} f(r) &= gr^{2p-3} \qquad \text{for } 1 \leq r \leq L \\ f(r) &= 0 \qquad\qquad\quad \text{for } r > L \end{aligned} \tag{2.3.2.2}$$

where $f(r)$ is the fraction of nets with wirelength r, L is a constant related to the size of the array, and g is a normalization constant.

Donath's derivation is based on a devised model for the placement process. He assumed a two-dimensional placement model based on recursive partitioning that has the smallest bisection net crossing (minimum cut). This imposes a hierarchical structure on the placement.

An alternative wirelength distribution model that does not impose a recursive/hierarchical partitioning placement assumption is Feuer's model. Feuer in [27] states that for a good placement the Rent relationship will hold, on average, for any geometric region/partition of the circuit. Furthermore, the distribution of wirelengths is

$$f(r) \propto r^{2p-4} \tag{2.3.2.3}$$

and the average wirelength in a partition of C logic blocks is

$$\overline{R} \propto C^{p-\frac{1}{2}} \tag{2.3.2.4}$$

both results for a Rent exponent $p > \frac{1}{2}$. Feuer's results agree with those of Donath in that the wirelength distribution is a power function, and Feuer extends Donath's model for any arbitrary partition of the circuit. Also Feuer formulated an exact expression for the average wirelength in Equation 2.3.2.5.

$$\begin{aligned} \alpha &= 2 - 2p \\ \overline{R} &= \sqrt{2}\frac{(2-\alpha)(5-\alpha)}{(3-\alpha)(4-\alpha)}C^{p-\frac{1}{2}} \end{aligned} \tag{2.3.2.5}$$

Both Feuer and Donath's wirelength distribution models, built from Rent's Rule, can be applied to estimate the wiring space/routing demands of a gate array. One such work that does this is El Gamal's stochastic interconnect model for gate arrays.

2.3.3 El Gamal's Stochastic Interconnect Model

In [28], El Gamal developed a stochastic model to predict the wiring requirements of Master Slice Gate Arrays that have a two-dimensional array of identical logic blocks, with horizontal and vertical routing channels between the rows and columns of logic blocks. The model divides the channels into channel segments that span the length or width of one logic block. For such an architecture, which is similar to an island-style FPGA, El Gamal's model predicts the amount of wiring required in a channel segment, referred to as the channel density. An example of the channel density calculation is given in Figure 2.11.

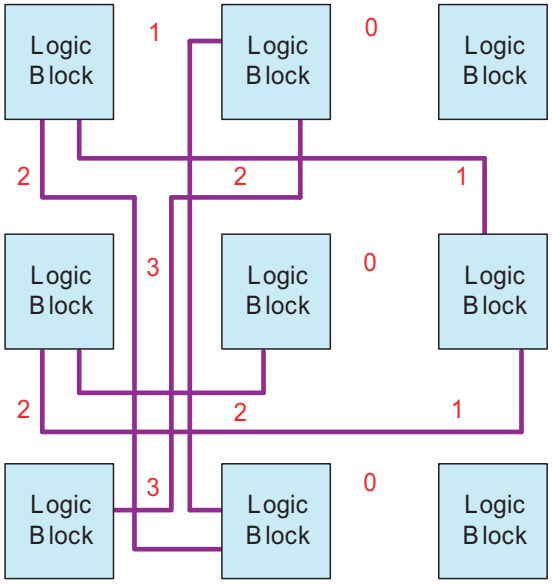


Figure 2.11: An example 2-D Master Slice routing with channel densities labeled

The model has the following assumptions. First it assumes that logic blocks can be represented by points in a two-dimensional grid. It is also assumed that all routings are independent two terminal connections originating stochastically at one logic block, and traveling a minimum distance (selected at random according to a distribution) through the channel segments to another logic block. This means routes can turn at any point and be of any length as in ASIC routing or routing in a fully flexible FPGA. It is further assumed that the number of connections per logic block can be drawn independently from a Poisson distribution with parameter λ, where λ is defined as the average number of used input pins per logic block. The average

wirelength per connection, in number of logic blocks traversed, is assumed to be given as $\overline{R}$. It can be estimated using either Donath or Feuer's wirelength distribution models described in Section 2.3.2. Finally, it is assumed that the trajectory of the routes follows a non-reversing random walk.

Under the above assumptions, in an array that has M × M routing channels, El Gamal found that the channel densities will be Poisson distributed, with the average channel density W_{avg} given by

$$W_{avg} = \frac{\lambda \overline{R}}{2} \tag{2.3.3.1}$$

This result holds for any distribution (not just Poisson) as long as $\overline{R}$ is finite and independent of M.

Although El Gamal's model was developed for Master Slice circuits, some of its results can also be applied to the island-style FPGAs we consider. This is true because both types of devices are based on a two-dimensional array of identical logic blocks/cells. The key assumption when using El Gamal's results for island-style FPGAs is that the routing architecture must consist of unit length wire segments (spanning only one logic block) and have the ability to turn at any point (very flexible routing architecture). A simple derivation illustrates the validity of El Gamal's interconnect model when applied to FPGAs.

Suppose there is an $M \times N$ array of identical logic blocks, each with 2 channel segments adjacent – one vertical and one horizontal. The total number of two-terminal connections is λMN, and since each is on average $\overline{R}$ in length, the total expected routing wirelength would be $\lambda \overline{R} MN$. Each channel segment contains wire segments of unit length and total number of channel segments is $2MN$, therefore the average number of unit length wires in a channel segment – average channel width – is

$$W_{avg} = \frac{\text{total wirelength}}{\text{number of channel segments}} = \frac{\lambda \overline{R} MN}{2MN} = \frac{\lambda \overline{R}}{2} \tag{2.3.3.2}$$

Later we will make use of El Gamal's result on average channel density to formulate an interconnect model for fully flexible FPGAs whose wire segments are all unit length.

2.3.4 Song's Channel Density Estimation

While El Gamal was one of the first to consider the problem of estimating routing channel densities for two-dimensional gate array integrated circuits, his model abstracted arrays of logic blocks as lattice points in a two-dimensional grid, which can not be used to describe nets from distinct boundaries of a logic block. To improve routing estimation precision, a similar stochastic model that considered block boundaries was derived in [57].

In [57] Song et al. extended El Gamal's channel density/wire space estimation to take into account geometries of identical logic blocks. They consider routes not emanating from a lattice point representing a logic block, but a boundary of a logic block with four sides. Some assumptions of El Gamal were also made in [57] namely, all nets are two-terminal connections, routes are made with minimum distance, trajectory of the routes follows a non-reversing random walk, and the number of connections per logic block can be drawn independently from a Poisson distribution with parameter λ.

Also similar to El Gamal's model, Song's model considers a gate array with ASIC-style routing. All routes can be of any length and turn at any point. There is no inflexibility in the routing choices, unlike the case of an FPGA routing architecture. However, similar to El Gamal's model, Song's model could be applied to FPGA interconnect prediction with empirical calibration to incorporate routing architecture inflexibility information.

The result of Song's model is that the channel density equals the sum of three independently drawn Poisson distributions X(L), X'(L) and X(LR) with means according to Equation 2.3.4.1, where each distribution represents the three types of wires shown in Figure 2.12 contributing to channel density.

$$
\begin{aligned}
E[X(L)] &= \frac{\lambda}{4} \\
E[X'(L)] &= \frac{\lambda}{4} \\
E[X(LR)] &= 2\lambda \sum_{r=1}^{\infty} \sum_{s=1}^{\infty} \binom{r+s}{r} 4 \sum_{j=1}^{\infty} q_{r+s+j} \sum_{k=0}^{j} \binom{j}{k} \Big/ \binom{r+s+j}{r+k} \\
&\quad + \lambda \sum_{r=1}^{\infty} 4 \sum_{j=1}^{\infty} q_{r+j} \sum_{k=0}^{j} \left(\binom{j}{k} \Big/ \binom{r+j}{r+k} + \binom{j}{k} \Big/ \binom{r+j}{k} \right)
\end{aligned}
\tag{2.3.4.1}
$$

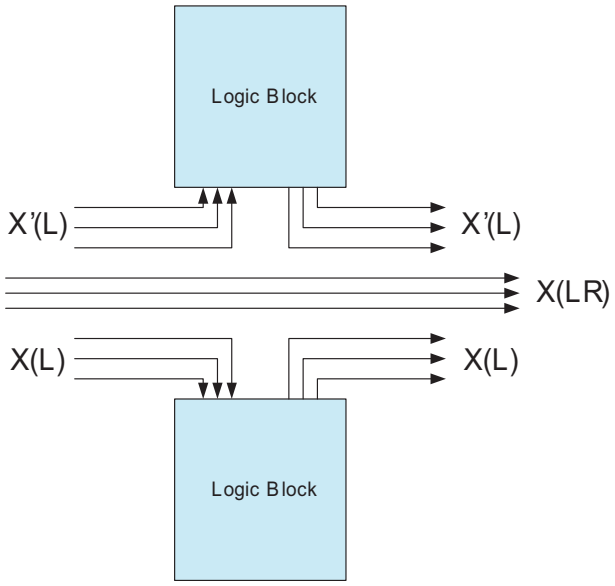


Figure 2.12: Model of channel density as three types of routing wires in [57]

where q_l is the probability a net terminates at a border of a logic block l distance away and has the property

$$\sum_{l=1}^{\infty} 8lq_l = 1 \tag{2.3.4.2}$$

and its distribution is determined by the wirelength distribution, which was first modeled by Donath and Feuer in Section 2.3.2. One could also use a modern wirelength distribution model such as that of Davis [21] overviewed in the next subsection.

2.3.5 Davis' Wirelength Distribution Model

Similar to Donath's and Feuer's wirelength distribution models, the more recent wirelength distribution model of Davis in [21] is based on Rent's Rule. Davis made a number of extensions to the classic models including, deriving a continuous distribution function of wirelength, extending to multi-terminal net model to account for multiple sink nets, and deriving a complete closed form analytic expression for wirelength distribution.

Davis starts out considering a point-to-point star-connected routing model of nets as shown in Figure 2.14(a). He considers the three partitions of a gate array of identical logic blocks as in Figure 2.13, where A is the *source partition* (always containing a single logic block), B is

the *buffer partition* and C is the *wavefront partition* (all logic blocks a distance r away from source partition). He formulates the expected number of IO connections connecting A and C by applying a conservation of terminals law and Rent's Rule. Since the wavefront partition C is by design a distance r away from source partition A, Davis has derived the number of routing wires of length r. This gives a complete stochastic wirelength distribution for nets connecting logic block 1.

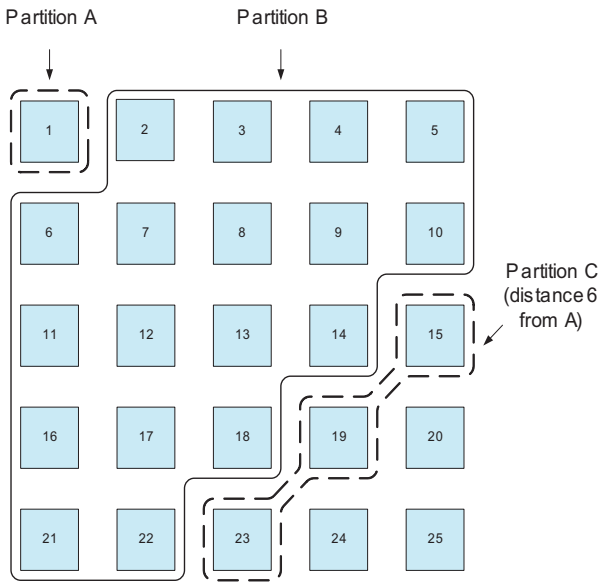


Figure 2.13: Davis model of determining wirelength distribution for a single logic block

Once the stochastic wirelength distribution for logic block 1 is determined, Davis "removes" logic block 1 from the array for calculating the remainder of the wiring distribution to avoid double counting. He designates logic block 2 as the new source partition A and perform the same calculation of wirelength distribution. He repeats the same process for all other logic blocks in the system and superimposes wirelength distributions for individual logic blocks to obtain the wirelength distribution for the entire system.

Davis then adjusts the model to match closer to real routing topology by a linear net model depicted in Figure 2.14(c).

The resulting analytical derivation for wirelength distribution for an M $\times$ M gate array is

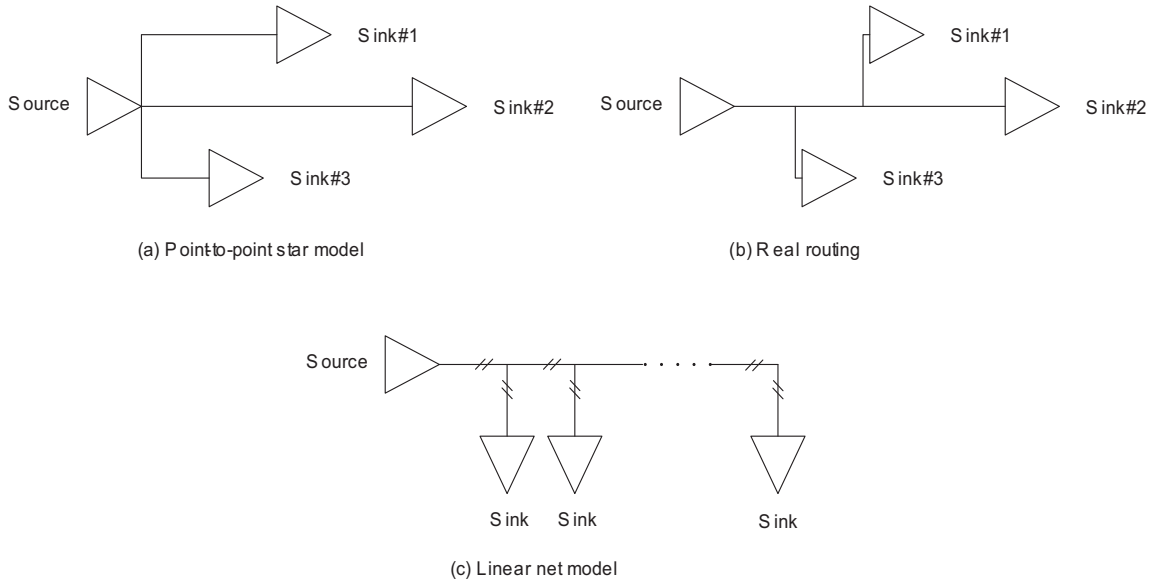


Figure 2.14: Davis models routing of a multi-sink net by (c) linear net model

$$\begin{aligned} \alpha &= \frac{f.o.}{f.o.+1} \\ f(r) &= \frac{\alpha t}{2}\Gamma(\frac{r^3}{3} - 2Mr^2 + 2M^2 r)r^{2p-4} \qquad \text{for } 1 \le r < M \\ f(r) &= \frac{\alpha t}{6}\Gamma(2M-r)^3 r^{2p-4} \qquad \text{for } M \le r < 2M \end{aligned} \tag{2.3.5.1}$$

where $f(r)$ is the fraction of nets with wirelength r, $f.o.$ is the average fanout, and t and p are Rent's Rule parameters according to Equation 2.3.1.1, repeated here

$$T = t \cdot C^p \tag{2.3.5.2}$$

and Γ is a constant determined by p and M.

It has been shown in [21] that this wirelength distribution model is more accurate than Donath's model in [24].

Up to now, all interconnect models we have reviewed are theoretic models built on the empirical observation known as Rent's Rule. There are many empirically derived models for predicting interconnect use. One such model we rely on for our work is the post-placement estimation of wirelength in the RISA placer, described in the next subsection.

2.3.6 RISA Wirelength Model

RISA, developed by Cheng [14], is a placement algorithm for standard cells. It uses a routing wirelength model in its cost function. Cheng uses the basic bounding box wirelength model with an enhancement to more accurately predict wirelength. A more general version of this enchancement was developed by Swartz [60]. We will review this enhanced wirelength model since we make use of it in the present work.

The basic bounding box wirelength model predicts that the wirelength needed to route a net is equal to its half-perimeter bounding box length. This is correct for nets with two or three terminals, but for nets with four or more terminals, the half-perimeter bounding box does not account for the extra wire needed to reach all of the terminals.

The RISA wirelength model accounts for the extra wire by scaling the half-perimeter bounding box wirelength of a net by a correction factor. The correction factor compensates for the fact that the half-perimeter bounding box model underestimates wirelength for nets with more than three terminals. Nets with just two or three terminals will have a correction factor of 1.0 as shown in Figure 2.15. The wire length correction factor for a four-terminal net is about 1.08, since extra wire is needed to connect the fourth terminal as shown in Figure 2.15.

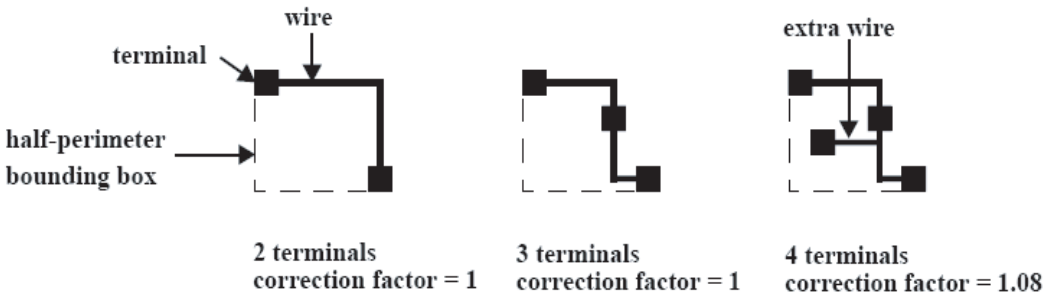


Figure 2.15: Example of correction factors [59]

The correction factor for different fanout nets were empirically determined by creating thousands of randomly distributed net terminals and averaging the correction factor for each of the different fanout nets. Table 2.1 lists the correction factors given in [14] for nets with up to fifty terminals.

Num. Terminals	Correction Factor	Num. Terminals	Correction Factor
$2 \sim 3$	1.00	15	1.69
4	1.08	20	1.89
5	1.15	25	2.07
6	1.22	30	2.23
7	1.28	35	2.39
8	1.34	40	2.54
9	1.40	45	2.66
10	1.45	50	2.79

Table 2.1: RISA correction factors for nets with up to fifty terminals [14]

For higher fanout nets Swartz [60] determined the correction factors for up to 3000 terminals by the same empirical approach. They extended RISA's wirelength model for nets with terminals greater than 50 using Equations 2.3.6.1.

$$\begin{aligned} C(k) &= 2.6 \times 10^{-2} \cdot k + 1.49 \qquad \text{for } 50 < \text{k} < 85 \\ C(k) &= -1.8 \times 10^{-6} \cdot k^2 + 1.1 \times 10^{-2} \cdot k + 2.79 \qquad \text{for k} \geq 85 \end{aligned} \tag{2.3.6.1}$$

We have found that RISA's wirelength model and its extension by Swartz are good at predicting wirelength after placement.

2.4 FPGA Specific Interconnect Models

In the previous section we reviewed general interconnect models from classic to contemporary, originally proposed for ASIC routing. While these models are not directly applicable to FPGA interconnect, many research efforts have extended them for FPGAs. In this section we will review some of such FPGA interconnect models.

Predicted Difficulty	Condition
Unroutable	$W_{need} > W_{FPGA} + 0.5$
Easily Routable	$W_{need} < W_{FPGA} - 0.5$
Marginally Routable	$W_{FPGA} - 0.5 \leq W_{need} \leq W_{FPGA} + 0.5$

Table 2.2: Routability prediction classification

2.4.1 Chan's FPGA Routing Difficulty Prediction

One popular FPGA interconnect model is the FPGA routing difficulty prediction model by Chan et al. Chan et al [11] developed a model to predict the difficulty of successfully routing a technology mapped circuit, before placement of the circuit. To predict whether or not a circuit will route successfully in a given FPGA, the model estimates the amount of routing resources needed by the circuit. If the target FPGA has more routing resources than needed by the circuit, then the circuit is considered routable.

Chan first estimates the minimum channel width (W_{need}) required by a circuit for successful routing using interconnect models of El Gamal [28] and Sastry and Parker [52]. Both of these models require the average number of pins per logic block, λ, and the average wirelength, $\overline{R}$. The parameter λ can be found after technology mapping. The average wirelength $\overline{R}$ is estimated using Feuer's wirelength distribution model in [27].

Having an estimate of the required channel width, W_{need}, and knowing the target FPGA channel width, W_{FPGA}, the difficulty of routing a circuit is predicted. If the circuit requires more channel width than available in the target FPGA, then the circuit is *unroutable*. If the circuit requires less channel width than available in the target FPGA, then the circuit is *easily routable*. The difficulty classification is susceptible to errors in the estimation of required channel width, W_{need}, particularly when the estimate is close to the available channel width, W_{FPGA}. For this reason Chan uses a margin of error of ± 0.5. Therefore when W_{need} lies within ± 0.5 of W_{FPGA}, the circuit is considered *marginally routable*, meaning that it is unknown whether the circuit is routable. Table 2.2 lists the three classifications and their conditions.

Chan's FPGA interconnect model serves as a predictor of whether a circuit is likely to fit or not in a fixed width FPGA. Its categorical modeling is not sufficient for our purposes, which is to

predict the required channel width for any circuit and FPGA architecture. However, we mirror Chan's approach to estimating the required channel width for a fixed FPGA architecture: we use El Gamal's model for average channel width to determine the required channel width for a fully flexible FPGA architecture. Instead of using an analytic model for wirelength computation, such as that of Feuer as did Chan, we use the empirical model of RISA.

2.4.2 Rahman's FPGA Wiring Requirement Model

In [49], Rahman et al presented an analytical model for FPGA interconnect. It predicts the channel width requirement of a design in FPGA before place and route steps in the CAD flow. It is a model suitable for providing early analysis and feedback for design trade-offs without experimentation.

Rahman's model takes as input an estimate of the circuit's Rent constant t and exponent p and its size in number of logic blocks C. It uses Davis's wirelength distribution model in [21] to estimate the total required wirelength. Rahman estimates the required channel width W_{need} for successful routing by equating the available total wirelength to the estimate total required wirelength, in Equation 2.4.2.1.

$$W_{need} = \frac{\sum_{r=1}^{2\sqrt{C}-2} r \cdot f(r,t,p)\chi_{fpga} + r_l}{2C \cdot e_t} \tag{2.4.2.1}$$

where $f(r,t,p)$ is the fraction of nets with wirelength r for designs with Rent parameters t and p, and r_l is the additional wirelengths from using longer than unit wire segments. They measure typical utilization e_t of FPGA wires and additional FPGA point-to-point to multi-sink net model conversion factor χ_{fpga} by place and route experiments using place and route tool SEGA [37] and VPR [9]. The parameter values are calibrated to each place and route algorithm. It is important to note that they ran experiments only on FPGA routing architectures consisting of $Fs = 3$, and $Fc_{in} = Fc_{out} = W$. However, they do model FPGA routing architectures consisting of various wire segment lengths by adjusting Davis's wirelength distribution model and r_l parameter.

For validation they ran experiments using various benchmark circuits, but predict assuming the same Rent parameters $t = K + 1$ (for K-input lookup table there are K input and 1 output

connection), and $p = 0.75$. They found acceptable prediction results and showed their model to be better than El Gamal's average channel density model when predicting required channel width.

Their other important contribution is in extending their two-dimensional analytical model for three-dimensional FPGAs. They argued that three-dimensional integration can significantly reduce total wirelength (higher dimensionality gives more flexibility), as a result reduce channel width requirement. They supported their argument with their three-dimensional model and validation experiments. Further validation on this model was later performed by the same research group in [35].

In sum, Rahman's model provides an estimate of FPGA interconnect using Davis's wirelength distribution model, which in turn requires Rent characterization of the circuit (thus requiring a placement to measure Rent parameters or an estimate of Rent parameters). Rahman's model, while able to account for different wire segment lengths, must be tuned to each routing architecture parameter combination (Fs, Fc_{in}, and Fc_{out} etc.). We like their modeling approach of equating required wirelength and available wirelength for one routing architecture, so we approach similarly in modeling the fully flexible FPGA. We differ, however, in the choice of circuit characterization. We use El Gamal's characterization of logic block and circuits λ and $\overline{R}$ instead of Rent parameters.

2.4.3 Kannan's FPGA Interconnect Estimation: fGREP

A more tool-based FPGA interconnect estimation model than previous models is fGREP [33] (**f**ast **G**eneric **R**outability **E**stimation for **P**laced FPGA circuits) by Kannan et al. This model takes in a placed circuit netlist and a routing resource graph, and predicts the required channel width by estimating channel segment occupancies. In essence it maps out the routing demand of the circuit placement on a per channel segment basis.

The key aspect in Kannan's model is the notion of *demand* by a net on a routing resource. If P is the set of all routing choices for a net, and $P_i \in P$ are those routes that use the routing resource v_i, then the demand D^i on v_i exerted by the net is defined as

$$D^i = \frac{|P_i|}{|P|} \tag{2.4.3.1}$$

It is clear that D^i can also be interpreted as the probability v_i is needed, since it's value ranges from 0 to 1. For example, when $D^i = 1$ for a net then routing resource v_i must exist and be used by that net. Every net exerts a demand on a routing resource in the FPGA. When one sums these individual demands, the final value is the probability-weighted count of number of routing resources needed. When the routing graph is a global routing resource graph, where each routing resource represents a channel segment, then the demand sum is the number of wire segments needed in each channel segment. An example of the demand exerted by a net on global routing resources is shown in Figure 2.16.

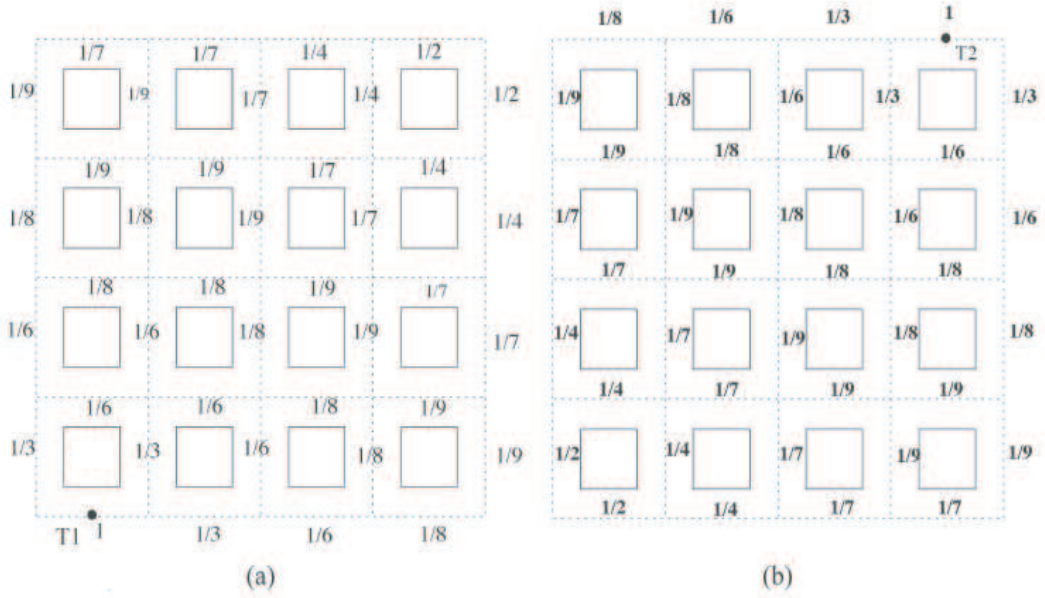


Figure 2.16: Example of demands exerted by (a) terminal T1. (b) terminal T2. [33]

The key step in their estimation is the computation of demands, which requires an enumeration of possible routing paths for nets. They perform this step using a breadth-first traversal on a routing resource graph, beginning at each net terminal specified by the circuit placement.

It is claimed in [33] that their model can estimate for any detailed routing architecture as long as the input routing resource graph captures it. However, their experimentation only

proved the validity of a single routing architecture that has $Fc_{in} = Fc_{out} = 1.0$. Their results showed that their model predicts channel width requirements within 4% over their benchmark circuits.

Kannan's model is applicable to a large space of routing architectures. This is a feature that we are interested in for our model. However, Kannan requires the routing architecture be specified in a routing resource graph. This is inconvenient for an FPGA architect since it takes a large file (on the order of 100's MB) to describe a routing resource graph. As an alternative we opt for an parametrization of routing architectures with input being a number of easy to understand parameter values defined in Section 2.1.2. An FPGA interconnect model that uses a similar parametrization of routing architecture is Brown's Stochastic Model for FPGAs described in the next section.

2.4.4 Brown's Stochastic Model for FPGAs

In [10], Brown developed a stochastic model for FPGA routing. It is in the form of a series of stochastic equations, that takes in parametrization of any of a large space of routing architectures, and circuit and logic block characterization, and outputs the probability that a net can successfully route. What sets this model apart from others is that it can model FPGA interconnect for a large space of routing architectures.

Similar to Chan's model in Section 2.4.1, Brown uses El Gamal's result on channel densities. Brown further assumes, as did El Gamal, that the number of connections per logic block is Poisson distributed. By El Gamal's derivations, this assumptions leads to the result that the channel densities are Poisson distributed also. Knowing channel densities as a Poisson distribution, Brown is able to compute the likelihood that a circuit will route in a fixed channel width architecture, where each channel segment has W wiring tracks.

In more detail, Brown takes as input circuit and logic block characterization parameters λ and $\overline{R}$ (Section 2.3.3), and routing architecture flexibility parameters Fs, Fc_{in} (fixing other parameters at $L = 1$ and $Fc_{out} = Fc_{in}$) and fixed channel width W. So although the model only captures unit length architectures, it is powerful in capturing various routing flexibilities and architectures.

With the inputs supplied, Brown begins by computing the probability $P(R_{C_i})$ a connection C_i can successfully route in the specified routing architecture by Equation 2.4.4.1.

$$P(R_{C_i}) = \sum_{r=1}^{max} P(R_{C_i}|L = r)P(L = r) \tag{2.4.4.1}$$

where $P(R_{C_i}|L = r)$ is the conditional probability of successful routing and $P(L = r)$ is the probability that connection C_i is of length r. The Brown adopted a geometric distribution for $P(L = r)$, but any one of Donath, Feuer and Davis's wirelength distribution model can also be applied (the distribution function $f(r)$ in Sections 2.3.2 and 2.3.5 is the same as $P(L = r)$).

The conditional probability of successful routing is modeled by Brown as a sequence of events of successful connections of routing resources expressed in Equation 2.4.4.2. The physical interpretation of Equation 2.4.4.2 can be seen in Figure 2.17.

$$\begin{aligned} P(R_{C_i}|L = r) &= P(X_1 \bigcap S_1 \bigcap ... \bigcap S_{r-1} \bigcap X_2) \\ &= P(X_1)P(S_1|X_1)P(S_2|S_1)...P(X2|S_{r-1}) \end{aligned} \tag{2.4.4.2}$$

where X_1 S_i and X_2 are events of successfully connecting to the first connection block (output pin to wire), successfully connecting two adjacent wires and successfully connecting to the final connection block (wire to input pin), respectively.

Figure 2.17: Brown's stochastic model

The conditional probability of each event can be computed by combinatoric arguments given channel densities from El Gamal's result.

Once the probability $P(R_{C_i})$ of successfully routing a connection C_i is computed, Brown repeats the calculations for the next connection C_{i+1} with updates to the channel densities to

reflect the fact that the first connection has been successfully routed. This process is repeated for all connections and the final output is *routability*, the average probability of successfully routing a connection over all, as given in Equation 2.4.4.3.

$$routability = \frac{1}{C_T} \sum_{i=1}^{C_T} P(R_{C_i}) \tag{2.4.4.3}$$

Brown's model is the FPGA interconnect model that most resembles our model in its inputs. It has the ability to model for a large space of FPGA routing architectures, with various switching flexibility. Its circuit and logic block characterizations are the same as ours, which were originally proposed in El Gamal's model. However, the difference lies in that Brown's model requires complex stochastic equations, but our model uses simple analytic equations to convey intuition on FPGA architecture.

2.5 Summary

In this chapter, we have introduced basic FPGA architecture terminology, including the logic block architecture and routing architecture. We looked at the CAD flow of implementing a circuit onto an FPGA architecture. We also overviewed general ASIC interconnect prediction models that FPGA interconnect models are based on, including aspects of our model. Finally, we described FPGA interconnect models that are comparable to our model and from which we have drawn ideas from.

In the next chapter, we describe in detail the modeling approach and experimental setup.

Chapter 3

Modeling Methodology

This chapter describes the methodology by which we develop the FPGA interconnect model for early architecture development. We begin by a formal definition of the inputs and outputs of this model in terms of parameters introduced in Chapter 2. Then we establish the context in which the model is intended to be used. Next we discuss the approach we take to develop the model, and how to assess its quality. Finally, we describe the experimental methodology our approach relies on.

3.1 Problem Definition

The goal of this research is to develop a simple and intuitive interconnect model for island-style FPGAs, for use in early architecture development stages. The model should take as inputs:

1. Characterization of logic block architecture in terms of its routing demand, and some circuit information in: λ and $\overline{R}$.

2. Routing architecture parameters: Fs, Fc_{in}, Fc_{out}, L and Eqv.

The parameters λ and $\overline{R}$, from El Gamal's interconnect model (in Section 2.3.3) characterize the routing demand of the logic block and circuit placed in it. The routing architecture parameters (defined in Section 2.1.2) model the flexibility of the routing architecture.

The single output of the model is the estimated routing demand/track count required for successful routing, which we denote W_{need}. So the completed model should look as:

$$\begin{aligned} W_{need} &= f(\text{logic block architecture and circuit information}, \text{routing architecture}) \\ &= f(\lambda, \overline{R}, Fs, Fc_{in}, Fc_{out}, L, Eqv) \end{aligned} \tag{3.1.0.1}$$

We seek to develop this model for a balance of **simplicity**, **intuition** and **accuracy**. The goal of simplicity and giving intuition are important in guiding an architect in the early architecture development. Accuracy is important because the model should be accurately enough to be useful.

3.2 Context of Model Application

The FPGA interconnect model developed in this work is intended for use in early stages of architecture development, to guide an FPGA architect in the absence of an empirical method that requires tools and circuits. Instead of resorting to an empirical method, the architect can employ this model for quick preliminary feedback on architectural ideas. To do so, the architect needs to:

1. Estimate values for λ and $\overline{R}$ for each logic block architecture under consideration

2. Provide values for the routing architecture parameters Fs, Fc_{in}, Fc_{out}, L, and Eqv.

The routing architecture parameters are easy to choose (and the literature gives a variety of suggestions), but the architect needs a method to estimate values λ and $\overline{R}$, which characterizes the logic block architecture and circuit. Furthermore, in the interest of developing a widely useful architecture the characterization needs to be for a set of circuits in a target market, and not simply one circuit. In Chapter 5, we discuss how this might be done and provide a model for picking λ and $\overline{R}$ for cluster-based logic block architectures. With all the inputs determined the model can be used without any circuits or experiments.

While our model is to be used without any specific circuit or experiment, its development requires actual benchmark circuits and experiments. While some derivations from first principles are used, a majority of the model development process is based on trends observed in

experimental data obtained from synthesizing benchmark circuits onto architectures being modeled. Therefore during the model development, we measure λ and $\overline{R}$ for each benchmark circuit individually, to provide accurate input parameter values. The empirical modeling methodology is described in the following sections.

3.3 Modeling Approach

Our modeling approach will be detailed in this section. We will first discuss the high-level strategy for developing the model and then go into details. Next we review the accuracy metrics and methods used to compare candidate models.

3.3.1 Incremental Modeling Strategy

The goal of this FPGA interconnect model is to provide intuition for architecture development in the early stages, with reasonable accuracy. For that purpose, the model should inform an architect the impact of each input parameter, as well as altogether. To gain such transparency in intuition, a suitable approach is incrementally creating the model, constructing intermediate models for a subset of architecture parameters.

To incrementally create the model we start (in Chapter 4) by creating a model for a fully flexible FPGA, in which all routing parameters have maximum values as given in Table 3.1. Since it is implicit that all routing parameters are at their maximum value the model for a fully flexible FPGA can omit the routing parameters, and simply be a function of logic block architecture and circuit characterization parameters as in Equation 3.3.1.1. This is a practice we adopt throughout: whenever a routing parameter is fixed at its maximum value it does not appear in the model equation, since its effect is not being modeled.

Once the fully flexible FPGA model is complete, we incrementally generalize it for each

Fs	Fc_{in}	Fc_{out}	L	Eqv
$3W$	W	W	1	1

Table 3.1: Fully flexible FPGA routing architecture

routing parameter at a time. We first generalize it for switch block flexibility Fs, to create an model of form in Equation 3.3.1.2. Keep in mind that since Equation 3.3.1.2 is a generalization of fully flexible FPGA model in Equation 3.3.1.1, we ensure that when Fs is at its maximum value Equation 3.3.1.2 reduces to (either exactly or approximately) Equation 3.3.1.1. We next generalize switch block flexibility model for input connection block flexibility Fc_{in} to create Equation 3.3.1.3 and so on. We follow this procedure until the model is generalized to include all parameters. The intermediate models give intuition on the impact of each parameter individually.

$$W_{need} = f(\lambda, \overline{R}) \quad \text{fully flexible FPGA} \tag{3.3.1.1}$$

$$W_{need} = f(\lambda, \overline{R}, Fs) \quad \text{introduce loss of flexibility in parameter } Fs \tag{3.3.1.2}$$

$$W_{need} = f(\lambda, \overline{R}, Fs, Fc_{in}) \quad \text{introduce loss of flexibility in parameter } Fc_{in} \tag{3.3.1.3}$$

We recognize that this incremental approach is less optimal compared to an approach where all parameters are allowed to vary together in one fitting step. However, the latter approach would be infeasible due to the large routing architecture space we wish to model. Our incremental approach mirrors that of routing architecture design optimization by Betz et al. in [9], where they perform a series of experiments in which they optimize for one architectural parameter, while fixing the other parameters. This approach is commonly adopted for optimization/modeling of a large design space such as in this work. Furthermore, the value of incremental approach is to gain transparency and intuition on routing architecture, an important goal of this work, as aforementioned.

Alternatively, one could adopt a different incremental approach: instead of generalizing the model starting from maximum routing flexibility, one could start from minimum (or any choice of) routing flexibility. We chose to start from maximum because it gives intuition on "what is the effect of a loss in flexibility", and it allows simpler analytic expressions.

This is our high-level model development strategy. It will be reflected in the presentation of modeling results in Chapter 4. In the next subsection we will go into the details of modeling methodology.

3.3.2 Detailed Modeling Methodology

The detailed construction of the model is a combination of derivations and guided empirical modeling. A summary of the methodology is presented as a flowchart in Figure 3.1. Whenever possible we derive analytic expressions that relate input parameters with the output. If no derivable expression or the derived expression is not sufficiently accurate for predicting the output, we use a guided empirical modeling process.

In the guided empirical modeling process, we begin by generating empirical data, and perform analysis on them to discover more trends that are important for intuition and model forms. From the trends, we formulate candidate models.

The proposed candidate models are fitted to the initially generated data. This set of data is the **training set**. Fitting/Training is performed using the most common method, least squares fitting, which minimizes the sum of the squares of the difference between the model and actual experimental data. To do this we use the $lsqcurvefit$ function in MATLAB optimization toolbox, which solves the nonlinear least-squares-fitting problem for any user-defined continuous equation. It is based on the interior-reflective Newton method described in [15, 16]. It finds values of fitting parameters that yields the best model fit.

We keep the number of fitting parameters low, 1 or 2 per intermediate model, to avoid overfitting. Overfitting is the problem where one has introduced too many fitting parameters (relative to the size of training set) to create a false model, which fits the training set well but predicts poorly for unseen future data points. The rule of thumb is the size of training set should be greater than 10 times the number of fitting parameters. We follow this rule of thumb, as our training set sizes range from 100's to 10000's for 1 or 2 fitting parameters. To check for overfitting we perform cross validation as suggested in [2, 56].

After candidate models are trained, for the purpose of cross validation, we generate another set of empirical data called the **validation set**, of the same number of data points as the training set. To check for overfitting, we examine the trained model's accuracy on validation data. If the model's accuracy on the validation data is significantly worse than its accuracy in fitting to the training data, then overfitting has occurred [56]. We check for this condition throughout the modeling process. The benchmark circuits corresponding to the two sets are

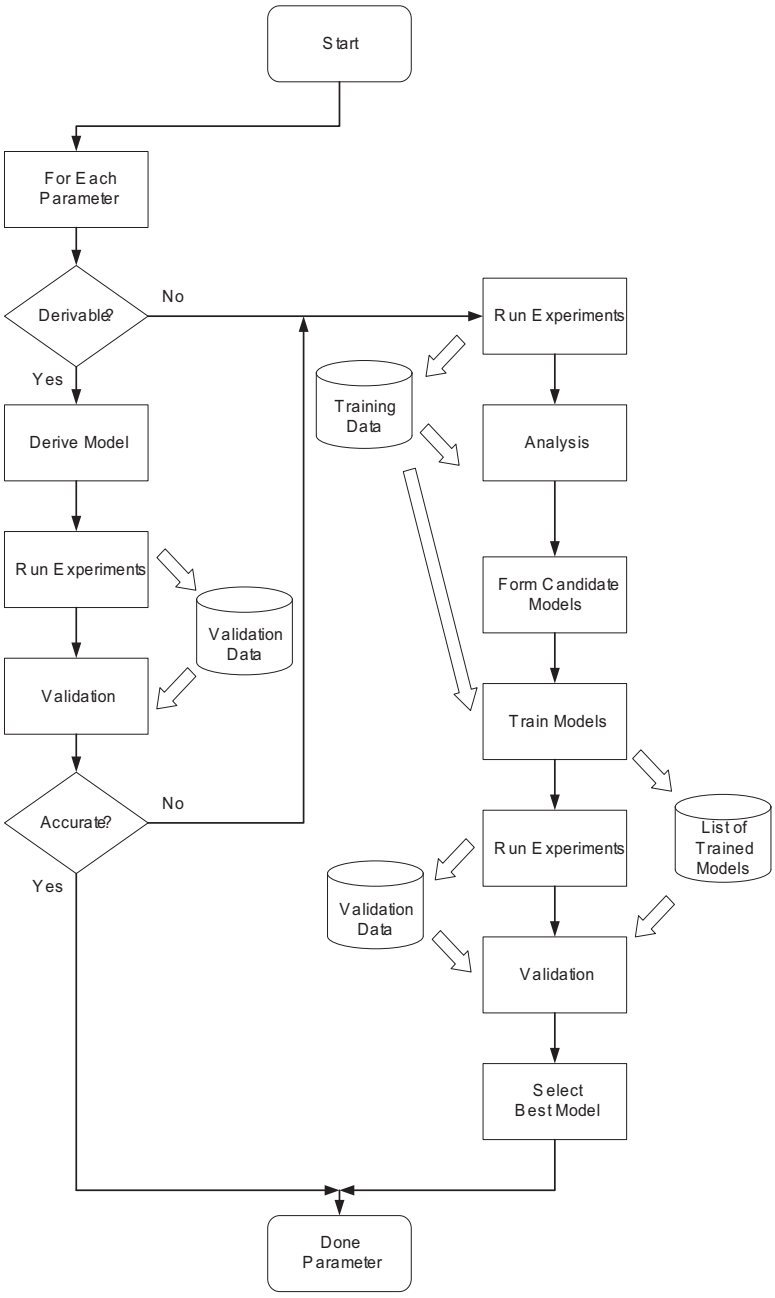


Figure 3.1: Detailed modeling methodology flowchart

listed in Table 3.2, where the 4-input basic logic element (BLE) counts of circuits are also listed, in the even columns.

The benchmark circuits come from the largest 20 MCNC circuits [66], and 8 additional benchmark circuits gathered from various open benchmark sources including: The Opencores

Training Set		Validation Set	
Circuit	# 4-Input BLEs	Circuit	# 4-Input BLEs
clma	8383	des_perf	12032
s38584.1	6447	s38417	6406
rs_decoder_2	4502	pdc	4575
mac1	3718	diffeq_paj_convert	3792
elliptic	3604	spla	3690
frisc	3556	des_area	2025
fir_scu_rtl	2201	apex2	1878
s298	1931	seq	1750
rs_decoder_1	1745	alu4	1522
diffeq	1497	apex4	1262
misex3	1397	ex5p	1064
tseng	1047	apex3	869
cps	757	apex1	700
misex3c	549	parker1986	663

Table 3.2: Benchmark circuit list and sizes in 4-Input BLEs

organization [48], SCU-RTL [53], and Texas-97 [61]. The latter includes various applications such as FIR and RS decoder design.

The method by which the training and validation sets are chosen is described in a later subsection, Section 3.3.4. Once validation is performed on the trained models, we can objectively evaluate and compare the candidate models using an accuracy metric. We also subjectively compare the candidate models for simplicity and the intuition they present. We select the final model based on a combination of subjective and objective evaluations.

The accuracy metric for the objective evaluation is discussed in the following subsection.

3.3.3 Accuracy Metric

We use the most commonly used metric for comparing model accuracy – Root Mean Squared Error (RMSE) – defined as in Equation 3.3.3.1.

$$RMSE = \sqrt{\frac{1}{C} \sum_{\text{all C circuits}} (Predicted - Measured)^2} \quad (3.3.3.1)$$

Root Mean Squared Error is a suitable choice because it is the metric whose value is minimized during the fitting process (recall least squares fitting minimizes the sum of squared error). Furthermore it is appropriate because RMSE heavily penalizes large error (due to squaring). This is important because in guiding an architect in determining routing demand it is better to be always off by a small amount, than to occasionally be off by a large error. Also, this metric is the most commonly used in regression modeling and analysis.

To measure the accuracy of a candidate model, we compare its prediction to experimentally measured data for each circuit (synthesized into the modeled architecture) individually, then average across all circuits for the validation set. Candidate models are compared based on their RMSE value in validation.

The reason for choosing validation RMSE (versus training RMSE) for quantitative comparison of candidate models is because validation is the true measure of a candidate model's predictive ability. However, we report both training RMSE (the goodness of the fit to data) and validation RMSE in throughout this work for completeness.

In addition to reporting RMSE we also report accuracy by a more human-comprehensible metric – Mean Absolute Percentage Error (MAPE) – defined as in Equation 3.3.3.2.

$$MAPE = \frac{1}{C} \sum_{\text{all C circuits}} \frac{|Predicted - Measured|}{Measured} \times 100\% \quad (3.3.3.2)$$

The MAPE metric is not used for quantitative comparison of models, but it varies in unison with the RMSE metric; they are correlated, particularly since all our data are positive values. The sole purpose of reporting MAPE is to give an intuitive understanding of error size.

3.3.4 Training and Validation Sets

In this subsection we review the method of assigning benchmark circuits to the training and validation sets. They are of equal size and are chosen from our total set of benchmark circuits. The reason for choosing equal size for the two sets is to reduce statistical noise; if one set was

	Average			Standard Deviation		
	Training Set	Validation Set	Ratio	Training Set	Validation Set	Ratio
Number of BLEs	2952	3016	0.98	2275	3098	0.73
Number of Nets	4004	4113	0.97	3072	3463	0.89
λ	12.0	12.7	0.95	1.26	1.49	0.85
$\overline{R}$	4.56	4.67	0.98	0.88	0.79	1.12

Table 3.3: Statistical similarity of training and validation sets

smaller then it would be sensitive to statistical noise. To further reduce statistical noise we carefully choose the two sets.

The training and validation sets are chosen by a random separation of the original set of benchmark circuits, with the constraint that the statistical properties of each set are similar to that of the combined set as proposed in [58]. This can be interpreted as that the training and validation sets should be statistically similar. We use BLE count and the number of two-terminal nets (net model given in Section 3.4) as metrics for comparing statistical properties of each set. Thus we measure the average and standard deviation of both metrics for both training and validation sets. Among a number of attempts of random separations, we pick one that yields the most similar average and standard deviation. This resulted in the assignment of training and validation sets in Table 3.2. The degree to which these two sets are statistically similar is summarized in Table 3.3. In Table 3.3, the first column contains statistical metrics for similarity test. The next three columns contain the metric average for training and validation sets and ratio of the averages, respectively. The final three columns is the same except it is for standard deviation. Note the inclusion of measured λ and $\overline{R}$ of each circuit and determination of their average and standard deviation in the final two rows; this is done to further show similarity between the two sets. They are measured by the experimental method described in the next section.

3.4 Experimental Methodology

In the guided empirical modeling process described in Section 3.3.2 we need to generate empirical data for analysis and fitting. For this purpose we use the CAD flow described in Section 2.2 to run experiments, synthesizing circuits into modeled architectures.

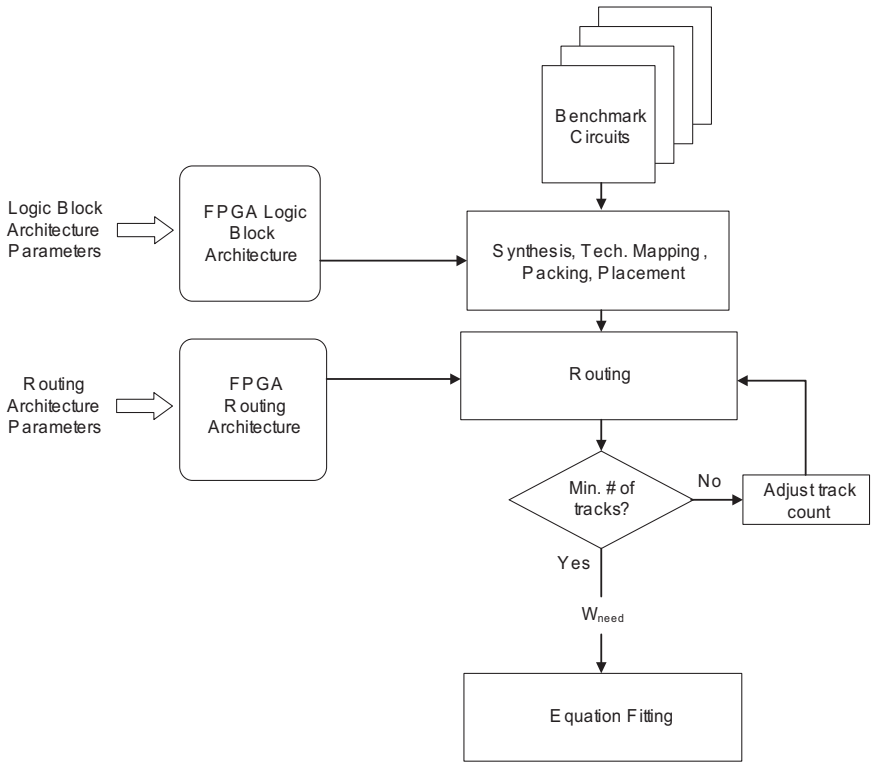


Figure 3.2: Experimental flow

The experimental flow is depicted in Figure 3.2. First, benchmark circuits are synthesized into lookup tables (LUT) and registers using the FlowMap and FlowPack tools in [18, 19]. Then the registers and LUTs are packed into logic clusters using the packing algorithm VPACK [9] followed by wireability-driven placement by VPR placer [9]. Each circuit is placed in the smallest square FPGA it can fit which is the common practice in FPGA architecture study. Finally, the placed circuit is routed using a modified version of VPR router [9], in routability-driven mode. Modifications were made to support single-driver routing architectures, with some effort to make good quality routing switch connectivity patterns. During routing we determine the minimum required channel width W_{need} to successfully route each circuit by iteratively

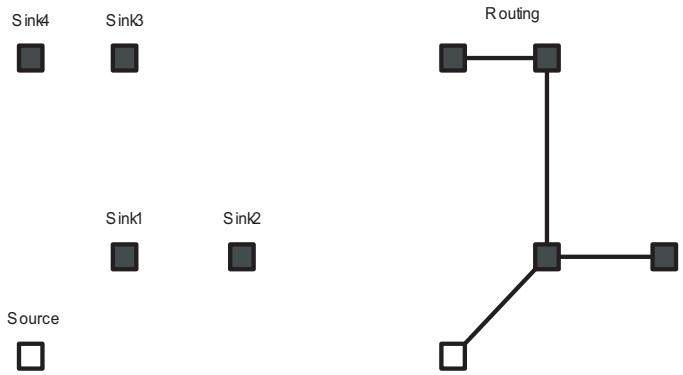


Figure 3.3: The minimum spanning tree net model

routing each circuit, reducing channel width of the routing architecture until it fails to route.

The remaining two input parameters λ and $\overline{R}$ are also measured during this flow. We measure λ using the circuit netlist after the packing step, by counting the total number of used pins on logic blocks. To measure $\overline{R}$ we need to assume a two-terminal net model.

We assumed the Minimum Spanning Tree (MST) two-terminal net model, depicted in Figure 3.3, since we have found it to be very close to the actual routing from experiments for the fully flexible FPGA architecture. We determine $\overline{R}$ by dividing the total expected wirelength, measured after placement using the wirelength estimation extension of RISA by Swartz (described in Section 2.3.6), by the total number of connections assuming the MST net model. In the experiment tool VPR, we also use the MST net model as the basis for the directed-search router, as it was found to give high quality and speed router in [60].

The base FPGA architecture based on which we run experiments and generate the training and validation data is summarized in Table 3.4. It specifies the logic block architecture in rows 2 to 4, and it specifies the routing architectures in rows 6 to 11, in terms of routing parameters. Experiments on additional architectures are run for further validation, and will be described in Chapter 5.

Logic Block Architecture	
LUT Size (k)	4
Cluster Size (N)	10
Cluster Input Pins (I)	22
Routing Architecture	
Fs	[3,21]
Fc_{in}	[2,W]
Fc_{out}	[2,W]
L	{1,2,4,6,8}
Eqv	{0,1}

Table 3.4: Base FPGA architecture for experiments

3.5 Summary

In this chapter, we have described a formal definition of the inputs and outputs of our FPGA interconnect model. We established the context in which the model is intended to be used. Then we discussed the approach we take to develop the model. The approach is incremental and based a combination of derivations and empirical modeling. We reviewed how to evaluate our model's accuracy and how to choose training and validations sets for empirical modeling. Finally, we reviewed the experimental methodology our approach relies on.

In the next chapter we walk through the model development process, detailing how the models are developed, the insights they give and their accuracy.

Chapter 4

Development of Routing Demand Model

In this chapter, we develop the FPGA interconnect model. As mentioned in Chapter 3, the model will be incrementally developed, with intermediate models of increasingly complex routing architectures. This development strategy is reflected in the structure of this chapter. We first overview the global structure of the model. We start the development by constructing model for a fully flexible FPGA. Then we incrementally generalize it to account for different routing architecture parameters, by introducing loss of flexibility accounted for in each parameter. This includes: switch block flexibility parameter (Section 4.3), input/output connection block flexibility parameter (Section 4.4), wire segment length (Section 4.5), and finally logical equivalence of pins (Section 4.6). We conclude by presenting the complete routing demand model.

4.1 Global Structure of Model

The goal of the model is to predict the number of tracks per channel, also known as channel width, needed to route a circuit in a given logic block and routing architecture. The channel width needed, W_{need}, can be viewed at the highest level as consisting of three components as:

$$W_{need} = \begin{matrix} absolute \\ minimum \end{matrix} + \begin{matrix} switching\ matrix \\ flexibility \\ penalty \end{matrix} + \begin{matrix} segment \\ length \\ penalty \end{matrix} \quad (4.1.0.1)$$

The **absolute minimum** is the portion of required channel width attributable to the fundamental logic block and circuit demand. One can view it as requirement of circuit in a given logic block, without consideration for the routing inflexibility (like in an ASIC routing architecture). The latter two terms are penalty that channel width requirement must increase by if inflexibility is introduced to the routing architecture, in order to successfully route. The term **switching matrix flexibility penalty** is the penalty associated with the loss of switches, compared to a fully flexible FPGA, in the switch blocks and input/output connection blocks. The term **segment length penalty** is the penalty associated with wire segments being greater than unit length.

4.2 Model for the Fully Flexible FPGA

We begin with a model for predicting routing demand of a fully flexible FPGA. A fully flexible FPGA has maximum connectivity between all adjacent routing resources (wires and pins). Effectively it has: a crossbar in the switch block, as well as input and output connection blocks; the smallest length wires possible; and logical equivalence between all input pins and between all output pins. An example of such architecture is shown in Figure 4.1.

The routing architecture parameter values of the fully flexible architecture is given in Table 4.1. Routing in a fully flexible FPGA is similar to ASIC routing, where routes can turn almost anywhere and have any length. Note that ASIC routing is even more flexible, such as two pins on the same side of a logic block can share the same track in an ASIC but not in a fully flexible

Fs	Fc_{in}	Fc_{out}	L	Eqv
$3W$	W	W	1	1

Table 4.1: Fully flexible FPGA routing architecture

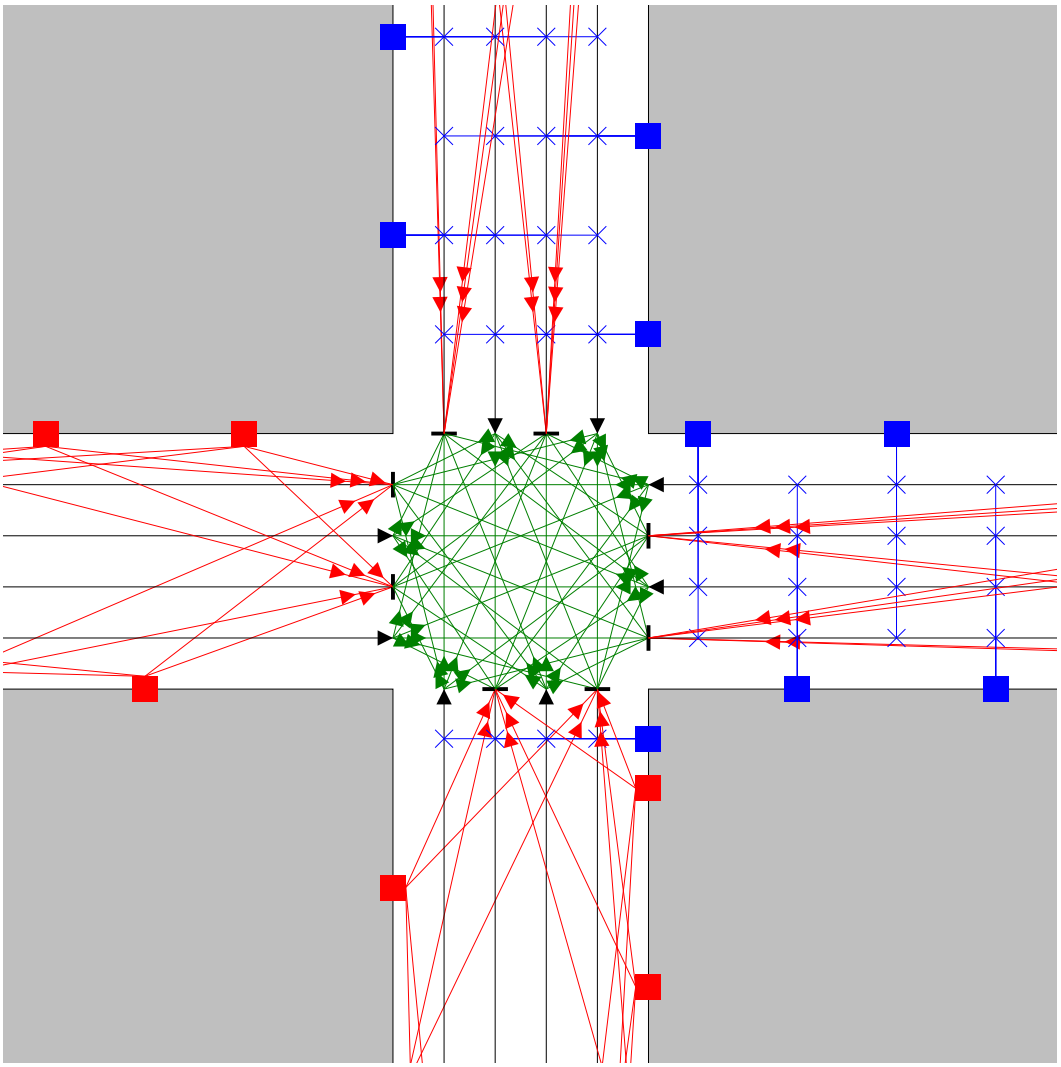


Figure 4.1: An example fully flexible FPGA architecture

FPGA as defined here. Nonetheless, we can draw upon a classic ASIC interconnect model for modeling routing demand of a fully flexible FPGA. We will use a key result from El Gamal's interconnect model described in Section 2.3.3, which is that the average channel width, W_{avg}, in a gate array with ASIC/fully flexible FPGA routing is given by

$$W_{avg} = \frac{\lambda \overline{R}}{2} \tag{4.2.0.2}$$

where λ is defined as the average number of used input pins per logic block, and $\overline{R}$ is the average wirelength per two-terminal connection, in number of logic blocks traversed [28].

The parameters λ and $\overline{R}$ will be used as inputs to our model. They characterize the routing demand of the logic block and circuit. During early stage development, the model should take as inputs λ and $\overline{R}$ representing not just one circuit but a domain/market of circuits mapped to a choice of logic block. In Chapter 5 describing the use of our FPGA interconnect model, we will provide a model for λ and $\overline{R}$, to inform the architect on how to pick a single value for each, representing a domain/market of circuits and choice of logic block, without having to measure by experiments. However, to ensure accuracy of our interconnect model, during each development step in this chapter we measure λ and $\overline{R}$ for each benchmark circuit individually, and train and validate our model with each circuit's λ and $\overline{R}$. We measure λ and $\overline{R}$ using the methodology described in the Chapter 3.

We can apply El Gamal's result on average channel width to derive a model predicting routing demand (maximum channel width) in a fully flexible FPGA.

4.2.1 Maximum versus Average Channel Width

Equation 4.2.0.2 gives the average channel width, but we are interested in estimating the *maximum* channel width across all channel segments, when the circuit is routed with the fewest number of tracks per channel, W_{need}. An expression for the maximum channel width, W_{need}, is derivable by observing that FPGA routing channels have an average *utilization* [60], U, (the fraction of wires that are actually used) given by

$$U = \frac{W_{avg}}{W_{need}} \tag{4.2.1.1}$$

Re-arranging Equation 4.2.1.1 to solve for W_{need} and substitute in Equation 4.2.0.2 for W_{avg} gives

$$W_{need} = \frac{1}{U} \cdot \frac{\lambda \overline{R}}{2} \tag{4.2.1.2}$$

We will employ the inverse of the utilization term U, which we call the *peak factor*, ($p = \frac{1}{U}$). We will re-write Equation 4.2.1.2 and refer to the channel width requirement of a fully flexible FPGA by a special term, called W_{abs_min}, given as

$$W_{abs\_min} = p\frac{\lambda \overline{R}}{2} \tag{4.2.1.3}$$

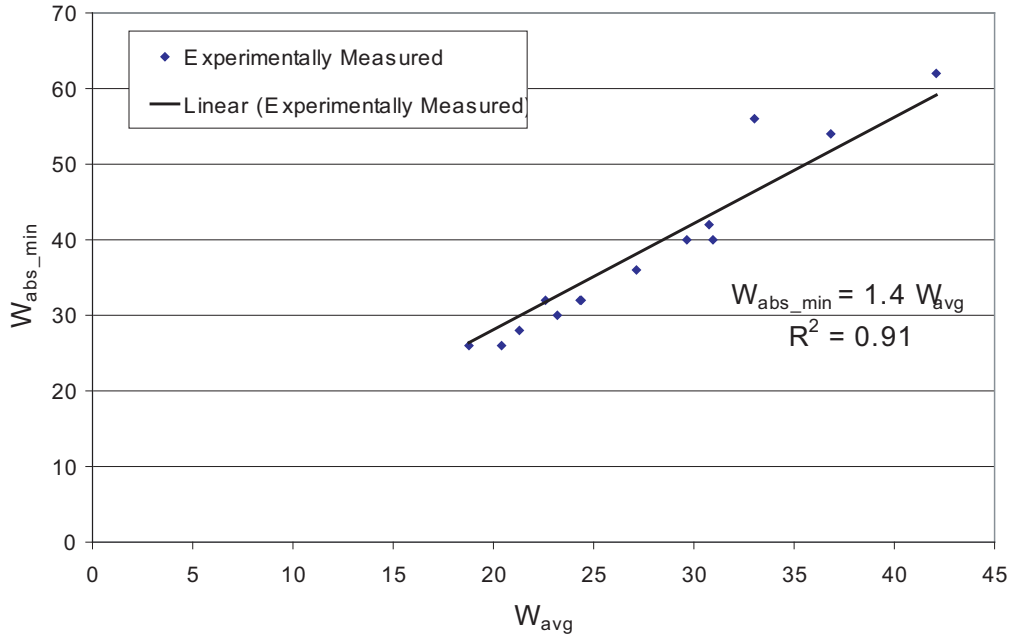


Figure 4.2: Experimental trend of W_{abs_min} using training set benchmark circuits

There is experimental support for the proportionality expressed in Equation 4.2.1.3. We ran experiments using the flow described in Section 3.4, on training set benchmark circuits under the fully flexible architecture indicated in Table 4.1. The experimental data is presented in Figure 4.2. Figure 4.2 is a plot of W_{abs_min} (y-axis) for each circuit ordered by each circuit's W_{avg} (x-axis). The linearity of the plot and correlation coefficient $R^2 = 0.91$ of W_{abs_min} with W_{avg} shows that the utilization U of all circuits are consistent under a fully flexible FPGA, suggesting modeling W_{abs_min} as a proportionality of W_{avg} as in Equation 4.2.1.3.

4.2.2 Model Accuracy and Intuition

The model for the fully flexible FPGA is determined by finding the peak factor in Equation 4.2.1.3, by fitting the model to experimental data of the training set. We will report the accuracy of the model by comparing its prediction to actual measured W_{need} for each individual circuit (here λ and $\overline{R}$ are also individually measured), and summarize by an average across all circuits. The best fit value of p is 1.4, for a Mean Absolute Percentage Error (MAPE) of 6.2% for the

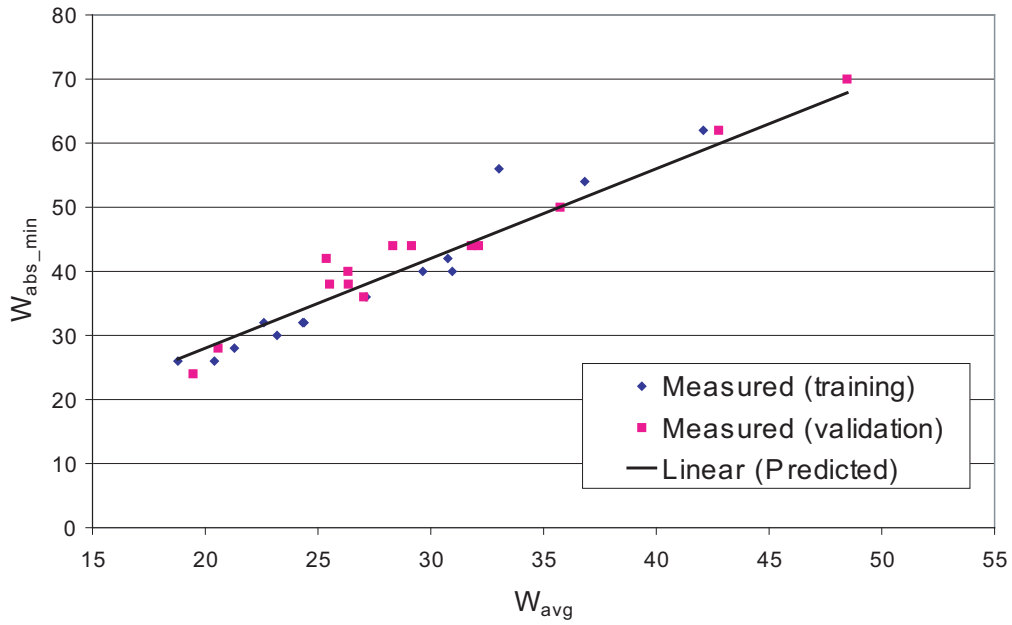


Figure 4.3: Accuracy of fully flexible FPGA interconnect model for all benchmark circuits

training set (14 data points) and 5.8% for the validation set (14 data points). The Root Mean Squared Error (RMSE) is 3.3 for training benchmark circuits and 2.8 for validation benchmark circuits. The accuracy of the model is shown in Figure 4.3, where W_{abs_min} (y-axis) is plotted for all circuits ordered by their W_{avg} (x-axis), and the straight line is our model (predicted values). The breakdown of the accuracy for training and validation benchmark circuits are shown in Table 4.2 and 4.3 respectively, where the columns 2 to 4 are measured W_{abs_min}, predicted W_{abs_min}, predicted − measured and error as percentage of the measured, respectively. The last row gives the absolute error percentage averaged across all circuits in the set.

This model of the fully flexible FPGA, captures the *intrinsic routing demand* of the logic block architecture and circuit mapped to it – intrinsic meaning independent of the routing architecture.

W_{abs_min} is the *absolute minimum* channel width needed to route a circuit. If the circuit is routed on an FPGA that is less than fully flexible, then it will require more channel width than W_{abs_min}. The following sections explore models for this reduction in flexibility.

Benchmark	Measured	Predicted	Error	Error %
clma	62	59	-3	-5.0%
elliptic	56	46	-10	-17%
frisc	54	52	-2	-4.5%
s298	42	43	1	2.5%
mac1	40	42	2	3.8%
misex3	40	43	3	8.3%
s38584.1	36	38	2	5.5%
rs_decoder_2	32	32	0	-1.1%
cps	32	34	2	6.4%
fir_scu_rtl	32	34	2	6.7%
tseng	30	32	2	8.2%
diffeq	28	30	2	6.5%
rs_decoder_1	26	26	0	1.1%
misex3c	26	29	3	9.9%
Average Absolute Error				**6.2%**

Table 4.2: Accuracy breakdown of fully flexible model on W_{abs_min} of training circuits

Benchmark	Measured	Predicted	Error	Error %
pdc	70	68	-2	-3.1%
spla	62	60	-2	-3.4%
apex2	50	50	0	0.1%
ex5p	44	40	-4	-9.9%
apex4	44	41	-3	-7.3%
seq	44	45	1	1.2%
alu4	44	45	1	2.2%
des_perf	42	36	-6	-15%
apex3	40	37	-3	-7.9%
apex1	38	36	-2	-6.0%
des_area	38	37	-1	-3.0%
s38417	36	38	2	5.1%
parker1986	28	29	1	2.9%
diffeq_paj_convert	24	27	3	14%
Average Absolute Error				**5.8%**

Table 4.3: Accuracy breakdown of fully flexible model on W_{abs_min} of validation circuits

4.3 Generalizing Routing Model for Switch Block Flexibility

The first choice for reduction in flexibility is the switch block. In the previous section, in a fully flexible routing architecture, the *switch block flexibility* Fs was set at the maximum value, $Fs = 3W$. For the routing architecture modeled in this section, all other routing architecture parameters are kept at maximum except Fs, which is reduced. We have found that the range of interest for Fs is [3,21], for two reasons. First, $Fs = 3$ is considered the minimum because it gives the wire only one chance to turn in each direction at every switch block. Although previous studies in [62] have shown $Fs = 2$ to be a routable architecture, it is commonly accepted that $Fs = 3$ is a minimum, and we adopt this convention in this work. Second, we have empirically found that $Fs \geq 21$ is empirically equivalent to the fully flexible $Fs = 3W$ (this can be expected since $Fs \geq 21$ is abundant in switch block flexibility). In experiments we choose seven Fs values evenly within 3 to 21 inclusively. In sum, the routing architectures

Fs	Fc_{in}	Fc_{out}	L	Eqv
$[3, 21]$	W	W	1	1

Table 4.4: FPGA routing architectures with reduced switch block flexibility

being modeled in this section are given in Table 4.4.

We model reduced switch block flexibility by analyzing the empirical trends and making hypotheses on model form, in the following subsection.

4.3.1 Empirical Analysis on Effects of Reduced Switch Block Flexibility

In this subsection hypotheses on the relationship between W_{need}, Fs, and other parameters are formed based on empirical evidence. In a later subsection, we create models based on these hypotheses.

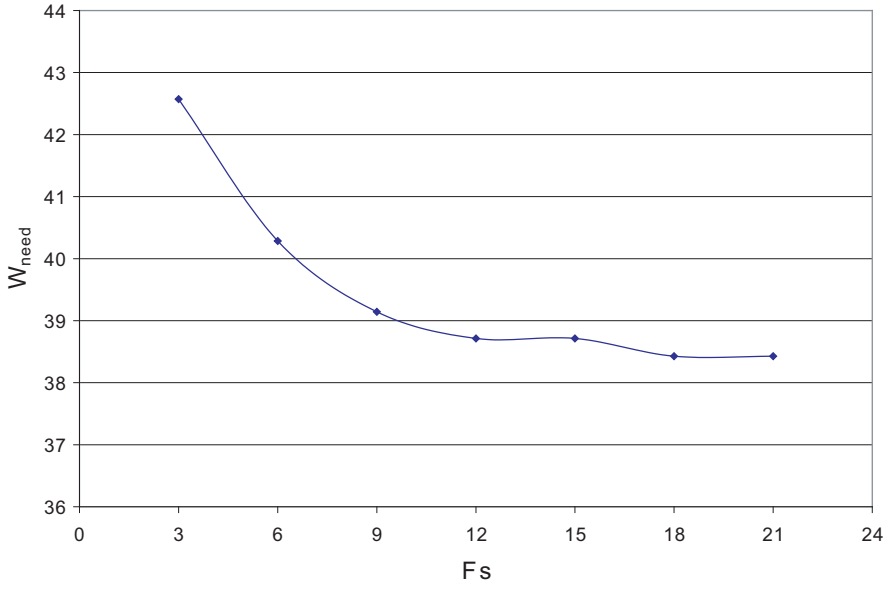


Figure 4.4: Experimental trend of average W_{need} for training benchmark circuits

Firstly, the relationship between W_{need} and Fs is an inverse relationship. To compensate for a loss in Fs flexibility, the required channel width W_{need} must increase, above W_{abs_min}. To see what type of inverse relationship, we ran experiments using the flow determining W_{need} (from Section 3.4) on training circuits, under the architectures given in Table 4.4. The measured

W_{need} (y-axis) averaged across all training circuits are plotted in Figure 4.4, against Fs (x-axis) varied over its range. The figure shows the trend of the average, but individually each circuit displays the same trend. In Figure 4.4, W_{need} tends to W_{abs_min} as Fs becomes large, and W_{need} approaches ∞ as Fs approaches 0. This shows W_{need} is an inverse power function of Fs with an offset W_{abs_min}:

$$(W_{need} - W_{abs\_min}) \propto \frac{1}{Fs^n} \tag{4.3.1.1}$$

where n is a positive real. We choose $n = 1$ for simplicity and will show alternatives later.

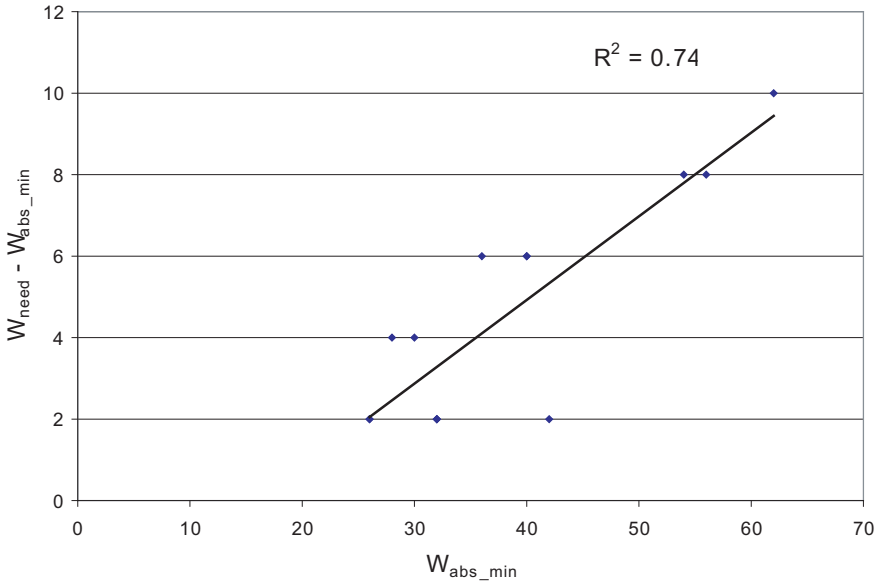
Secondly, the amount of increase in W_{need} (due to decrease in Fs) depends on the circuit's intrinsic routing demand W_{abs_min}. We noticed this trend in the experimental data obtained for the first hypothesis, as we examined the data for each circuit individually.

In Figure 4.5(a) we plot for each training circuit its measured increase in W_{need} (y-axis) at $Fs = 3$ (from W_{abs_min}) against its measured W_{abs_min} (x-axis). The experimental trend shows there is dependence on W_{abs_min}, with a correlation coefficient $R^2 = 0.74$ (some data points overlap). The same trend exists for increase in W_{need} at higher Fs values. One should note that this architecture is so abundantly flexible (all other parameters at maximum except Fs) that the increases in W_{need} are small (2 to 10) relative to the experimental accuracy[1] of ± 2.

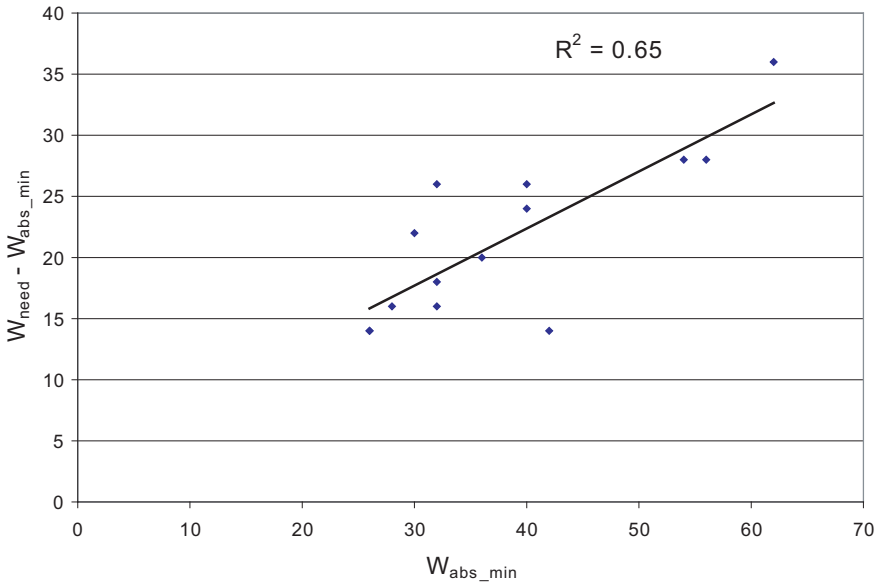
To ensure our experimental conclusion about the dependence on W_{abs_min} is not due to noise, we examine a less flexible architecture. Keeping $Fs = 3$ and all other parameters at maximum value except $Fc_{in} = 2$, with the same experimental flow searching for W_{need} (Section 3.4), we obtain data shown in Figure 4.5(b). The values of increase range from 14 to 36, much higher than the experimental accuracy of ± 2, and there is still dependence on W_{abs_min} with a correlation coefficient $R^2 = 0.65$.

An intuitive explanation for this dependence is that reduction in Fs is applied to every wire. Circuits with more wires to begin with, intrinsic routing demand W_{abs_min}, lose more flexibility as a result of loss in Fs. Hence such circuits would require more wires and higher increase in W_{need} to compensate.

[1] for single-driver routing architectures channel width takes on even values only, so searches for W_{need} are accurate to ± 2



(a) $Fs = 3$



(b) $Fs = 3$, $Fc_{in} = 2$

Figure 4.5: Experimental trend of training circuits shows dependence on W_{abs_min}

While the plots of Figure 4.5(a) and 4.5(b) and the correlation coefficients confirms the dependence of increase in W_{need} on W_{abs_min}, it is not clear in the experimental data what function best models this dependence. Although the correlation coefficients are not 1.0, they are close enough that we choose a linear function to model the dependence for simplicity:

$$(W_{need} - W_{abs\_min}) \propto W_{abs\_min} \tag{4.3.1.2}$$

Using relationship hypotheses 4.3.1.1 and 4.3.1.2 we form the model for reduced switch block flexibility in the following subsection.

4.3.2 Constructing Candidate Models for Fs

In this subsection candidate models of W_{need} as a function of Fs are constructed, using hypotheses formed in the previous subsection. We denote the partial construction of a model by $W^*_{need}(Fs)$ to show the steps.

We first note that the first hypothesis in Relationship 4.3.1.1 says W_{need} should be W_{abs_min} (the absolute minimum), plus a penalty that is inversely proportional to Fs:

$$\begin{aligned} W^*_{need}(Fs) &= W_{abs\_min} + \text{penalty due to } Fs \text{ loss} \\ &= W_{abs\_min} + \frac{1}{\beta}\frac{1}{Fs} \end{aligned} \tag{4.3.2.1}$$

where β is the proportionality constant for Relationship 4.3.1.1. Based on the second hypothesis in Relationship 4.3.1.2, the penalty term must be dependent on W_{abs_min}. So the final model is:

Candidate Model 1:

$$W_{need}(Fs) = W_{abs\_min} + \frac{1}{\beta}\frac{W_{abs\_min}}{Fs} \tag{4.3.2.2}$$

The proportionality constant β (accounting for the proportionality expressed in both Relationships 4.3.1.1 and 4.3.1.2) is used as fitting parameter, to fit this model to training set experimental data. Equation 4.3.2.2 is one candidate model.

Alternatively, a more complex model is constructed as we remove the simplification of Fs having an exponent of 1 in Relationship 4.3.1.1. This adds complexity to the model by introducing an additional parameter n, the Fs exponent. The corresponding candidate model is:

Candidate Model 2:

$$W_{need}(Fs) = W_{abs\_min} + \frac{1}{\beta}\frac{W_{abs\_min}}{Fs^n} \tag{4.3.2.3}$$

where we vary the value of n and β simultaneously to fit Equation to data. This added model complexity is acceptable if it could achieve significantly better model accuracy.

Towards the other end of the complexity spectrum, we form candidate models that are simpler than Equation 4.3.2.2. One simpler candidate model is the equation that does not conform to Relationship 4.3.1.2, repeated here:

Candidate Model 3:

$$W_{need}(Fs) = W_{abs\_min} + \frac{1}{\beta}\frac{1}{Fs} \tag{4.3.2.4}$$

where β is the fitting parameter.

Another is the model that does not conform to the Relationship 4.3.1.1, using a different type of inverse function, but does conform Relationship 4.3.1.2:

Candidate Model 4:

$$W_{need}(Fs) = \beta W_{abs\_min} - Fs \tag{4.3.2.5}$$

Note that for this negative Fs inverse equation W_{abs_min} must be multiplied by a fitting parameter β value greater than 1.0, since W_{abs_min} is the absolute minimum value W_{need} can take on.

We fit these candidate models, determine validation accuracy, and choose the best model based on simplicity, intuition and accuracy in the following subsection.

4.3.3 Model Accuracy and Intuition

Using the experimental W_{need} data from the training benchmark circuits, under the architectures in Table 4.4, we fit all four candidate models. Validation is done using the same experiment flow, searching for W_{need} on validation circuits under the architectures given in Table 4.4. Training and validation results are summarized in Table 4.5. In Table 4.5, the first column identifies the candidate model, and the next four columns are the accuracy metrics. The final column gives the values of fitted parameters. There are 98 data points for each training and validation. The rows are sorted by descending validation RMSE (column 3), which is used to compare candidate model accuracy. Model selection is based on quantitative comparison of accuracy and qualitative comparison of simplicity and intuition.

Model No.	RMSE Training	RMSE Validation	MAPE Training	MAPE Validation	Fitted Parameters
4	5.8	6.0	13%	12%	$\beta = 1.33$
3	3.7	3.2	6.7%	6.4%	$\beta = 0.0979$
2	3.5	2.9	6.3%	6.0%	$\beta = 1.92$, $n = 1.32$
1	3.5	2.8	6.6%	5.9%	$\beta = 3.0$

Table 4.5: Accuracy of candidate models for reduced switch block flexibility

As can be seen in Table 4.5, the most accurate model is Candidate Model 1 with the smallest validation RMSE of 2.8. Candidate Model 4 is significantly worse than the others so we don't consider it. Candidate Model 2 and 3 are comparable in accuracy to Candidate Model 1, but they are worse in qualitative measures. We deem Candidate Model 2 worse than 1 because it has an extra fitting parameter (more complexity) but with no appreciable improvement in accuracy (only its training MAPE improved a little). Candidate Model 3 is simpler than 1 but the intuition in the former is that all circuits require the same amount of increase in W_{need} due to loss in Fs, which is not true as evidenced by Figure 4.5(a). We believe that the accuracy of Candidate Model 3 will deteriorate much more than 1, when generalized to less flexible architectures, as evidenced by Figure 4.5(b), where the increase in W_{need} ranges from 14 to 36

for different circuits (the span of the range, 22, cannot be accounted for by Candidate Model 3). Therefore Candidate Model 1 is better than 3. In sum, Candidate Model 1 is the best model for its combination of simplicity, intuition and accuracy. We shall refer to it from now on as the switch block flexibility model.

The value of β that gives the switch block flexibility model (Equation 4.3.2.2) the best fit to training data is 3 (actually 3.04 but rounded up for simplicity). The accuracy of this model is shown in Figures 4.6 and 4.7, where measured W_{need} (y-axis) is plotted against Fs (x-axis) over its range. The solid dots are the measured values, and the hollow dots are the model predicted values. We only show the maximum, minimum and median circuit (sorted by W_{abs_min} as shown in Tables 4.2 and 4.3) of each set due to the large volume of data.

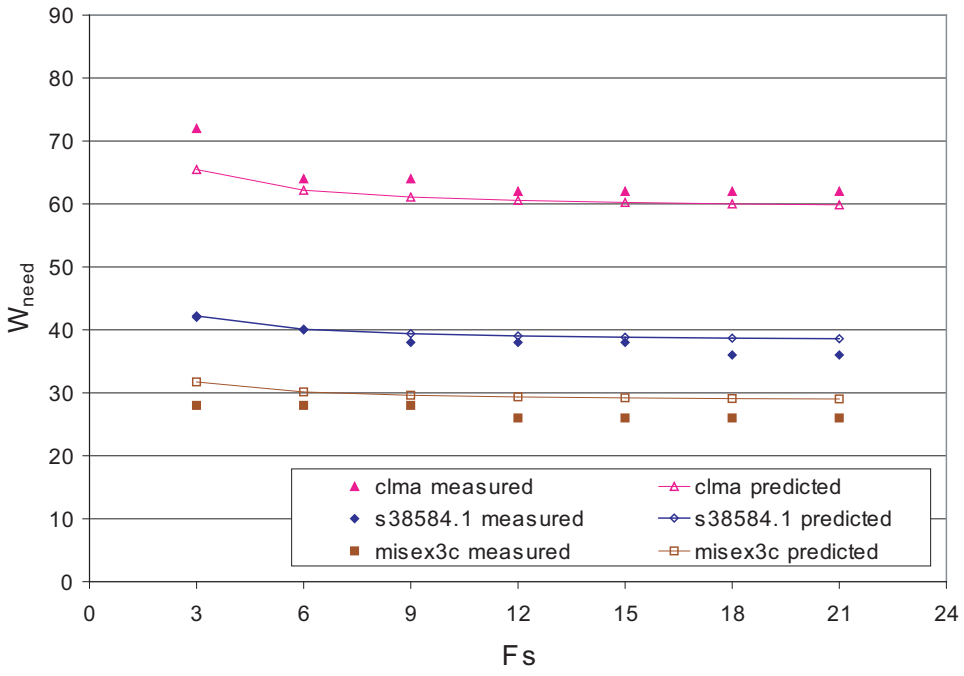


Figure 4.6: Accuracy of switch block flexibility model for circuits in the training set

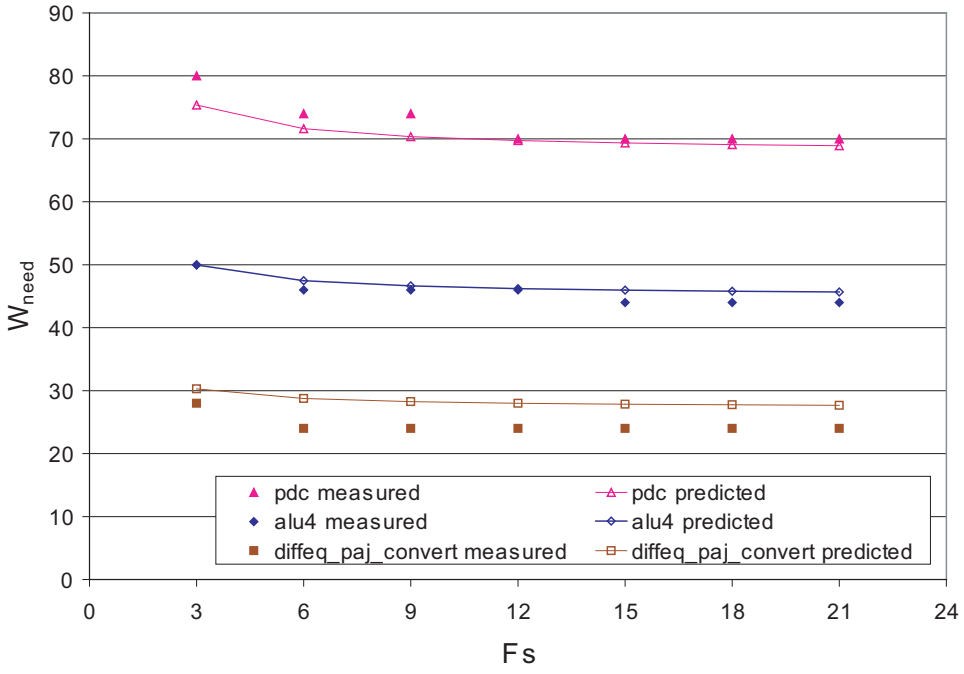


Figure 4.7: Accuracy of switch block flexibility model for circuits in the validation set

The breakdown of the accuracy of the model for the circuits shown in Figures 4.6 and 4.7 are presented in Table 4.6. In Table 4.6, the modeled architecture specified by Fs is listed in column 1. The next four columns are the measured W_{need}, predicted W_{need}, predicted $-$ measured, and the error as a percentage of the measured, for the training circuits. The final four columns are the same but for the validation circuits. The absolute error for each circuit average across different architectures specified by Fs is also computed and listed as the last row of each circuit's sub-table.

The intuition in the switch block flexibility model is in how it is constructed: the required channel width for successful routing is W_{abs_min} for a fully flexible FPGA, and a penalty on top of that must be paid as Fs is reduced. The penalty is proportional to the circuit and logic block's intrinsic demand W_{abs_min} and inversely related to Fs.

Further penalty must be paid if the flexibility in the connection block is reduced. The following subsection explores models for this reduction in flexibility.

	Training				Validation			
Fs	Measured	Predicted	Error	Error %	Measured	Predicted	Error	Error %
	clma				pdc			
3	72	65	-7	-9.1%	80	75	-5	-5.8%
6	64	62	-2	-2.8%	74	72	-2	-3.2%
9	64	61	-3	-4.5%	74	70	-4	-4.9%
12	62	61	-1	-2.3%	70	70	0	-0.4%
15	62	60	-2	-2.9%	70	69	-1	-0.9%
18	62	60	-2	-3.2%	70	69	-1	-1.3%
21	62	60	-2	-3.5%	70	69	-1	-1.5%
	Average Absolute Error			4.0%	Average Absolute Error			2.6%
	s38584.1				alu4			
3	42	42	0	0.5%	50	50	0	-0.1%
6	40	40	0	0.3%	46	47	1	3.2%
9	38	39	1	3.7%	46	47	1	1.4%
12	38	39	1	2.8%	46	46	0	0.5%
15	38	39	1	2.2%	44	46	2	4.5%
18	36	39	3	7.5%	44	46	2	4.1%
21	36	39	3	7.2%	44	46	2	3.8%
	Average Absolute Error			3.4%	Average Absolute Error			2.5%
	misex3c				diffeq_paj_convert			
3	28	32	4	13%	28	30	2	8.1%
6	28	30	2	7.7%	24	29	5	20%
9	28	30	2	5.8%	24	28	4	18%
12	26	29	3	13%	24	28	4	17%
15	26	29	3	12%	24	28	4	16%
18	26	29	3	12%	24	28	4	16%
21	26	29	3	12%	24	28	4	15%
	Average Absolute Error			11%	Average Absolute Error			16%

Table 4.6: Accuracy breakdown of switch block flexibility model

4.4 Generalizing for Connection Block Flexibilities

In previous sections, the routing architectures are assumed to have maximum connection block flexibility, i.e. $Fc_{in} = W$ and $Fc_{out} = W$. In this section, we model architectures of reduced connection block flexibility, where each input pin and output pin on a logic block can programmably connect to only a select number of routing wires adjacent to it, for example 2 in the simplified architecture shown in Figure 4.8.

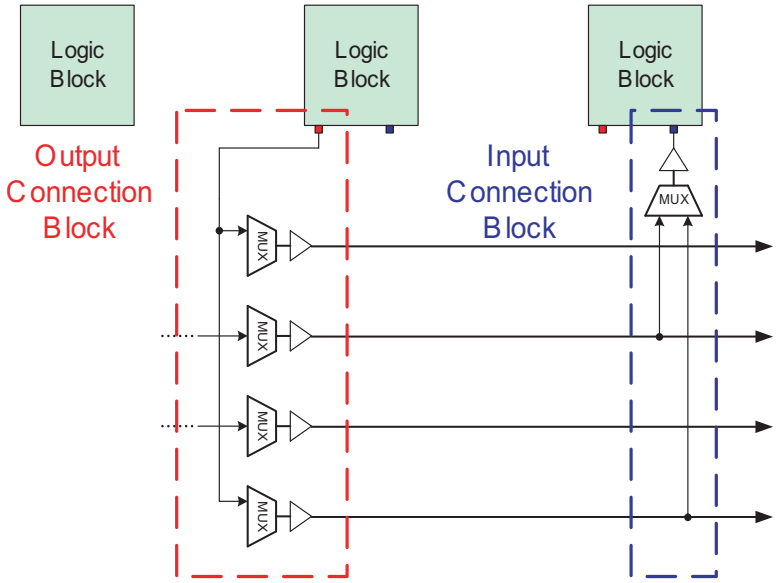


Figure 4.8: Simplified example showing input and output connection block

We generalize the switch block flexibility model to include Fc_{in} and Fc_{out}; we vary Fs, Fc_{in} and Fc_{out}, and keep the remaining routing architecture parameters at maximum flexibility. The range of Fs is as in the previous section. The range of Fc_{in} and Fc_{out} are the same under this architecture space. We consider 2 as the minimum value Fc_{in} and Fc_{out} can take on. The reason is that each pin being only able to connect to 1 routing wire would cause directional bias in the single-driver directional routing architecture, resulting in extremely poor routability which we have empirically found. We have also found that $Fc_{in} \geq W_{abs\_min}$ and $Fc_{out} \geq W_{abs\_min}$ are empirically equivalent to maximum flexibility $Fc_{in} = W$ and $Fc_{out} = W$, respectively. Therefore the range of Fc_{in} and Fc_{out} is chosen as [2,W_{abs_min}]. In experiments under architectures of varying Fc_{in} and Fc_{out}, we choose eight points from 2 to W_{abs_min}

Fs	Fc_{in}	Fc_{out}	L	Eqv
$[3, 21]$	[2,W_{abs_min}]	[2,W_{abs_min}]	1	1

Table 4.7: FPGA routing architectures with reduced flexibility

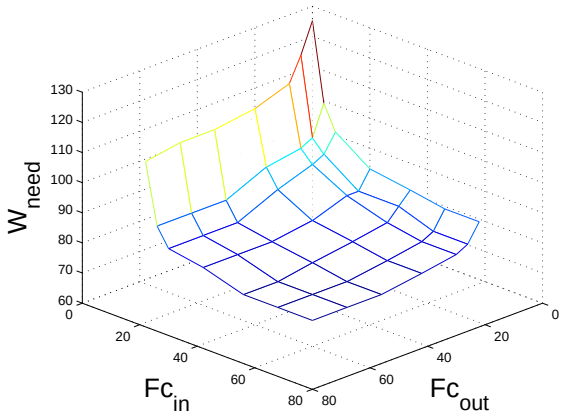
inclusively for each parameter. In sum, the routing architectures being modeled in this section are given in Table 4.7.

4.4.1 Empirical Analysis on Effects of Reduced Connection Block Flexibilities

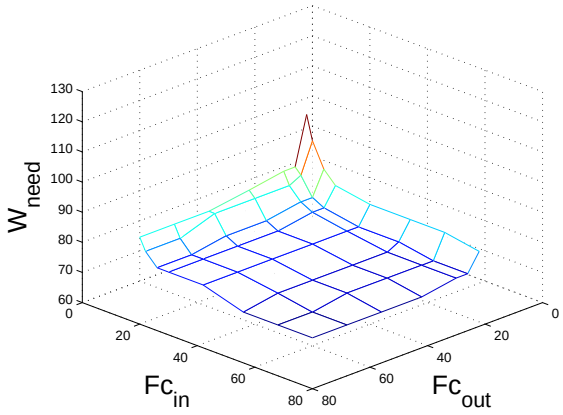
To generalize the interconnect model for reduced connection block flexibility, we first analyze experimental data. Using the W_{need} search experimental flow described in Section 3.4, on training benchmark circuits under the architectures indicated in Table 4.7, we generate training data. Since data for all training benchmark circuits have consistent trends, for clarity only the results for the largest circuit *clma* are displayed, in Figure 4.9.

In Figure 4.9 the x- and y-axes are Fc_{in} and Fc_{out} and each plot for a different Fs value (only $Fs = 3, 6, 9$ are shown for clarity). The z-axes show the W_{need} for successfully routing under the various routing architectures. There are a number of trends from which we can hypothesize analytical relationships between parameters and W_{need}.

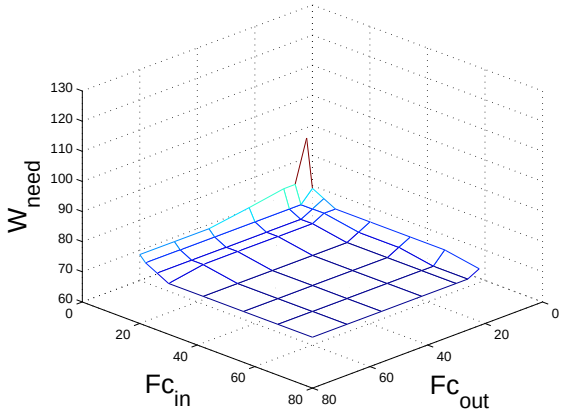
Firstly, in Figure 4.9(a) it can be seen that W_{need} is inversely related to Fc_{in} and Fc_{out}. This is obvious since less flexibility in either Fc_{in} and Fc_{out} must be compensated by an increase in track count, in order to still be routable. Moreover, the relationship between W_{need} and either Fc_{in} or Fc_{out} depends on the other parameter. For example, the W_{need} vs. Fc_{out} curve flattens out as Fc_{in} increases. This is expected since as there is more flexibility in Fc_{in}, less flexibility is needed in Fc_{out} – so a low Fc_{out} and a high Fc_{out} achieves similar W_{need} (a flat curve). This trade-off relationship is best captured by a multiplication of the Fc_{in} or Fc_{out} parameters. Also the trade-off is not symmetrical (the surface in Figure 4.9(a) is not symmetrical about the plane $Fc_{in} = Fc_{out}$); the impact of Fc_{in} on W_{need} is greater than that of Fc_{out}. Intuitively, this means it is more important to have flexibility at the end of a route (Fc_{in}) than at its beginning (Fc_{out}). A simple and effective way to capture this asymmetry is using different exponents for



(a) $Fs = 3$



(b) $Fs = 6$



(c) $Fs = 9$

Figure 4.9: Experimental trend of W_{need} over Fc_{in} and Fc_{out} for training circuit *clma*

Fc_{in} and Fc_{out}. Combining these empirical observations, We hypothesize the relationship of W_{need}, Fc_{in} and Fc_{out} to be

$$(W_{need} - W_{abs\_min}) \propto \left(\frac{1}{Fc_{in}}\right)^{\alpha_{in}} \left(\frac{1}{Fc_{out}}\right)^{\alpha_{out}} \tag{4.4.1.1}$$

Intuitively, one can think of input and output connection blocks together as an interconnected *switching matrix*, of which parameters Fc_{in} and Fc_{out} allocates switches in different parts. One can architect with different allocations and result in the same routability i.e. same W_{need}, just as one can trace a path of constant height on the surface plot of Figure 4.9(a). This trade-off is intuitively represented by the multiplication of parameters in Relationship 4.4.1.1.

As an extension one can think of this *switching matrix* as consisting of the input and output connection blocks and the switch blocks, together programmably connecting all logic block pins and routing wires in the FPGA. The parameters Fs, Fc_{in} and Fc_{out} describe the amount of switching in these three parts of the FPGA, and they can be traded off amongst each other to maintain a fixed level of routability (same W_{need}). This is observed in the empirical data as one examines the W_{need} surface for different Fs from Figure 4.9(a) to Figure 4.9(c). Therefore our hypothesis in Relationship 4.4.1.1 should be multiplied by the Fs term developed in Section 4.3.

Finally, in Figure 4.9 we notice that as Fc_{in} and Fc_{out} both approach the value W_{abs_min}, W_{need} is very close to the value determined by the switch block flexibility model in Equation 4.3.2.2, which we generalize for Fc_{in} and Fc_{out}. In other words, when $Fc_{in} = W_{abs\_min}$ the term associated with Fc_{in} should disappear from the generalized model. This can be effectively modeled using the term $\left(\frac{W_{abs\_min}}{Fc_{in}}\right)^{\alpha_{in}}$ instead of $\left(\frac{1}{Fc_{in}}\right)^{\alpha_{in}}$. The same for Fc_{out}.

Using relationship hypotheses formed in this subsection we construct candidate models for input and output connection block flexibility in the following subsection.

4.4.2 Constructing Candidate Models for Fc_{in} and Fc_{out}

In this subsection, candidate models of W_{need} as a function of Fs, Fc_{in} and Fc_{out} are constructed by generalizing the model for switch block flexibility in Equation 4.3.2.2, which is repeated here in Equation 4.4.2.1.

$$W_{need}(Fs) = W_{abs\_min} + \frac{1}{\beta}\frac{W_{abs\_min}}{Fs} \tag{4.4.2.1}$$

In all the following candidate models for Fc_{in} and Fc_{out} by generalizing Equation 4.4.2.1, the parameter β is fixed at 3, optimal value determined previously.

The first candidate model is based on all the observational hypotheses in the previous subsection.

Candidate Model 1:

$$W_{need}(Fs, Fc_{in}, Fc_{out}) = W_{abs\_min} + \frac{1}{\beta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)^{\alpha_{in}}\left(\frac{W_{abs\_min}}{Fc_{out}}\right)^{\alpha_{out}} \tag{4.4.2.2}$$

where α_{in} and α_{out} are fitting parameters to capture the asymmetry in the dependence of W_{need} on Fc_{in} and Fc_{out}.

We construct other candidate models by modifying Candidate Model 1. First, we can test the importance of the asymmetry hypothesis by assuming the opposite; let Fc_{in} and Fc_{out} have the same exponent of 1 resulting in:

Candidate Model 2:

$$W_{need}(Fs, Fc_{in}, Fc_{out}) = W_{abs\_min} + \frac{\alpha}{\beta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)\left(\frac{W_{abs\_min}}{Fc_{out}}\right) \tag{4.4.2.3}$$

Instead of having two fitting parameters as in Candidate Model 1, this model only has one fitting parameter α to scale the penalty term accounting for Fc_{in} and Fc_{out}. This form is especially simple because there is only one fitting parameter and it is easy for analytic calculations since all exponents are 1. Its simplicity is so compelling that we give it a special name: *the symmetric simple model*.

Alternatively, we can remove W_{abs_min} from Fc_{in} and Fc_{out} terms:

Candidate Model 3:

$$W_{need}(Fs, Fc_{in}, Fc_{out}) = W_{abs\_min} + \frac{\alpha}{\beta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{1}{Fc_{in}}\right)^{\alpha_{in}}\left(\frac{1}{Fc_{out}}\right)^{\alpha_{out}} \tag{4.4.2.4}$$

however, due to the multiplication we must add one more fitting parameter to ensure that for any value of Fc_{in} and Fc_{out} $W_{need}(Fs, Fc_{in}, Fc_{out})$ must be higher than $W_{need}(Fs)$ in Equation 4.4.2.1.

Finally, we can hypothesize an additive penalty term, instead of the multiplication. This leads to a candidate model:

Candidate Model 4:

$$W_{need}(Fs, Fc_{in}, Fc_{out}) = W_{abs\_min} + \frac{1}{\beta}\frac{W_{abs\_min}}{Fs} + \alpha\left(\frac{1}{Fc_{in}}\right)^{\alpha_{in}}\left(\frac{1}{Fc_{out}}\right)^{\alpha_{out}} \quad (4.4.2.5)$$

these candidate models are fit to training data and the best model is chosen based on a balance of simplicity, intuition and accuracy in the following subsection.

4.4.3 Model Accuracy and Intuition

We fit all four candidate models to training data, obtained under architectures in Table 4.7. Validation data are obtained by the same experimental flow, searching for W_{need} under architectures in Table 4.7. The results of candidate model fitting and prediction are summarized in Table 4.8. In Table 4.8, the first column identifies the candidate model, and the next four columns are the accuracy metrics. The final column gives the values of fitted parameters. There are 6020 data points for each training and validation. The rows are sorted by descending validation RMSE (column 3). Model selection is based on a balance of validation RMSE accuracy and qualitative judgement of simplicity and intuition.

Model No.	RMSE Training	RMSE Validation	MAPE Training	MAPE Validation	Fitted Parameters
2	4.8	6.9	6.3%	8.6%	$\alpha = 0.0418$
4	4.8	6.6	7.2%	7.8%	$\alpha = 16.1$, $\alpha_{in} = 0.881$, $\alpha_{out} = 0.434$
3	4.1	5.3	7.5%	7.0%	$\alpha = 13.0$, $\alpha_{in} = 0.457$, $\alpha_{out} = 0.273$
1	3.9	4.6	7.6%	6.5%	$\alpha_{in} = 0.50$, $\alpha_{out} = 0.25$

Table 4.8: Accuracy of candidate models for reduced connection block flexibility

Quantitatively the best model is Candidate Model 1, with the smallest validation RMSE of 4.6. Candidate Models 3 and 4 are more complex than Candidate Model 1, having an extra fitting parameter. However, they are not significantly better in accuracy to justify being better than Candidate Model overall. In fact they are worse in predicting validation data,

suggesting that the inclusion of the extra fitting parameter has led to over-fitting; the inclusion was necessary in both Candidate Models 3 and 4 because of the hypotheses on which they are formed. Therefore Candidate Model 1 is a better choice overall compared to 3 and 4.

Candidate Model 2 is worse than Candidate Model 1 in all accuracy metrics except MAPE Training. Although Candidate Model 2 is simpler as it has two fewer fitting parameters than Candidate Model 1 (since all exponents are set to 1), it is inferior for two reasons. One, it does not give the insight that Fc_{in} has a greater impact on W_{need} than Fc_{out} (their exponents are equal), which is observed in the asymmetry of Figure 4.9 (equal exponents give symmetry). Candidate Model 2 misses the important intuition that it is more important to have flexibility at the end of a route (Fc_{in}), when the router has nearly exhausted all choices, than at its beginning (Fc_{out}).

The second reason, for which Candidate Model 2 is inferior to 1, is that we believe that the accuracy gap between Candidate Model 2 and 1 is significant and will dramatically increase in favor of Candidate Model 1 for less flexible architectures (modeled later on). We tested and confirmed this hypothesis, using candidate model 2 – *the symmetric simple model* – to model less flexible architectures in Section 4.5.5.

In conclusion, Candidate Model 1 is the best overall model considering a balance of simplicity, intuition and accuracy. We shall call it the switching matrix model, referring to that it models the three components of the switching matrix that connects all routing wires and logic block pins of the FPGA.

The value of α_{in} and α_{out} that gives the switching matrix model (Equation 4.4.2.2) best fit to training data are 0.5 and 0.25 respectively. These are rounded up values for simplicity, and easier analytic calculations (note that in Table 4.8 the accuracy values for candidate model 1 correspond to the rounded up parameter values). Due to the large volume of data, we only show the accuracy of the model for the largest training circuit and validation circuit, respectively in Figures 4.10 and 4.11. In Figures 4.10 and 4.11, the x- and y-axes are Fc_{in} and Fc_{out}. There are two identical subplots (placed horizontally together) viewed at different azimuths, and each pair of subplots are shown for a Fs value (ordered vertically, only $Fs = 3, 6, 9$ are shown for clarity). The z-axes show the W_{need} for successfully routing under the various routing

architectures. For each set of axes, there is a plot of measured and a plot of model predicted values of W_{need} overlayed to show model accuracy. Notice in some cases the two plots intersect, which shows good prediction.

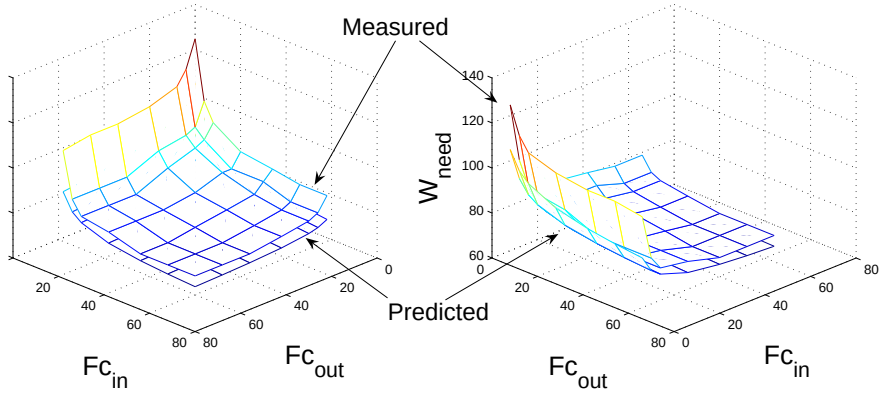
The numerical breakdown of the accuracy for the circuits in Figures 4.10 and 4.11 are presented in Tables A.1 to A.6 in Appendix A. Note both training and validation data on some very inflexible routing architectures (e.g. $Fs = 3$, $Fc_{in} = 2$ and $Fc_{out} = 2$) are not available on the large circuits due to prohibitively long experimental run times; these data are omitted (only for the these large circuits).

The intuition behind the current form of interconnect model is as follows. The model is reprised here

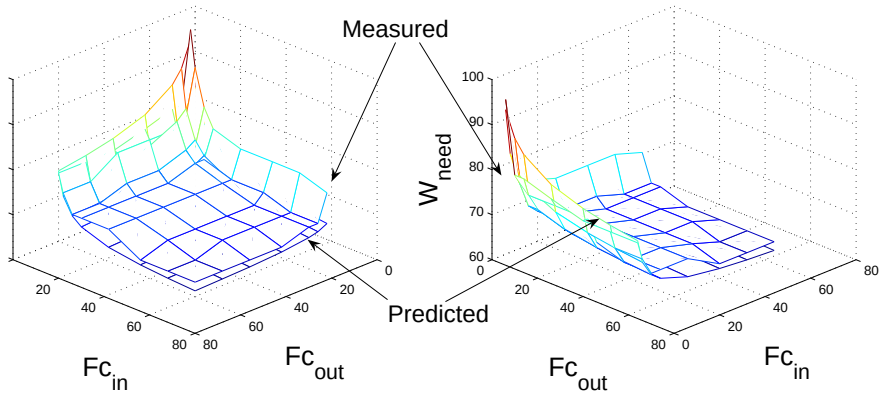
$$W_{need}(Fs, Fc_{in}, Fc_{out}) = W_{abs\_min} + \frac{1}{\beta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)^{\alpha_{in}}\left(\frac{W_{abs\_min}}{Fc_{out}}\right)^{\alpha_{out}} \quad (4.4.3.1)$$

The FPGA routing demand is an absolute minimum, W_{abs_min}, plus a penalty for reduced flexibilities in the three components, switch/input conection/output connect blocks, of the switching matrix that programmably connects all logic block pins and routing wire segments in the FPGA. The three parameters Fs, Fc_{in} and Fc_{out} describe the amount of switching in these parts of the FPGA and they can be traded off amongst each other to maintain the same routing demand. Our model reflects this in that as long as their product in Equation 4.4.3.1 is the same, different distribution of switching can result in the same routing demand. The choice of a product to model the three parameters conveys their "connectedness" in the switching matrix, much akin to Brown's model [10] (in Section 2.4.4) in which the likelihood of events associated with switch/input conection/output connect blocks are multiplied.

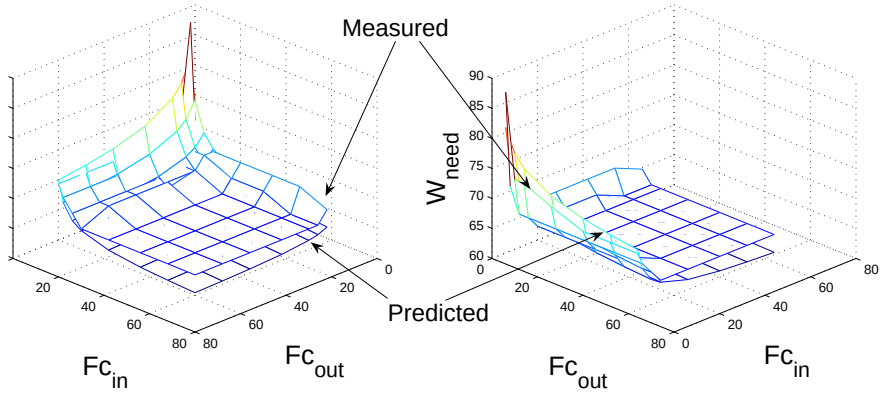
The inflexibility modeled so far pertains to reduced switching in the FPGA. A different type of inflexibility exists when routing wire segments span multiple logic blocks. As wire segments span longer, more total wires are needed to complete routing, thus incurring an additional penalty to routing demand. We model this inflexibility/penalty in the following section.



(a) $Fs = 3$

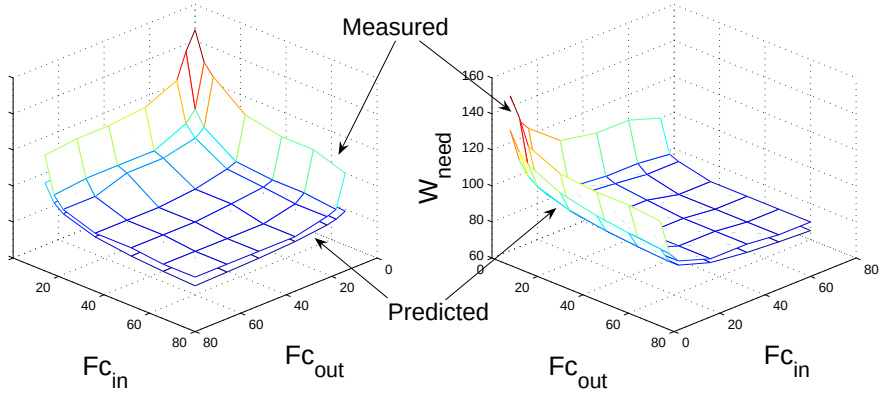


(b) $Fs = 6$

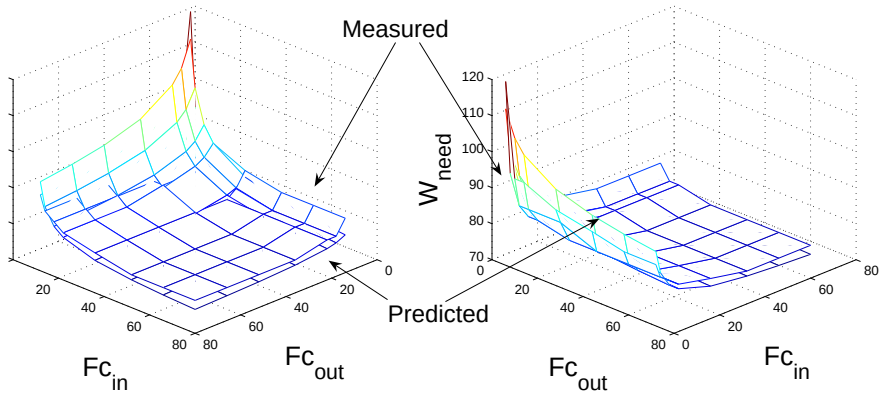


(c) $Fs = 9$

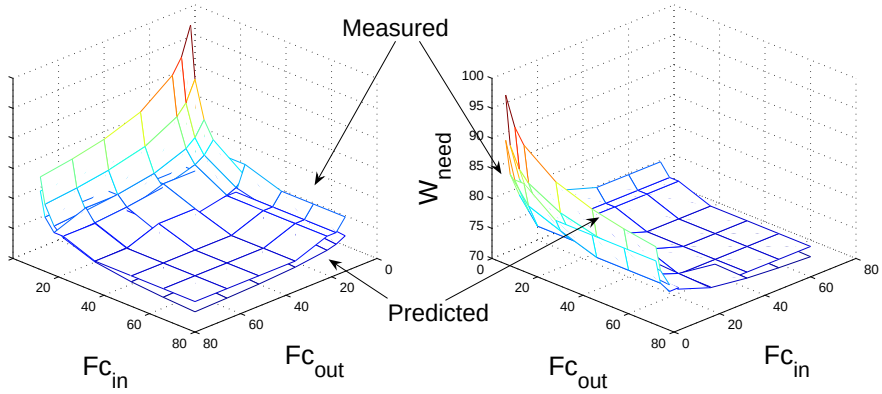
Figure 4.10: Accuracy of the switching matrix model for training circuit *clma*



(a) $Fs = 3$



(b) $Fs = 6$



(c) $Fs = 9$

Figure 4.11: Accuracy of the switching matrix model for validation circuit *pdc*

4.5 Model of Long Wire Segments

All intermediate models up to this point model routing architectures of varying switching flexibilities under the assumption that interconnect wire segments are all unit length ($L = 1$) – measured in number of logic blocks it spans. Longer wire segments (where the parameter L is greater than 1), depicted in Figure 4.12, bring a different form of inflexibility. This inflexibility causes higher total wire length use, leading to a higher W_{need}. This section aims at modeling the penalty in W_{need} as a result of increased wire length.

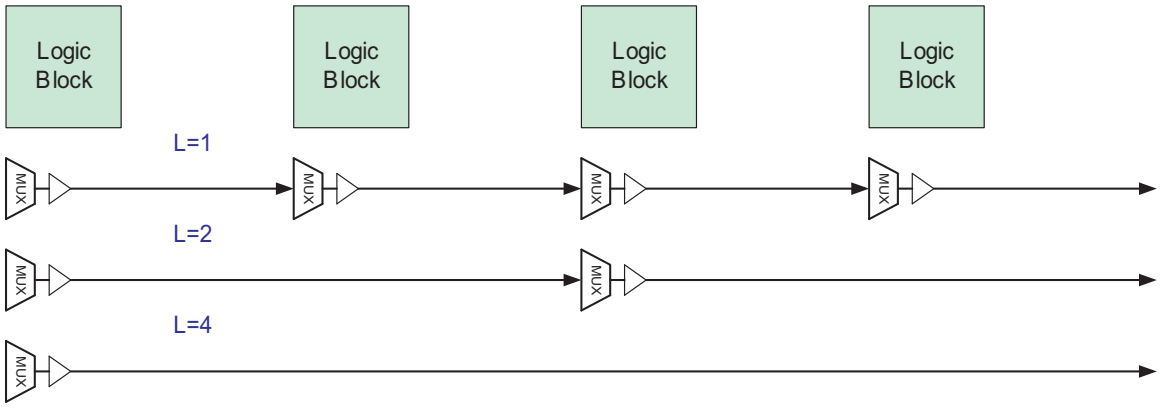


Figure 4.12: Wire segment lengths 1, 2 and 4

Recall from Chapter 2 that in the scope of this work we assume all wires in an architecture will have the same length L. The values of interest for L is from lengths {1, 2, 4, 6, 8}, as current research [8, 55] and modern commercial architectures are in this range. Stratix II predominantly uses length 4 wires (89% of all tracks) [42], Virtex 4 predominantly uses length 6 wires (65% of all tracks) [65]. The architectures modeled in this section are summarized in Table 4.9.

There are two major effects of using longer wire segments that increase routing demand W_{need}: *segmentation waste*, and Fc_{in} *Reduction*, which we discuss and model in turn in the

Fs	Fc_{in}	Fc_{out}	L	Eqv
$[3, 21]$	[2,W_{abs_min}]	[2,W_{abs_min}]	[1,8]	1

Table 4.9: FPGA routing architectures with long interconnect wire segments

next two subsections.

4.5.1 Segmentation Waste

Architectures with wire segments of length $L > 1$ has the inflexibility that routes sometimes use a wire segment to make a connection that is less than length L. When this occurs the wire segment is only used for a portion of its length, and the remaining portion is wasted. We refer to the portion of a used wire segment that is wasted as *segmentation waste*.

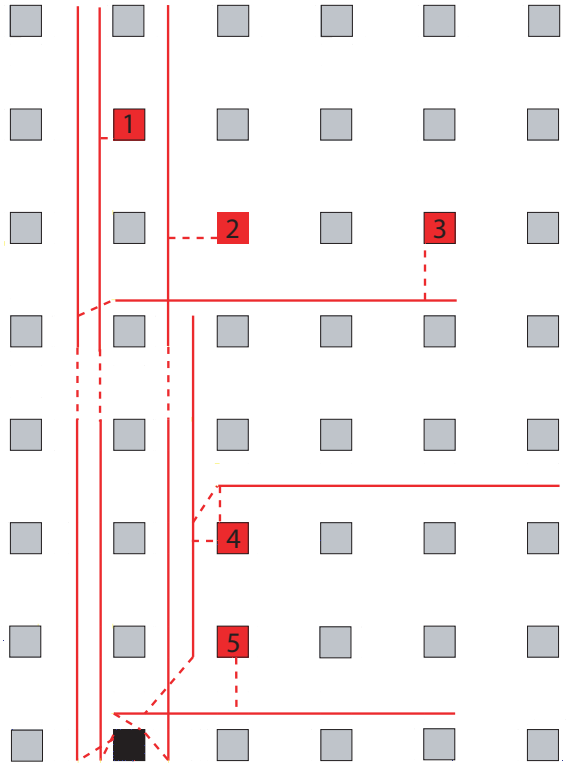


Figure 4.13: Routing using length 4 wire segments

Segmentation waste occurs either at horizontal-to-vertical (or vice versa) turns – which we call *turn waste*– or wire to input pin connections – which we call *pin waste*, as illustrated in Figure 4.13. Figure 4.13 shows an example routing of a net with an architecture consisting of only length 4 wire segments. The source block is the black block (near bottom-left) and sink blocks are numbered 1 to 5; solid lines are wire segments and dashed lines indicate switches in use. For example, the route to sink block 3 has a turn waste of 3, and the route to sink block

1 has a pin waste of 1. The amount of waste is measured by length.

Waste does not occur at output pin connections because, in a single-driver routing architecture all output pins connect at the beginning of a wire segment by definition. Also waste does not occur when all wire segments are of unit length; every length of used wire is useful for $L = 1$ architecture.

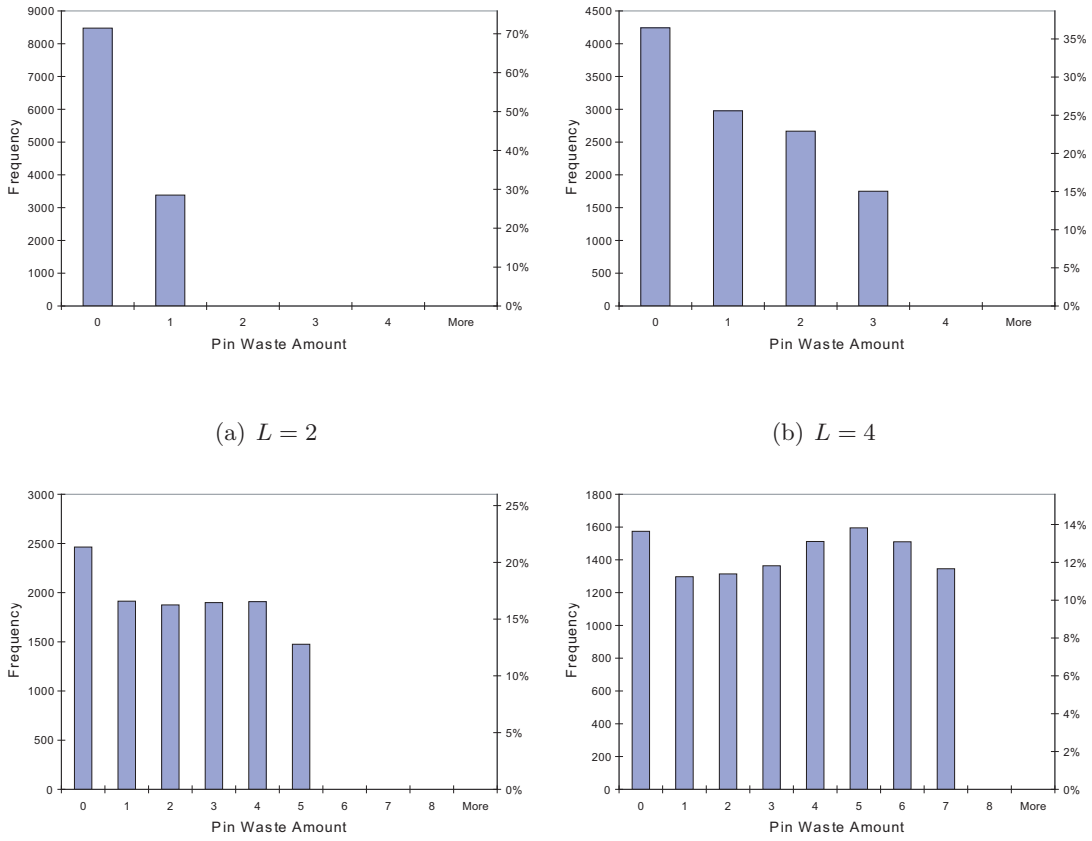
A model of total segmentation waste is derived to capture its effect on routing demand W_{need}. First, we empirically found that the dominant type of waste is from pin waste. This is seen in Table 4.10, which shows the average percentage of total segmentation waste attributed to pin waste, averaged across all training benchmark circuits. The experiment flow is the same W_{need} search, and at W_{need} the routing solution is parsed to count the total segmentation waste and waste due to wire to input pins. The architecture is indicated in column 1 and row 1 (values of Fs and L) and other parameters are $Fc_{in} = 8$, $Fc_{out} = 8$, and $Eqv = 1$ (and same logic block architecture, $N = 10$ and $I = 22$).

Shown in Table 4.10, in the majority of architectures high percentage of segmentation waste is due to pin waste. And this trend is stronger as L increases. This can be expected since a good placement leads to lots of short distance connections, which causes pin waste. As L increases, the connections' distance does not increase, but more pin waste results. This leads to higher percentage of total segmentation waste being attributed to pin waste as L increases. This is a key point to consider for the modeling of total segmentation waste.

The second observation is that the possible values of pin waste, for a wire segment of length L connecting to an input pin, are 0, 1, 2, ..., $L - 1$ (0 is no waste, and $L - 1$ is using L as a length 1 wire and maximum waste). Furthermore we empirically found that the likelihood of

Fs/L	2	4	6	8
3	66.2%	77.2%	80.5%	81.3%
6	57.6%	74.2%	78.5%	80.6%
9	57.3%	74.8%	78.8%	80.9%
12	57.9%	74.5%	79.2%	81.0%

Table 4.10: Percentage of total segmentation waste that is pin waste, of training circuits



(a) $L = 2$

(b) $L = 4$

(c) $L = 6$

(d) $L = 8$

Figure 4.14: Pin waste distribution for training circuit *clma*

occurrence for each value of pin waste is approximately uniform, and more so as L increases. This is not unexpected since all segments in the FPGA are the same length. This can be seen in Figure 4.14.

Figure 4.14 shows the pin waste distribution of the largest training circuit *clma*, for segment lengths 2, 4, 6, and 8; the other routing parameters are set at $Fs = 6$, $Fc_{in} = 8$, $Fc_{out} = 8$, and $Eqv = 1$ (and same logic block architecture, $N = 10$ and $I = 22$). The trend Figure 4.14 is typical of all circuits. The experimental flow is the same W_{need} search (from Section 3.4), and at the required channel width W_{need} the routing solution is parsed to count the segmentation waste on each used input pin. In Figure 4.14, the x-axis shows the waste amount on a used input pin and y-axis shows the number of occurrences. For reference of statistical significance, each histogram has 12092 data points (which is the number of used input pins in the circuit).

Looking at Figure 4.14, a trend certainly can be observed. For architectures $L = 6$ and 8 the pin waste distribution is approximately uniform. For architectures $L = 2$ and 4, zero waste is more likely to occur than other waste values, which can be expected because of the short lengths.

We combine the two observations to create a model of total segmentation waste as a function of L and the number of used input pins: *We model total segmentation waste as the sum of H random draws from a uniform distribution ranging from 0 to $L-1$, where H is the total number of used input pins.* In essence, we model total segmentation waste as pin waste with a uniform distribution.

This uniform distribution models total segmentation waste well. At high values of L (6 and 8), a uniform distribution is approximately the same as the pin waste distribution, in Figure 4.14(c) and 4.14(d). This models total segmentation waste distribution well, since at high values of L most of total segmentation waste is due to pin waste, as seen in Table 4.10.

At low values of L (2 and 4), a uniform distribution models more waste than the pin waste distribution, since the in latter distribution (in Figure 4.14(a) and 4.14(b)) zero waste is more likely to occur than other waste values. The additional waste in the uniform distribution effectively models the turn waste, which accounts for good portion of the total segmentation waste for low values of L, as seen in Table 4.10.

Therefore, for low and high values of L, a uniform distribution models total segmentation waste well. This argument is supported by accuracy results reported later in this section and in Section 4.5.4.

With the model of total segmentation waste in hand, we can derive an expression for the additional tracks per channel needed due to segmentation waste. Since the probability of segmentation waste is $\frac{1}{L}$ for all possible values 0, 1, 2, ..., $L-1$, the expected value of segmentation waste per used input pin can is:

$$\begin{aligned}\text{Segmentation Waste Per Used Input Pin} =& \frac{1}{L}(0) + \frac{1}{L}(1) + ... + \frac{1}{L}(L-1) \\ =& \frac{L-1}{2}\end{aligned} \tag{4.5.1.1}$$

The number of used input pins per channel segment is $\frac{\lambda}{2}$, since there are two channel

Fs	Fc_{in}	Fc_{out}	L	Eqv
$[3, 21]$	W	W	[1,8]	1

Table 4.11: FPGA routing architectures with reduced switch block flexibility and long wires

segments per logic block. The product of the segmentation waste per used input pin times the number of used input pins per channel segment gives the extra wire required in each channel segment, on average, which is the amount of increase needed in track count per channel (W_{need}):

$$\text{Segmentation Waste Per Channel Segment} = \frac{\lambda(L-1)}{4} \tag{4.5.1.2}$$

This model of segmentation waste confirms some intuitive notions. When $L = 1$, there is no waste. As L increases above 1, the segmentation waste per channel segment increases as proportional to λ and $L - 1$.

To validate this model of segmentation waste's impact on W_{need} we generalize an earlier intermediate model, the switch block flexibility model, for the architecture in Table 4.11. The switch block flexibility model was developed for architectures of only wire segments of length 1. It is repeated here:

$$W_{need}(Fs) = W_{abs\_min} + \frac{1}{\beta}\frac{W_{abs\_min}}{Fs} \tag{4.5.1.3}$$

For the same architectures but with wire segments greater than 1, the total wire length increases, due to segmentation waste. The wire length increase per channel segment is $\frac{\lambda(L-1)}{4}$ given in 4.5.1.2. To accommodate the width of each channel segment, W, must increase by the same amount. The maximum channel width, W_{need}, is expected to increase by the same amount. Therefore the generalized model of W_{need}, as a function of switch block flexibility and wire segment length, can be modeled as:

$$W_{need}(Fs, L) = W_{abs\_min} + \frac{1}{\beta}\frac{W_{abs\_min}}{Fs} + \frac{\lambda(L-1)}{4} \tag{4.5.1.4}$$

where β remains fixed at previously determined value of 3.

For validation, we ran experiments determining W_{need} using the flow in Section 3.4, under architectures where Fs and L are varied across their range in Table 4.11 and all other parameters

L	2	4	6	8
RMSE	3.2	3.2	5.2	9.7
MAPE	5.3%	4.3%	6.2%	11%
% of W_{need} Accounted for by Segmentation Waste	6.7%	18%	26%	33%

Table 4.12: Breakdown of accuracy of model of Fs and L (Equation 4.5.1.4)

at maximum flexibility[2]. Notice that this model of segmentation waste is *not* constructed based on observation empirical data of training circuits (i.e. their W_{need} values) and it requires no fitting parameters. The accuracy of model in Equation 4.5.1.4 has an RMSE of 5.9 and MAPE of 6.8%.

For more insight on the validity of the derived model on wire segmentation length, we break down the model accuracy for different lengths in Table 4.12. Table 4.12 shows the model accuracy for segment lengths 2, 4, 6 and 8 in first three rows. The fourth rows gives the percentage of W_{need} that is accounted for by the derived term $\frac{\lambda(L-1)}{4}$, averaged across all benchmark circuits for the column-specified segmentation length. This metric shows that this quantity accounts for a significant portion of the routing demand modeled in Equation 4.5.1.4.

Going forward, it is not sufficient to generalize the switching matrix model (containing Fs, Fc_{in}, Fc_{out}) simply by considering segmentation waste, because another important effect of long wire segments on W_{need} needs to be captured in the model. This effect is described in the following subsection.

4.5.2 Fc_{in} Reduction

Using longer wire segment lengths not only increases routing demand W_{need} because of increased total wire length due to segmentation waste, but it has a second effect that causes an increase in W_{need}, which comes from an interaction between input connection block flexibility (the Fc_{in} parameter) and the segmentation waste. This effect is illustrated in Figure 4.15, which shows

[2]Because of restrictions imposed by long wire segment lengths, some architectures in this space are not possible for certain benchmark circuits. For example, $Fs = 21$ and $L = 8$ would require a channel width of 112 ($\frac{2L \cdot Fs}{3}$) just to meet architecture specification – i.e. to have enough distinct wires to connect to in the switch block. Benchmark circuits that require smaller channel widths than 112 will be clipped at demanding $W_{need} = 112$ – for such cases we discard the data point (for that circuit only)

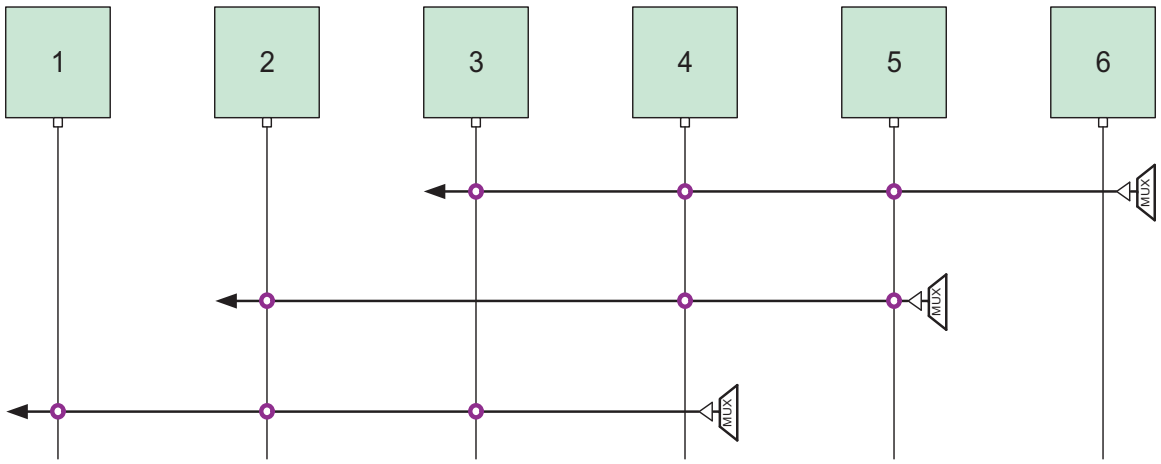


Figure 4.15: Fc_{in} reduction effect due to long wire segment lengths

an architecture with three length four wire segments passing by six input pins. The wire-to-input-pin programmable switches are shown as circles and the figure shows an architecture where Fc_{in} is less than the maximum possible. Suppose that the top length four wire segment is used to connect into logic block pin number 5. A side effect of this routing choice is that the associated segmentation waste led to fewer opportunities to connect into pin number 3 and 4. This amounts to an effective reduction of Fc_{in} due to segmentation waste. For contrast, Fc_{in} would not be effectively reduced if unit length wires were used – where there is no segmentation waste. We intuitively expect this effect to most dramatically increase routing demand when Fc_{in} is low. The effect of Fc_{in} reduction is not noticeable when there is an abundance of switching as in the architectures of Table 4.11 in the previous subsection, where Fc_{in} is at maximum. Which is why without modeling this effect we can still predict accurately in that architecture subspace.

This observation is supported by empirical data. We ran experiments using the flow searching for W_{need} of Section 3.4, under the architectures in Table 4.9, using training benchmark circuits. Figure 4.16 is a plot of subset of the experimental data (representing the typical trends), showing the difference $W_{need}(L = 4) - W_{need}(L = 1)$ (z-axis) averaged across training benchmark circuits for a range of Fs (x-axis) and Fc_{in} (y-axis), at $Fc_{out} = 8$. Figure 4.16 shows that the increase in W_{need}, as a result of using $L = 4$ wire segments instead of unit length wire

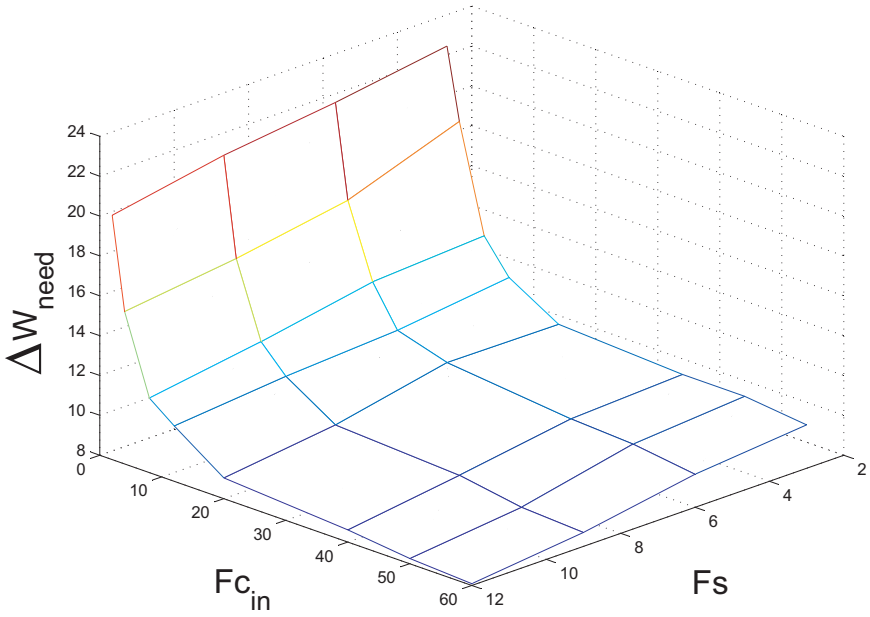


Figure 4.16: Average increase in W_{need} going from $L = 1$ to $L = 4$ for training circuits

segments, is dependent on Fc_{in}. The dependence is not significant when Fc_{in} is large: from $Fc_{in} = 30$ to $Fc_{in} = 60$ the curve is nearly flat, showing a near constant increase in W_{need} of around 8 tracks (This shows why the $W_{need}(Fs, L)$ model in Equation 4.5.1.4, which does not consider Fc_{in} reduction effect, is sufficiently accurate for routing architectures of Table 4.11). However, at low values of Fc_{in}, the increase in W_{need} is dramatic as hypothesized by the Fc_{in} reduction effect due to longer wire segment length, described above.

Therefore it is clear that, to generalize the switching matrix model (containing Fs, Fc_{in}, Fc_{out}) for L, it is not enough to model increase in W_{need} by segmentation waste $\frac{\lambda(L-1)}{4}$ alone; one must also consider Fc_{in} reduction effect. To account for the increased routing demand due to Fc_{in} reduction effect a number of candidate models are proposed.

4.5.3 Constructing Candidate Models

The increase due to Fc_{in} reduction occurs as a function of the amount of segmentation waste: more segmentation waste (caused by longer L) leads to more Fc_{in} reduction. So the term

accounting for Fc_{in} reduction should be multiplicative to segmentation waste quantity $\frac{\lambda(L-1)}{4}$. Furthermore, the increase due to Fc_{in} reduction is inversely related to Fc_{in} as seen in Figure 4.16, suggesting the term $\frac{1}{{Fc_{in}}^{\alpha_{seg}}}$, where α_{seg} is a parameter fitted to accurately model the inverse trend. Combining the above hypotheses, one possible expression for the Fc_{in} reduction increase is:

$$\frac{\lambda(L-1)}{4}\left(\frac{1}{{Fc_{in}}^{\alpha_{seg}}}\right) \tag{4.5.3.1}$$

Recall previous routing demand model (containing Fs, Fc_{in}, Fc_{out}) is

$$W_{need}(Fs, Fc_{in}, Fc_{out}) = W_{abs\_min} + \frac{1}{\beta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)^{\alpha_{in}}\left(\frac{W_{abs\_min}}{Fc_{out}}\right)^{\alpha_{out}} \tag{4.5.3.2}$$

and previous section concluded that segmentation waste contributes to W_{need} by $\frac{\lambda(L-1)}{4}$. Adding that increase and the increase due to Fc_{in} reduction (in Equation 4.5.3.1) to the previous model, we obtain Candidate Model 1:

Candidate Model 1:

$$\begin{aligned} W_{need}(Fs, Fc_{in}, Fc_{out}, L) = W_{abs\_min} &+ \frac{1}{\beta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)^{\alpha_{in}}\left(\frac{W_{abs\_min}}{Fc_{out}}\right)^{\alpha_{out}} \\ &+ \frac{\lambda(L-1)}{4}\left(1+\frac{1}{{Fc_{in}}^{\alpha_{seg}}}\right) \end{aligned} \tag{4.5.3.3}$$

where β, α_{in} and α_{out} are fixed at their previously determined values of 3, 0.5 and 0.25 respectively (the same statement is made for all candidate models presented in this section), and α_{seg} is a fitting parameter, to be determined by training data.

However, Candidate Model 1 is very complex to use because of potentially having two different exponents for Fc_{in}. While we want to model the effect of Fc_{in} reduction accurately, we also want model simplicity. It may be sufficient to simply have a fixed positive exponent for Fc_{in} in the penalty term of Equation 4.5.3.1: so the trend (in Figure 4.16) of Fc_{in} reduction is modeled, but (perhaps) not as accurately as when model contains a parameter α_{seg} fitted to training data.

Based on this, we propose a candidate model that sacrifices accuracy for simplicity, by fixing the exponent α_{seg} to equal α_{in}, which is determined to be 0.5. This choice is driven by model simplicity:

Candidate Model 2:

$$W_{need}(Fs, Fc_{in}, Fc_{out}, L) = W_{abs\_min} + \frac{1}{\beta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)^{\alpha_{in}}\left(\frac{W_{abs\_min}}{Fc_{out}}\right)^{\alpha_{out}} + \frac{\lambda(L-1)}{4}\left(1 + \frac{1}{{Fc_{in}}^{\alpha_{in}}}\right) \tag{4.5.3.4}$$

On the other end of the spectrum one can argue for a more complex model, noting that Figure 4.16 shows increase in W_{need} due to L has a slight dependence on Fs. Such a model is:

Candidate Model 3:

$$W_{need}(Fs, Fc_{in}, Fc_{out}, L) = W_{abs\_min} + \frac{1}{\beta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)^{\alpha_{in}}\left(\frac{W_{abs\_min}}{Fc_{out}}\right)^{\alpha_{out}} + \frac{\lambda(L-1)}{4}\left(1 + \frac{1}{{Fc_{in}}^{\alpha_{seg1}} Fs^{\alpha_{seg2}}}\right) \tag{4.5.3.5}$$

where α_{seg1} and α_{seg2} are fitting parameters of the model, that capture the curvature of Figure 4.16.

As discussed in the previous subsection: "generalizing the switching matrix model (containing Fs, Fc_{in}, Fc_{out}) simply by considering segmentation waste is not sufficient". To test this hypothesis we construct a candidate model that only models segmentation waste:

Candidate Model 4:

$$W_{need}(Fs, Fc_{in}, Fc_{out}, L) = W_{abs\_min} + \frac{1}{\beta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)^{\alpha_{in}}\left(\frac{W_{abs\_min}}{Fc_{out}}\right)^{\alpha_{out}} + \frac{\lambda(L-1)}{4} \tag{4.5.3.6}$$

Finally, instead of modeling the penalty due to L as additive to switching matrix flexibility penalty, one can model L in a similar fashion to Fc_{in} and Fc_{out} in the last section, ignoring the segmentation waste derivation, arriving at:

Candidate Model 5:

$$W_{need}(Fs, Fc_{in}, Fc_{out}, L) = W_{abs\_min} + \frac{1}{\beta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)^{\alpha_{in}}\left(\frac{W_{abs\_min}}{Fc_{out}}\right)^{\alpha_{out}}(L)^{\alpha_L} \tag{4.5.3.7}$$

where α_L is a fitting parameter.

These five candidate models of generalizing switching matrix model for L are evaluated and compared in the following subsection.

4.5.4 Model Accuracy

Experimental data are generated by the W_{need} search flow in Section 3.4, under architectures in Table 4.9, using training and validation circuits. Candidate models that require fitting are fitted to training data. The results are summarized in Table 4.13. In Table 4.13, the first column identifies the candidate model and the next four columns are the accuracy metrics. The final column gives the fitted parameter values. The rows are sorted by descending validation RMSE (column 3). For reference of statistical significance, there are 13688 data points from training circuits and 13049 data points from validation circuits.

Model No.	RMSE Training	RMSE Validation	MAPE Training	MAPE Validation	Fitted Parameters
5	10	9.0	11%	8.6%	$\alpha_L = 0.686$
4	8.6	7.8	8.7%	7.2%	no new fitting parameter
2	6.7	6.2	7.9%	6.5%	no new fitting parameter
1	6.1	5.6	8.0%	6.8%	$\alpha_{seg} = 0.231$
3	6.0	5.6	8.0%	6.8%	$\alpha_{seg1} = 0.412$, $\alpha_{seg2} = 0.248$

Table 4.13: Accuracy of candidate models for wire segmentation length

Note that although Candidate Model 2 and 4 are not fitted to data, their accuracy are measured for training and validation set circuits separately to allow a fair comparison of validation RMSE with other candidate models.

The best candidate model is selected based on a balance between simplicity, intuition and accuracy. Candidate Model 5 is significantly worse than the rest, showing that multiplying a power of L with the switching matrix flexibility penalty term is not as a good model as having an additive segment length penalty term (as in the four other models). Thus Candidate Model 5 is ruled out. Candidate Model 3, while more complex (requiring two fitting parameters), is not

significantly better than Candidate Model 2; this reflects that modeling the slight dependence on Fs in Figure 4.16 is not worth the added complexity. Thus Candidate Model 3 can not be the best choice.

The choice of best model comes down to Candidate Models 1, 2 and 4. Although Candidate Model 1 is more accurate than 2, we believe the gap is not significant enough to justify the added complexity of a fitting parameter for the exponent of Fc_{in}. Candidate Model 2 is simpler than 1 because the exponent of Fc_{in} is chosen same as $\alpha_{in} = 0.5$ relating Fc_{in} flexibility and W_{need}. Candidate Model 4 is worse than 2 in accuracy but not significantly. However, we expect the accuracy gap between them to increase as they are generalized for less flexible architectures, because Candidate Model 4 is missing the Fc_{in} reduction effect which will be a more prominent cause of increase in W_{need} when logical equivalence is reduced (later in Section 4.6).

In conclusion, based on a balance between simplicity, intuition and accuracy, Candidate Model 2 is the best model for long wire segment architectures (architectures listed in Table 4.9). We shall refer to this model as the segmentation model, modeling L, Fs, Fc_{in} and Fc_{out}.

The segmentation model accuracy is shown in Figures 4.17 and 4.18. For clarity, only results for the largest circuit in each training (*clma*) and validation (*pdc*) sets are shown (remember the model is not fitted on either), under a subset of architectures. In Figures 4.17 and 4.18, y-axis shows W_{need}, for the architecture $Fs = 3$, 6 and 9, $Fc_{in} = 8$, $Fc_{out} = 8$, and $L = 1$, 2, 4, 6 and 8 (x-axis). The measured values are in solid dots, and model predicted values are in hollow dots. The breakdown of the accuracy shown in Figures 4.17 and 4.18 are presented in Tables 4.14 and 4.15 respectively. Each table consists of four sub-tables; top two sub-tables give W_{need} at various Fs (row) and L (column) values, for the measured and predicted; the bottom two sub-tables give the error (predicted − measured) and error as a percentage of measured. The second last row shows the average absolute error percentage for each wire segment length L (averaged for each column). The final row gives the average absolute error percentage across all data points in table.

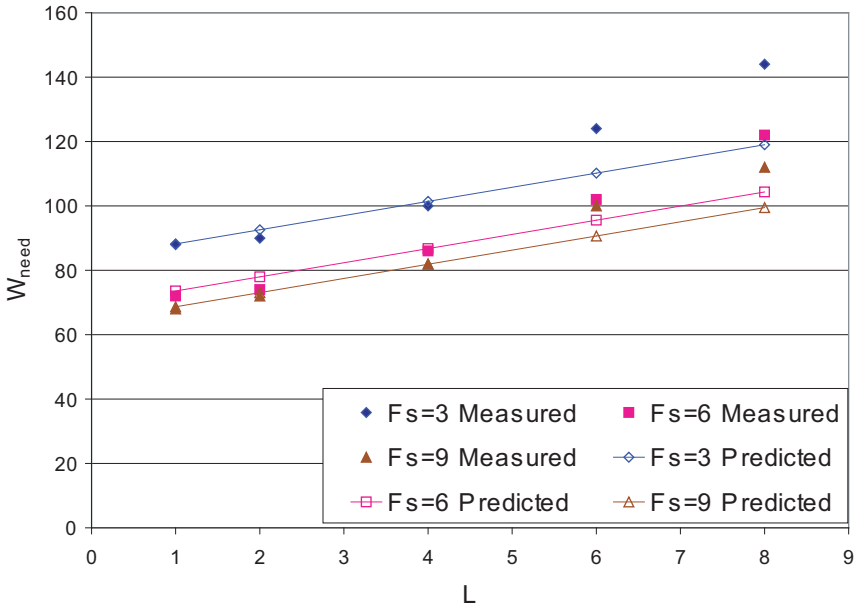


Figure 4.17: Accuracy of segmentation model for training circuit *clma*

	Measured					Predicted				
Fs/L	1	2	4	6	8	1	2	4	6	8
3	88	90	100	124	144	88	93	101	110	119
6	72	74	86	102	122	74	78	87	96	104
9	68	72	82	100	112	69	73	82	91	99
	Error					Error %				
Fs/L	1	2	4	6	8	1	2	4	6	8
3	0	3	1	-14	-25	0.2%	2.9%	1.4%	-11%	-17%
6	2	4	1	-6	-18	2.2%	5.3%	0.9%	-6.3%	-14%
9	1	1	0	-9	-13	1.0%	1.5%	-0.1%	-9.3%	-11%
	Average Absolute Error per L					*1.1%*	*3.2%*	*0.8%*	*8.9%*	*14%*
	Average Absolute Error Overall					**5.7%**				

Table 4.14: Breakdown of segmentation model accuracy for training circuit *clma*

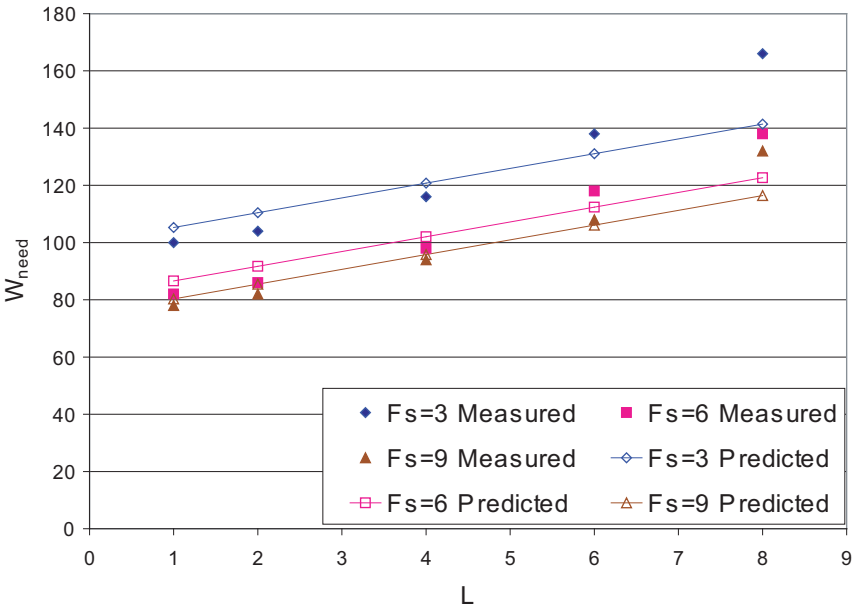


Figure 4.18: Accuracy of segmentation model for validation circuit *pdc*

	Measured					Predicted				
Fs/L	1	2	4	6	8	1	2	4	6	8
3	100	104	116	138	166	105	110	121	131	141
6	82	86	98	118	138	87	92	102	112	123
9	78	82	94	108	132	80	85	96	106	116
	Error					Error %				
Fs/L	1	2	4	6	8	1	2	4	6	8
3	5	6	5	-7	-25	5.3%	6.2%	4.1%	-5.0%	-15%
6	5	6	4	-6	-15	5.6%	6.7%	4.1%	-4.8%	-11%
9	2	3	2	-2	-16	3.0%	4.2%	1.9%	-1.7%	-12%
	Average Absolute Error per L					*4.6%*	*5.7%*	*3.4%*	*3.8%*	*13%*
	Average Absolute Error Overall					**6.0%**				

Table 4.15: Breakdown of segmentation model accuracy for validation circuit *pdc*

4.5.5 Simplicity-Driven Model

Before going on to extend the segmentation model in Equation 4.5.3.4 for more inflexible architectures, we will evaluate the symmetric simple model. In Section 4.4.3 the symmetric simple model, repeated here in Equation 4.5.5.1, was found to be worse than the the switching matrix model (in Equation 4.4.2.2), for a balance of simplicity, intuition and accuracy. Although the symmetric simple model is simpler than the switching matrix model (one less fitting parameter), we hypothesized that the accuracy gap between them will increase for more inflexible architectures, because the former does not model the important trend that Fc_{in} has a greater impact on W_{need} than Fc_{out}. In this subsection, this hypothesis will be tested by extending the symmetric simple model to model L.

$$W_{need}(Fs, Fc_{in}, Fc_{out}) = W_{abs\_min} + \frac{\alpha}{\beta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)\left(\frac{W_{abs\_min}}{Fc_{out}}\right) \tag{4.5.5.1}$$

The procedure of formulating candidate models derived from the symmetric simple model is identical to Section 4.5.3. The candidate models are:

Candidate Model 1:

$$\begin{aligned} W_{need}(Fs, Fc_{in}, Fc_{out}, L) = W_{abs\_min} &+ \frac{\alpha}{\beta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)\left(\frac{W_{abs\_min}}{Fc_{out}}\right) \\ &+ \frac{\lambda(L-1)}{4}\left(1 + \frac{1}{{Fc_{in}}^{\alpha_{seg}}}\right) \end{aligned} \tag{4.5.5.2}$$

where α and α_{seg} are fitting parameters, to be determined by training data. The parameter β is fixed at 3 (same statement holds for all following candidate models).

Candidate Model 2:

$$\begin{aligned} W_{need}(Fs, Fc_{in}, Fc_{out}, L) = W_{abs\_min} &+ \frac{\alpha}{\beta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)\left(\frac{W_{abs\_min}}{Fc_{out}}\right) \\ &+ \frac{\lambda(L-1)}{4}\left(1 + \frac{1}{Fc_{in}}\right) \end{aligned} \tag{4.5.5.3}$$

where α is the only fitting parameter. Note the exponent of Fc_{in} is adjusted to 1 for simplicity, similarly to before (adjusted to α_{in}).

Candidate Model 3:

$$W_{need}(Fs, Fc_{in}, Fc_{out}, L) = W_{abs\_min} + \frac{\alpha}{\beta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)\left(\frac{W_{abs\_min}}{Fc_{out}}\right) + \frac{\lambda(L-1)}{4}\left(1 + \frac{1}{{Fc_{in}}^{\alpha_{seg1}} {Fs}^{\alpha_{seg2}}}\right) \tag{4.5.5.4}$$

where α, α_{seg1} and α_{seg2} are all fitting parameters.

Candidate Model 4:

$$W_{need}(Fs, Fc_{in}, Fc_{out}, L) = W_{abs\_min} + \frac{\alpha}{\beta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)\left(\frac{W_{abs\_min}}{Fc_{out}}\right) + \frac{\lambda(L-1)}{4} \tag{4.5.5.5}$$

where α is the only fitting parameter.

Candidate Model 5:

$$W_{need}(Fs, Fc_{in}, Fc_{out}, L) = W_{abs\_min} + \frac{\alpha}{\beta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)\left(\frac{W_{abs\_min}}{Fc_{out}}\right)(L)^{\alpha_L} \tag{4.5.5.6}$$

where α and α_L are fitting parameters.

By the same model fitting approach as Section 4.5.4, we obtain results in Table 4.16.

Model No.	RMSE Training	RMSE Validation	MAPE Training	MAPE Validation	Fitted Parameters
5	20	19	24%	20%	$\alpha = 0.00319$, $\alpha_L = 2.18$
4	13	13	12%	10%	$\alpha = 0.0609$
2	12	12	11%	9.8%	$\alpha = 0.0554$
1	11	11	10%	9.5%	$\alpha = 0.0375$, $\alpha_{seg} = 0.0967$
3	11	11	10%	9.6%	$\alpha = 0.0356$, $\alpha_{seg1} = 0.185$, $\alpha_{seg2} = 0.135$

Table 4.16: Accuracy of symmetric simple candidate models

Comparing Table 4.16 and Table 4.13, it is clear that all candidate models derived from the symmetric simple model are worse in accuracy than the segmentation model in Equation 4.5.3.4 (extended from the switching matrix model): The validation RMSE of all candidate models are at least 1.8 times that of the segmentation model (recall validation MAPE of segmentation model is 6.2) – they are worse by approximately twice the number of tracks mis-predicted.

The validation MAPE of all candidate models are significantly worse as well, at least 1.5 times greater (recall validation MAPE of segmentation model is 6.5%).

The hypothesis that the accuracy gap between symmetric simple model and the switching matrix model will increase for more inflexible architectures is verified. The accuracy gap between them is 6.9 / 4.6 = 1.5 times worse validation RMSE, and 8.6% / 6.5% = 1.3 times worse validation MAPE (numbers from Table 4.8 in Section 4.4.3). The accuracy gap between them, extended for L, is 1.8 times in validation RMSE and 1.5 times in validation MAPE.

This study confirms that for the balance of simplicity, intuition and accuracy, the segmentation model is the best. However, we believe the symmetric simple model has compelling value in the theme of simplicity. The value of such a simple model lays in the ease with which it can be analytically manipulated to express tradeoff equations. For this reason, we will develop a second routing demand model that is more simplicity-driven, by extending the symmetric simple model. It will be referred to as the *simplicity-driven model*.

For simplicity-driven model, the best choice is Candidate Model 4, which can be re-arranged as:

Simplicity-Driven Model:

$$W_{need}(Fs, Fc_{in}, Fc_{out}, L) = W_{abs\_min} + \frac{1}{\theta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)\left(\frac{W_{abs\_min}}{Fc_{out}}\right) + \frac{\lambda(L-1)}{4} \tag{4.5.5.7}$$

where $\frac{\alpha}{\beta}$ is replaced by $\frac{1}{\theta}$, and θ equals 50. Its accuracy is a validation RMSE of 13, and a validation MAPE of 10% (as given in Table 4.16). In modeling for L, this model does not account for the Fc_{in} reduction effect, for simplicity. In the following section of modeling logical equivalence, this simplicity-driven model will be further developed, alongside the main routing model. Before moving onto the modeling of logical equivalence, we will review the main routing model and present the insights and intuitions it gives.

4.5.6 Segmentation Model Intuition

The routing model developed so far presents the following insights and intuitions. The routing demand W_{need} has three additive components:

$$W_{need} = \begin{matrix} absolute \\ minimum \end{matrix} + \begin{matrix} switching\ matrix \\ flexibility \\ penalty \end{matrix} + \begin{matrix} segment \\ length \\ penalty \end{matrix} \tag{4.5.6.1}$$

The first component "*absolute minimum*" is W_{abs_min} needed for a fully flexible FPGA. A "*switching matrix flexibility penalty*" must be added if the switching matrix described by Fs, Fc_{in} and Fc_{out} is reduced from its maximum flexibility. In detail this penalty equals:

$$switching\ matrix\ flexibility\ penalty = \frac{1}{\beta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)^{\alpha_{in}}\left(\frac{W_{abs\_min}}{Fc_{out}}\right)^{\alpha_{out}} \tag{4.5.6.2}$$

where $\beta = 3$, $\alpha_{in} = 0.5$ and $\alpha_{out} = 0.25$.

Furthermore a "*segment length penalty*" must be paid if the wire segment length L of the routing architecture is greater than 1:

$$segment\ length\ penalty = \frac{\lambda(L-1)}{4}\left(1 + \frac{1}{{Fc_{in}}^{\alpha_{in}}}\right) \tag{4.5.6.3}$$

This penalty has two components. Primarily, increased segment length L causes increase in W_{need} due to segmentation waste, by the amount $\frac{\lambda(L-1)}{4}$. Second order effects of segmentation waste causes an effective reduction in Fc_{in} flexibility, captured by the $\left(1 + \frac{1}{{Fc_{in}}^{\alpha_{in}}}\right)$ term, and this effective reduction is greater for greater segmentation waste – thus it is multiplied with the amount of segmentation waste $\frac{\lambda(L-1)}{4}$.

Further inflexibility in the routing architecture arises when the input pins and output pins are no longer logically equivalent as assumed so far. The following subsection models the impact of logical equivalence.

4.6 Impact of Logical Equivalence

The routing model developed up to this point models architectures in which all input pins on the same logic block are logically equivalent and all output pins on the same logic block are

logically equivalent. Recall from Chapter 2 that pins are evenly distributed on the four sides of a square logic block in the manner shown in Figure 2.3, which is repeated here in Figure 4.19. Having logical equivalence (parameter $Eqv = 1$) allows the router to take advantage of the physical location of pins – an incoming connection has the choice of connecting to any unused input pin on the four sides (similarly for output pins). On the other hand, removing logical equivalence between pins (so that all pins are distinct) introduces an inflexibility in routing, in turn causing an increase in W_{need}. The logical equivalence parameter Eqv is binary and we incorporate it into the routing model by capturing the effects of removing logical equivalence ($Eqv = 0$). We leave it to future work to model less binary gradations of logical equivalence. The architectures to be modeled in this section are summarized in Table 4.17.

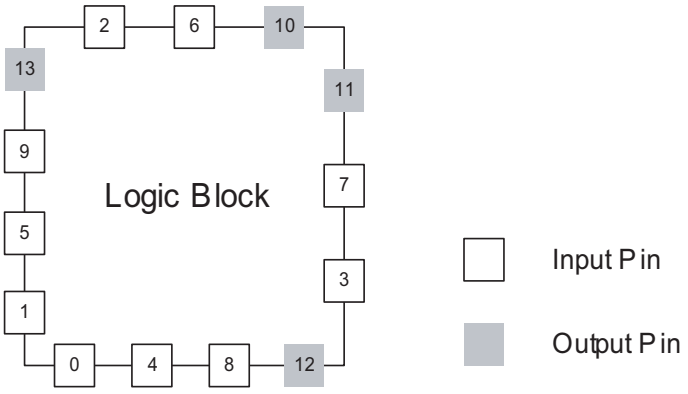


Figure 4.19: Distribution of pins on the logic block for logic cluster size 10

There are two main effects of removing logical equivalence. One is the distance each connection must travel is increased, because it may have to travel to the far side of a logic block, rather than connecting to the nearest. A second effect will be to reduce the effective connection block flexibility, because the number of opportunities to connect in and out of the logic block is dramatically reduced. The routing model is extended to capture these two effects of removing logic equivalence in turn in the following subsections.

Fs	Fc_{in}	Fc_{out}	L	Eqv
$[3, 21]$	[2,W_{abs_min}]	[2,W_{abs_min}]	[1,8]	0

Table 4.17: FPGA routing architectures without logical equivalence

4.6.1 Increased Routing Distance

The first effect of removing logical equivalence, $Eqv = 0$, is that the distance connections must travel is increased. This causes an increase in routed wire length, and in turn W_{need}. To determine the distance increase, we isolate it by assuming a fully flexible architecture but with $Eqv = 0$, in Table 4.18. In such an architecture, connections are not impacted by lack of flexibility in other routing parameters, thus are made by near minimum distance routes which reflect distance increase.

Using the W_{need} search experimental flow in Section 3.4, under the architecture described in Table 4.18, we place and route training circuits. Using the routing solution at W_{need}, the post-routing wire length is parsed out. The total routed wire length increase, expressed as the ratio:

$$\frac{\text{Average Wire Length for } Eqv = 0}{\text{Average Wire Length for } Eqv = 1} \tag{4.6.1.1}$$

is shown in Figure 4.20. In Figure 4.20, the wire length increase ratio (y-axis) of each training circuit is plotted against the W_{abs_min} (x-axis) of the circuit. The average ratio across all training circuits is 1.166 with standard deviation 0.036. The near-constant plot and low standard deviation suggest that routing distance increase due to removing logical equivalence causes consistent increase in routed wire length.

Motivated by the consistency we model this effect in the input parameter, $\overline{R}$, the placement-estimated average two-pin wire length (which uses a Minimum Spanning Tree two-pin net model). Note the placement estimation of $\overline{R}$ does not consider logic block boundaries, hence naturally assumes logical equivalence. The model for removed logical equivalence is:

$$\overline{R}_{NE} = \sigma * \overline{R} \tag{4.6.1.2}$$

where $\sigma = 1.166$. The term $\overline{R}_{NE}$ is the average two-pin wire length when the routing architec-

Fs	Fc_{in}	Fc_{out}	L	Eqv
$3W$	W	W	1	0

Table 4.18: Fully flexible FPGA routing architecture except no logical equivalence

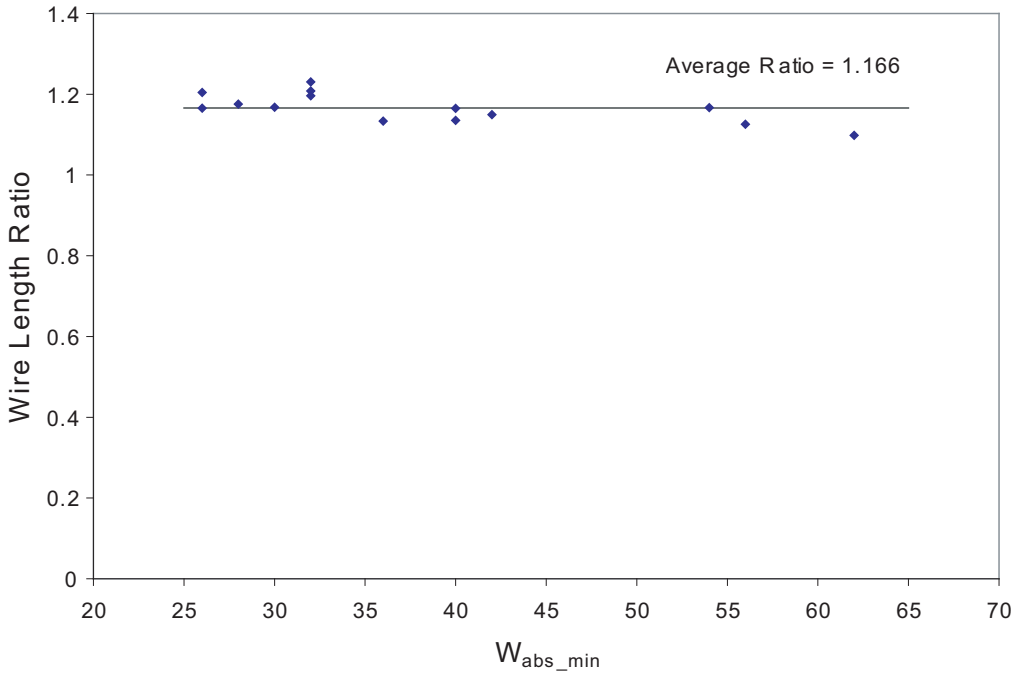


Figure 4.20: Routed wire length increase as a result of removing logical equivalence

ture has no logical equivalence.

To check the validity of this model, we substitute it into the fully flexible FPGA routing model and compare its predicted W_{need} to actual measured W_{need} under the $Eqv = 0$ architecture in Table 4.18. We shall refer to W_{need} under this architecture as W_{abs_minNE} – the absolute minimum routing demand for any routing architecture without logical equivalence. The model after substitution is:

$$W_{abs\_minNE} = p\frac{\lambda \overline{R}_{NE}}{2} = \sigma * p\frac{\lambda \overline{R}}{2} \tag{4.6.1.3}$$

where p remains fixed at previously determined value of 1.4.

The accuracy of the fully flexible FPGA routing model for $Eqv = 0$, Equation 4.6.1.3, is 3.4 RMSE and 6.1% MAPE for training circuits and 3.2 RMSE and 5.5% MAPE for validation circuits, 14 data points each. The accuracy is displayed in Figure 4.21, where W_{need} (y-axis) is plotted for all circuits ordered by their W_{avg} (x-axis), and the straight line is the fully flexible model for $Eqv = 0$ (slope is again 1.4).

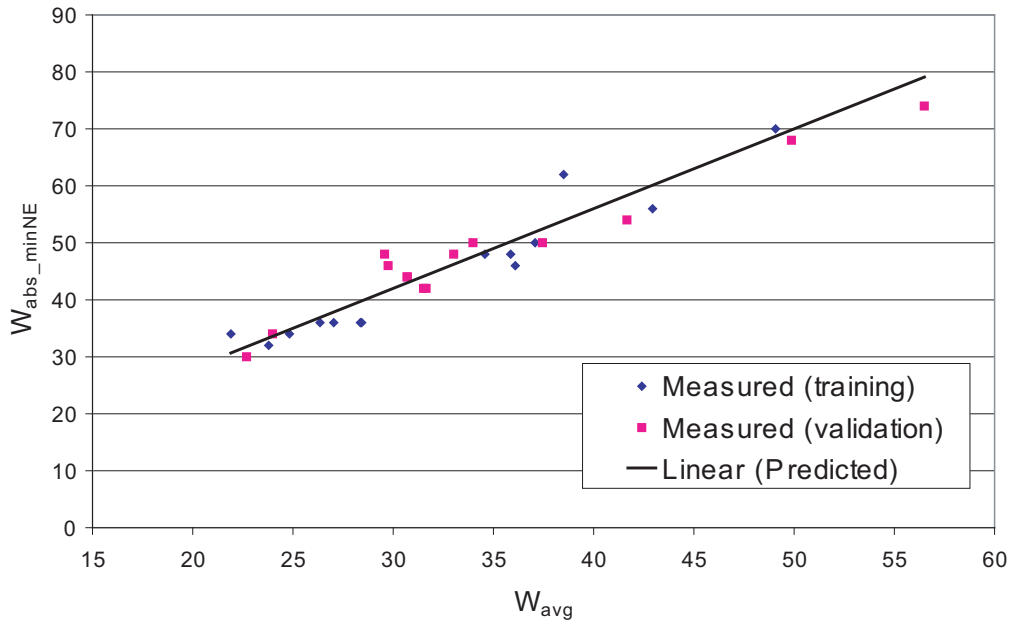


Figure 4.21: Accuracy of fully flexible but $Eqv = 0$ FPGA model for all benchmark circuits

The accuracy shows that the model in Equation 4.6.1.2 captures well the increased routing distance. Routing models developed for $Eqv = 1$ can account for the routing distance increase due to removing logical equivalence ($Eqv = 0$) by replacing $\overline{R}$ with $\overline{R}_{NE}$.

In addition to increased routing distance, removing logical equivalence has the effect of reducing connection block flexibility, which in turn causes further increase in W_{need}. This effect is modeled in the following subsection.

4.6.2 Effective Connection Block Flexibility Reduction

Without logical equivalence the choices of making a connection in the connection block is effectively reduced. This can be best seen by an example, illustrated in Figure 4.22. Figure 4.22 depicts example architectures in a) and b), both having $W = 5$ and $Fc_{in} = 2$, but a) has logical equivalence and b) does not. In Figure 4.22 a), having logical equivalence enables a connection going to input pin 0 to choose between 18 (in blue) out of the 20 total wires surrounding the logic block (since it can connect to any of the ten input pins). On the other

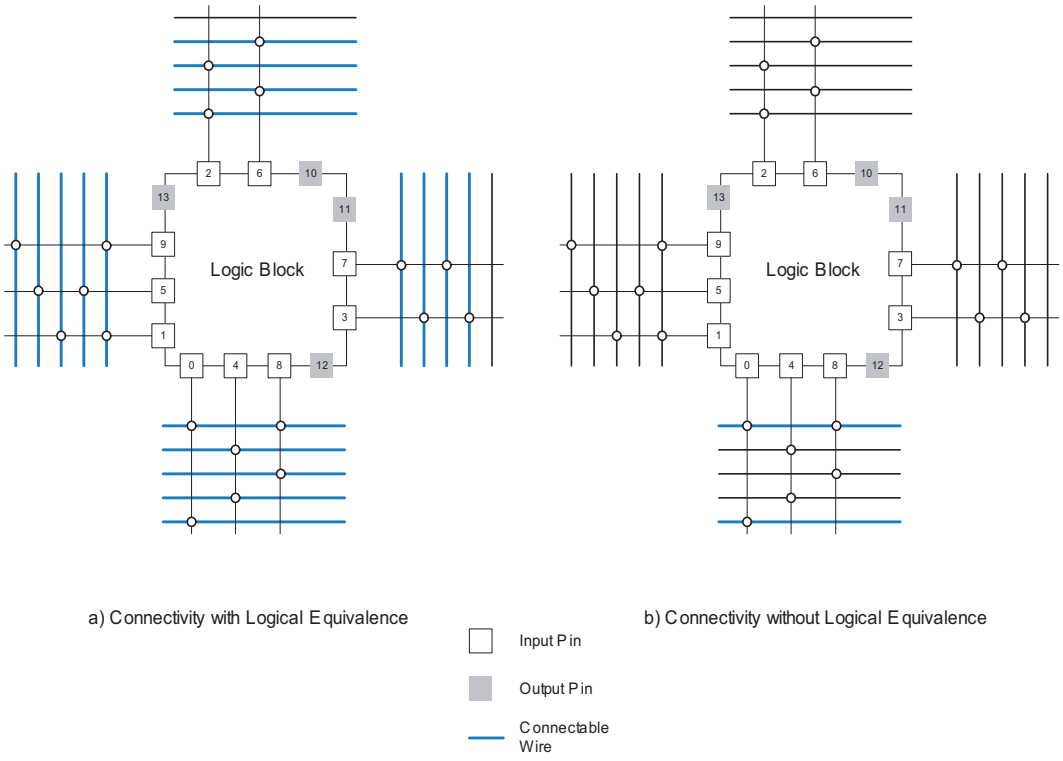


Figure 4.22: Removing logical equivalence reduces the number of choices in connection blocks

hand in Figure 4.22 b), in the absence of logical equivalence, the same connection going to input pin 0 must be made through only two wires. This example shows the number of choices in the connection block is reduced when logical equivalence is removed; in effect, the connection block flexibility is reduced.

This effective connection block flexibility reduction in turn causes an increase in W_{need}. This can be seen empirically in Figure 4.23. The data in Figure 4.23 is obtained using the W_{need} search experimental flow, under the architecture $Fs = 6$, $Fc_{in} = 8$, $Fc_{out} = 8$ and $L = 1$, using training set circuits. The figure shows the average W_{need}, across training circuits, when logical equivalence is removed ($W_{need}(Eqv = 0)$, y-axis) plotted against W_{need} when logical equivalence exists ($W_{need}(Eqv = 1)$, x-axis). The average ratio of $\frac{W_{need}(Eqv=0)}{W_{need}(Eqv=1)}$ is 1.327. While approximately a factor of 1.166 of the increase in W_{need} is accounted for by routing distance increase, the remaining increase in W_{need} is explained by the effective connection block flexibility reduction.

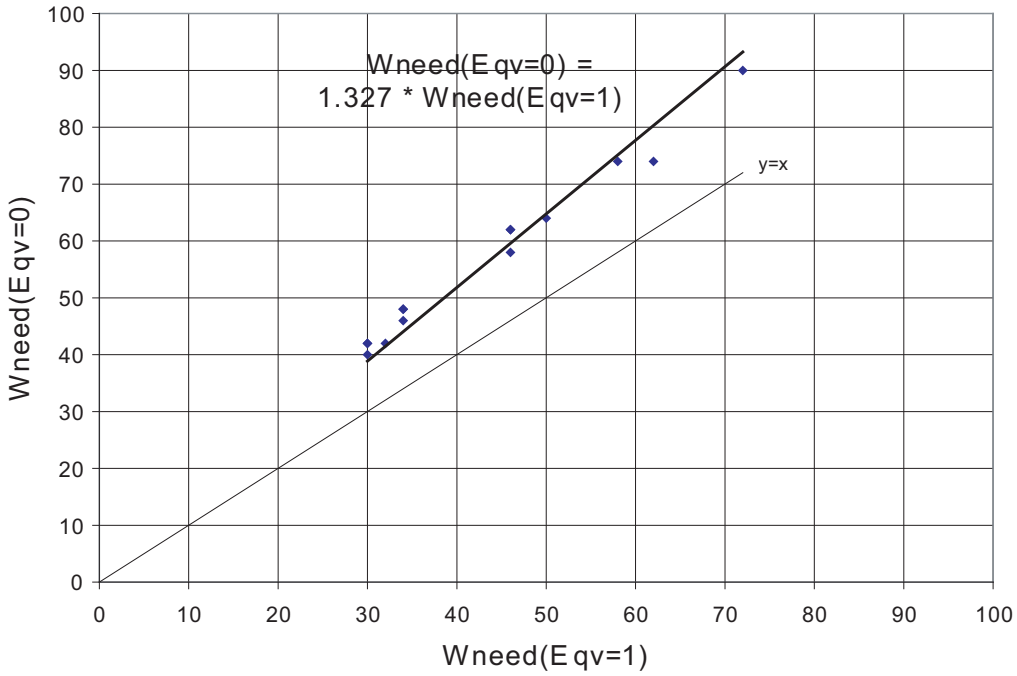


Figure 4.23: Average W_{need} of training circuits for with vs. without logical equivalence

The output connection block flexibility, Fc_{out}, can be reduced, but its effect is small that it is ignored for model simplicity. Therefore we will focus on modeling effective connection block flexibility reduction on the input pins only, reducing Fc_{in}.

To model effective input connection block flexibility reduction, we observe that, when logical equivalence exists for the I input pins on the logic block, each one of those pins contributes to the effective Fc_{in}. As one candidate model the loss of those I pins, due to removing logical equivalence, reduces the effective Fc_{in} by a constant amount each. That is we replace Fc_{in} by the term $FcNE_{in}$ (effective Fc_{in} for no logical equivalence) defined as follows:

$$FcNE_{in} = \frac{Fc_{in}}{I\mu} \tag{4.6.2.1}$$

to arrive at

Candidate Model 1:

$$W_{need}(Fs, Fc_{in}, Fc_{out}, L, Eqv = 0) = W_{abs\_minNE} + \frac{1}{\beta}\left(\frac{W_{abs\_minNE}}{Fs}\right)\left(\frac{W_{abs\_minNE}}{FcNE_{in}}\right)^{\alpha_{in}}\left(\frac{W_{abs\_minNE}}{Fc_{out}}\right)^{\alpha_{out}} + \frac{\lambda(L-1)}{4}\left(1 + \frac{1}{FcNE_{in}{}^{\alpha_{in}}}\right) \tag{4.6.2.2}$$

where β, α_{in} and α_{out} are fixed at their previously determined values of 3, 0.5 and 0.25 respectively, and μ is a fitting parameter. The term W_{abs_minNE} is defined in Equation 4.6.1.3.

Alternatively, $FcNE_{in}$ can be modeled as a power function of the nominal Fc_{in}, as:

$$FcNE_{in} = Fc_{in}{}^{\frac{1}{I\mu}} \tag{4.6.2.3}$$

to obtain Candidate Model 2.

As a more complex variant of Candidate Model 1, the effective Fc_{in} reduction need not be the same in switching matrix flexibility penalty and segmentation penalty. This is motivated by the possibility of interaction between Fc_{in} reduction due to segmentation waste and Fc_{in} reduction due to removing logical equivalence. The proposed model is:

Candidate Model 3:

$$W_{need}(Fs, Fc_{in}, Fc_{out}, L, Eqv = 0) = W_{abs\_minNE} + \frac{1}{\beta}\left(\frac{W_{abs\_minNE}}{Fs}\right)\left(\frac{W_{abs\_minNE}}{\frac{Fc_{in}}{I\mu_1}}\right)^{\alpha_{in}}\left(\frac{W_{abs\_minNE}}{Fc_{out}}\right)^{\alpha_{out}} + \frac{\lambda(L-1)}{4}\left(1 + \frac{1}{\frac{Fc_{in}}{I\mu_2}}^{\alpha_{in}}\right) \tag{4.6.2.4}$$

where all parameters are fixed at previously determined values, except μ_1 and μ_2 are fitting parameters.

These three candidate models capturing the impact of removing logical equivalence are evaluated and compared in the following subsection.

4.6.3 Model Accuracy

Data are generated by the W_{need} search experimental flow in Section 3.4, under architectures in Table 4.17, using training and validation circuits. Candidate models are fitted to training

data. The model accuracies are summarized in Table 4.19. In Table 4.19, the first column identifies the candidate model, the next four columns are the accuracy metrics, and the final column gives the fitted parameter values. The rows are sorted by descending validation RMSE (column 3). There are 14552 training data points and 13834 validation data points.

Model No.	RMSE Training	RMSE Validation	MAPE Training	MAPE Validation	Fitted Parameters
1	18	15	15%	14%	$\mu = 0.330$
2	17	15	15%	14%	$\mu = 0.225$
3	14	11	11%	9.1%	$\mu_1 = 0.1$, $\mu_2 = 2.77$

Table 4.19: Accuracy of candidate models for removed logical equivalence

The best candidate model, based on a balance of simplicity, intuition and accuracy, is Candidate Model 1. It contains one fewer parameter than Candidate Model 3, at the loss of 30-36% in RMSE, an amount that can not justify the added complexity of 3. Candidate Model 2 is not better in accuracy than 1, and its form – a power function – is more complex to work with than the ratio in Equation 4.6.2.1. In conclusion, Candidate Model 1 is the best.

The value of μ that gives the final model best fit to training data is 0.33. Due to the large volume of data, only results for the largest circuit in the each training (*clma*) and validation (*pdc*) sets, under a selected set of architectures, are shown in Figures 4.24 and 4.25, respectively. In Figures 4.24 and 4.25, W_{need} for $Eqv = 0$ (y-axis) is plotted against Fc_{in} (x-axis). There are three sub-figures in each, showing for architectures of different L values: 1, 4 and 6. In each sub-figure, are plots for $Fs = 3$ and 6 of the measured W_{need} (solid dots) and the model predicted W_{need} (hollow dots). The architecture parameter Fc_{out} is set to 24 for all, except when $L = 6$ in which case $Fc_{out} = 18$[3]

The general trend in Figures 4.24 and 4.25, which is typical, is that the final routing demand model (for $Eqv = 0$) over-predicts for low wire segment lengths (particularly for $L = 1$) and improves for larger lengths.

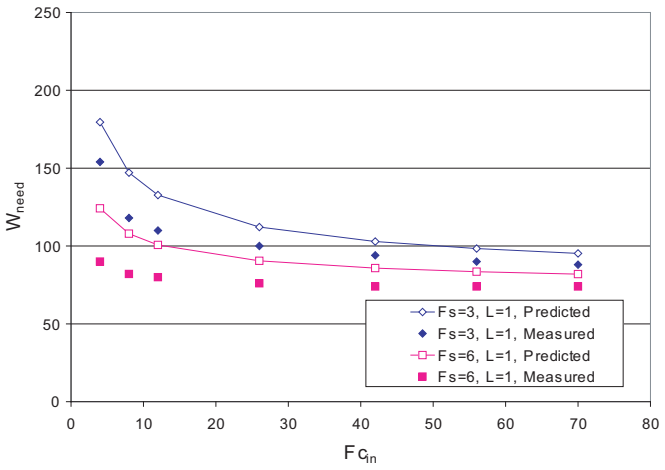
[3]Choice of Fc_{out} for experiments is dependent on L, since L controls the number of distinct starting wires (which output pins can connect) adjacent an output pin. In this case, there was no $Fc_{out} = 24$, $L = 6$ architecture in the experiment set.

Numerical breakdown of the accuracy is shown in Tables 4.20 and 4.21. Each table consists of four sub-tables; top two sub-tables give W_{need} at various architecture, consisting of Fs and L (row) and Fc_{in} (column) values, for the measured and predicted; the bottom two sub-tables give the error (predicted − measured) and error as a percentage of measured. The second last row shows the absolute error percentage averaged for each column. The final row gives the average absolute error percentage across all data points in table.

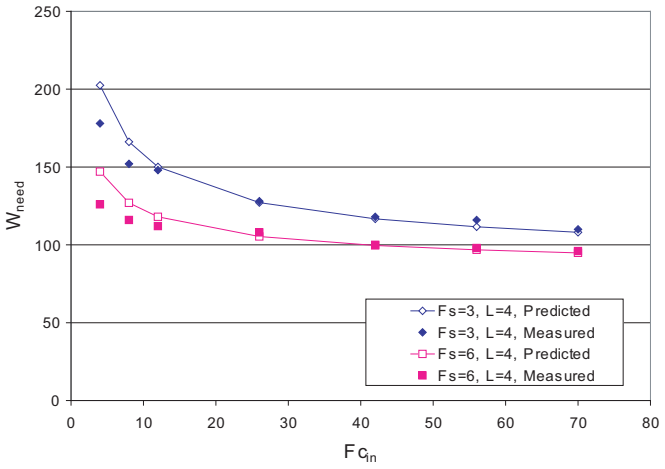
The final routing demand model is complete and it will be summarized, with intuitions given, in Section 4.8.

	Measured						Predicted					
Fs	3	3	3	6	6	6	3	3	3	6	6	6
Fc_{in}/L	1	4	6	1	4	6	1	4	6	1	4	6
4	154	178	234	90	126	174	180	202	226	124	147	166
8	118	152	200	82	116	150	147	166	185	108	127	143
12	110	148	188	80	112	148	133	150	166	101	118	132
26	100	128	164	76	108	130	112	127	140	90	105	117
42	94	118	146	74	100	128	103	117	128	86	100	110
56	90	116	140	74	98	122	98	112	123	84	97	107
70	88	110	134	74	96	116	95	108	119	82	95	104
	Error						Error %					
Fs	3	3	3	6	6	6	3	3	3	6	6	6
Fc_{in}/L	1	4	6	1	4	6	1	4	6	1	4	6
4	26	24	-8	34	21	-8	17%	14%	-3.4%	38%	17%	-4.4%
8	29	14	-15	26	11	-7	25%	9.3%	-7.7%	32%	9.4%	-5.0%
12	23	2	-22	21	6	-16	21%	1.4%	-11%	26%	5.4%	-11%
26	12	-1	-24	14	-3	-13	12%	-0.7%	-14%	19%	-2.4%	-10%
42	9	-1	-18	12	0	-18	9.5%	-1.1%	-12%	16%	-0.4%	-14%
56	8	-4	-17	10	-1	-15	9.3%	-3.8%	-12%	13%	-1.2%	-13%
70	7	-2	-15	8	-1	-12	8.2%	-1.7%	-11%	11%	-1.2%	-10%
	Average Absolute Error per Column						*14%*	*4.5%*	*10%*	*22%*	*5.3%*	*9.5%*
	Average Absolute Error Overall						**11%**					

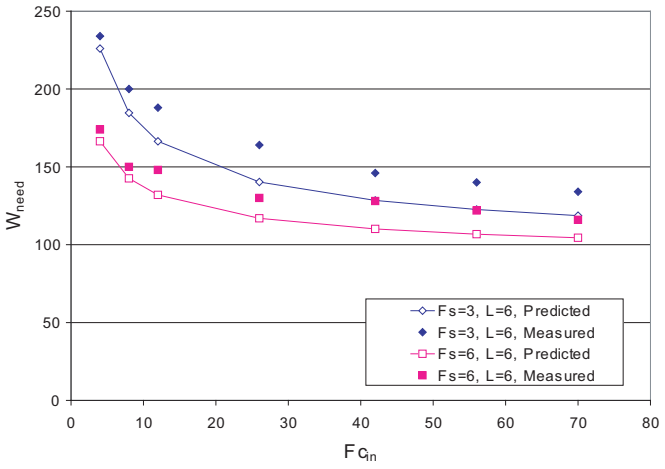
Table 4.20: Breakdown of final routing model accuracy for training circuit *clma*



(a) $L = 1$

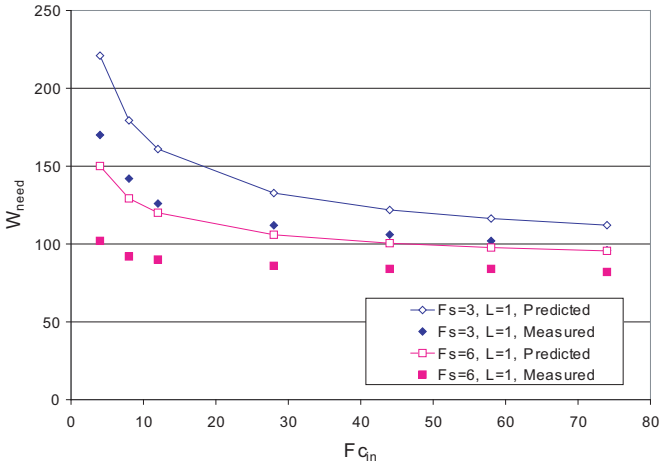


(b) $L = 4$

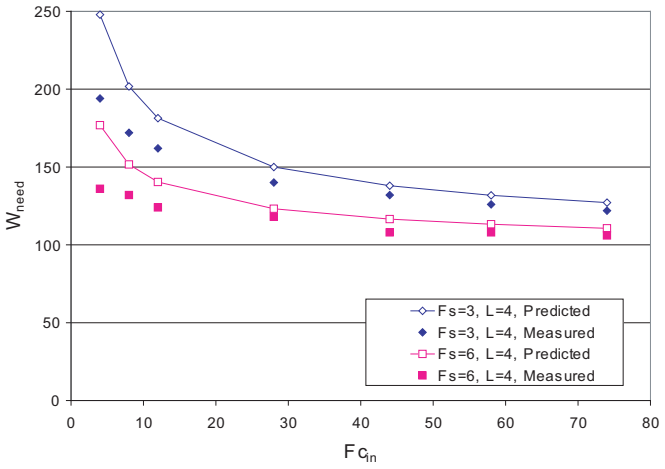


(c) $L = 6$

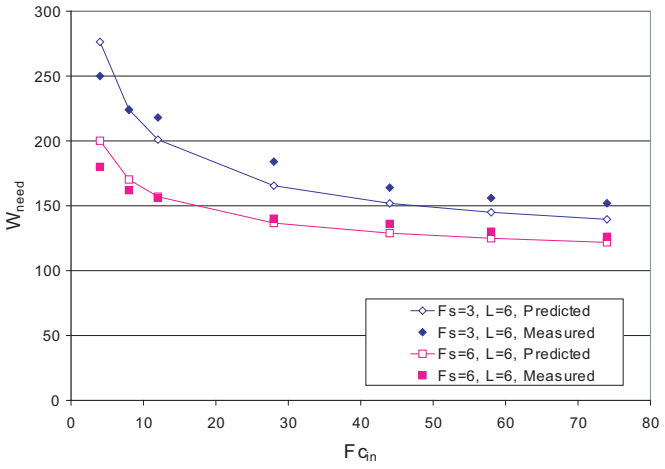
Figure 4.24: Accuracy of the final routing model for training circuit *clma*



(a) $L = 1$



(b) $L = 4$



(c) $L = 6$

Figure 4.25: Accuracy of the final routing model for validation circuit *pdc*

	Measured						Predicted					
Fs	3	3	3	6	6	6	3	3	3	6	6	6
Fc_{in}/L	1	4	6	1	4	6	1	4	6	1	4	6
4	170	194	250	102	136	180	221	248	276	150	177	200
8	142	172	224	92	132	162	179	202	224	129	152	170
12	126	162	218	90	124	156	161	181	201	120	140	157
26	112	140	184	86	118	140	133	150	165	106	123	137
42	106	132	164	84	108	136	122	138	152	100	117	129
56	102	126	156	84	108	130	116	132	145	98	113	125
70	96	122	152	82	106	126	112	127	140	96	111	122
	Error						Error %					
Fs	3	3	3	6	6	6	3	3	3	6	6	6
Fc_{in}/L	1	4	6	1	4	6	1	4	6	1	4	6
4	51	54	26	48	41	20	30%	28%	11%	47%	30%	11%
8	37	30	0	37	20	8	26%	17%	0.1%	41%	15%	5.1%
12	35	19	-17	30	16	1	28%	12%	-7.8%	33%	13%	0.6%
26	21	10	-19	20	5	-3	19%	7.1%	-10%	23%	4.4%	-2.4%
42	16	6	-12	16	9	-7	15%	4.5%	-7.4%	20%	7.9%	-5.2%
56	14	6	-11	14	5	-5	14%	4.6%	-7.1%	16%	4.8%	-3.9%
70	16	5	-12	14	5	-4	17%	4.2%	-8.2%	17%	4.3%	-3.3%
	Average Absolute Error per Column						*21%*	*11%*	*7%*	*28%*	*11%*	*4.5%*
	Average Absolute Error Overall						**14%**					

Table 4.21: Breakdown of final routing model accuracy for validation circuit *pdc*

4.6.4 Final Simplicity-Driven Model

The simplicity-driven model is extended to capture the impact of logical equivalence by the same approach as the the main routing model, because that approach leads to the simplest analytic form.

Recall from Section 4.5.5, for $Eqv = 1$ the simplicity- driven model is

Simplicity-Driven Model:

$$W_{need}(Fs, Fc_{in}, Fc_{out}, L, Eqv = 1) = W_{abs\_min} + \frac{1}{\theta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)\left(\frac{W_{abs\_min}}{Fc_{out}}\right) + \frac{\lambda(L-1)}{4} \tag{4.6.4.1}$$

with

$$W_{abs\_min} = p\frac{\lambda\overline{R}}{2} \tag{4.6.4.2}$$

where $p = 1.4$, and $\theta = 50$.

When $Eqv = 0$, $\overline{R}$ in Equation 4.6.4.2 is replaced by $\overline{R}_{NE}$, defined by:

$$\overline{R}_{NE} = \sigma * \overline{R} \tag{4.6.4.3}$$

where $\sigma = 1.166$. And Fc_{in} in Equation 4.6.4.1 is replaced by $FcNE_{in}$, defined by:

$$FcNE_{in} = \frac{Fc_{in}}{I\mu} \tag{4.6.4.4}$$

where I is the number of input pins from which logical equivalence is removed, and $\mu = 0.06$. The accuracy of the final simplicity-driven model is a training RMSE of 32 and validation RMSE of 26, and a training MAPE of 22% and validation MAPE of 18%.

Although the simplicity-driven model is much worse in accuracy than the main routing demand model, in terms of individual circuit prediction, it holds value in ease of use. This is demonstrated in the next chapter, in Section 5.2.

The main routing demand model is completely developed, but before concluding this chapter, it is important to scrutinize the training process of this model. In the next section, we study the effect of training set choice on the model.

4.7 Bias of Training Set Choice

In this section we study the impact of training set on the main routing demand model, to ensure that the model fitting parameter values are not biased to the choice of training circuits.

As described in Section 3.3.4, a systematic approach is adopted for selecting the training circuits (to which the model is fit to) and validation circuits, to ensure there is no bias: both sets are similar in circuit nature, measured by a number of statistical metrics (LUT count and the number of nets). However, to check that there is no bias in the final model fitting parameters to the choice of training circuits, we swap the original training and validation sets, in Table 3.2. With the new training and validation circuit sets, we step through the entire process of developing the routing demand model again, in exactly the same approach. The results are summarized in Table 4.23. The routing demand model trained on the original training set is summarized in Table 4.22 for comparison.

Each table show the five intermediate models in the first column. The next four columns give the accuracy metrics, and the final column gives the fitted parameter value.

Looking at the last column of Tables 4.22 and 4.23, the parameters when fitted to either training or validation sets have the very similar values. Since we make the choice to simplify the model by rounding the parameters, the fitting parameter values can be considered the same when fitted to either set. This verifies that the choice of training set does not matter, due to our careful systematic approach of training and validation circuit selections.

What differs between the model trained on original training set or validation set lays in the reported accuracy. The training and validation RMSE values, in columns 2 and 3 of tables, appear to have swapped positions between Tables 4.22 and 4.23 – this can be expected since the newly trained model is nearly the same as originally trained. The validation RMSE in Table 4.23 are higher than those in Table 4.22. This reflects the fact that although the two sets of circuits are similar in nature, the model performs poorer on original training circuits than validation circuits – this is simply statistical noise. This suggests – for reporting accuracy of the model in predicting individual circuit's W_{need} – it might be best to use the worse (highest) of training RMSE and validation RMSE. We shall always report for both training and validation

sets and leave it to the user's discretion.

	RMSE		MAPE		
Model	Training	Validation	Training	Validation	Fitted Parameters
fully flexible	3.3	2.8	6.2%	5.8%	$p = 1.4$
Fs	3.5	2.8	6.6%	5.9%	$\beta = 3$
Fc_{in} & Fc_{out}	3.9	4.6	7.6%	6.5%	$\alpha_{in} = 0.5$, $\alpha_{out} = 0.25$
L	6.7	6.2	7.9%	6.5%	no new fitted parameter
Eqv	18	15	15%	14%	$\sigma = 1.166$, $\mu = 0.33$

Table 4.22: Routing demand model trained on original training set of circuits

	RMSE		MAPE		
Model	Training	Validation	Training	Validation	Fitted Parameters
fully flexible	2.5	3.5	5.5%	8.0%	$p = 1.44$
Fs	2.7	3.7	5.9%	8.5%	$\beta = 3.05$
Fc_{in} & Fc_{out}	4.6	4.4	7.0%	9.8%	$\alpha_{in} = 0.455$, $\alpha_{out} = 0.293$
L	6.0	6.6	7.2%	9.2%	no new fitted parameter
Eqv	14	18	13%	15%	$\sigma = 1.153$, $\mu = 0.288$

Table 4.23: Routing demand model trained on original validation set of circuits

The trend in MAPE values is different, however. While the new training MAPE values (column 4 of Table 4.23) are approximately the same as the original validation MAPE values (column 5 of Table 4.22) – expected since they are results from the same set of circuits – the new validation MAPE values (column 5 of Table 4.23) are noticeably worse than original training MAPE values (column 4 of Table 4.22). This can be explained by the fact that the original training and validation sets, although similar in BLE count and two-terminal net count, have different routing demands. In Table 4.24, data from Section 4.2 are re-hashed to show that the routing demands of circuits in the original training set are greater than those of circuits in the original validation set. Since MAPE is calculated (in Equation 3.3.3.2) using routing demand as the base value, smaller base value leads to higher error percentage. The new validation

	Average W_{abs_min}	Standard Deviation W_{abs_min}
Original Training Set	38.3	11.6
Original Validation Set	43.1	11.9

Table 4.24: Comparison of average W_{abs_min} of original training and validation sets

MAPE values (column 5 of Table 4.23) are larger due to the fact that the slight increase in error, from the model being trained on the other set, is accentuated by the small base values of new validation set (original training set). This suggests MAPE is a easy to comprehend metric, but is sensitive to training and validation set selection, unless circuits in both sets have the approximately the same routing demand.

4.8 Summary

In this chapter, we described the development of an FPGA interconnect model. It was developed by a guided empirical modeling approach, in combination with some fundamental derivations, for a balance of simplicity, intuition and accuracy. The final model predicts the routing demand, W_{need}, of island-style single-driver routing architectures parameterized by Fs, Fc_{in}, Fc_{out}, L and Eqv, to successful route circuits mapped to a logic block architecture described by, I, λ and $\overline{R}$. For architectures with logical equivalence between pins, $Eqv = 1$, the model is:

$$\begin{aligned} W_{need} = & W_{abs\_min} \\ & + \frac{1}{\beta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)^{\alpha_{in}}\left(\frac{W_{abs\_min}}{Fc_{out}}\right)^{\alpha_{out}} \\ & + \frac{\lambda(L-1)}{4}\left(1+\frac{1}{{Fc_{in}}^{\alpha_{in}}}\right) \end{aligned} \tag{4.8.0.5}$$

with

$$W_{abs\_min} = p\frac{\lambda\overline{R}}{2} \tag{4.8.0.6}$$

where, $p = 1.4$, $\beta = 3$, $\alpha_{in} = 0.5$, $\alpha_{out} = 0.25$.

When $Eqv = 0$, the model has two changes. First, the distance between each connection's source and sink increases when logical equivalence is removed, which is reflected in the model by replacing the average routing length, $\overline{R}$ in Equation 4.8.0.6, by $\overline{R}_{NE}$ defined by:

$$\overline{R}_{NE} = \sigma * \overline{R} \tag{4.8.0.7}$$

where $\sigma = 1.166$.

The second change is Fc_{in} in Equation 4.8.0.5 should be replaced by $FcNE_{in}$, defined by:

$$FcNE_{in} = \frac{Fc_{in}}{I\mu} \tag{4.8.0.8}$$

where I is the number of pins from which logical equivalence is removed, and $\mu = 0.33$. This reflects the effective loss of connection block flexibility.

On average the model is off by 6.2 tracks, which is 6.5% of actual W_{need}, in validation for $Eqv = 1$. For $Eqv = 0$, the model is on average 15 tracks off, which is 14% of actual W_{need}, in validation.

The routing demand model holds the following intuition. The routing demand has three additive components:

$$W_{need} = \begin{matrix} absolute \\ minimum \end{matrix} + \begin{matrix} switching\ matrix \\ flexibility \\ penalty \end{matrix} + \begin{matrix} segment \\ length \\ penalty \end{matrix} \tag{4.8.0.9}$$

The first component "*absolute minimum*" is W_{abs_min} needed for a fully flexible FPGA. A "*switching matrix flexibility penalty*" must be added if the switching matrix described by Fs, Fc_{in} and Fc_{out} is reduced from its maximum flexibility:

$$switching\ matrix\ flexibility\ penalty = \frac{1}{\beta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)^{\alpha_{in}}\left(\frac{W_{abs\_min}}{Fc_{out}}\right)^{\alpha_{out}} \tag{4.8.0.10}$$

Furthermore a "*segment length penalty*" must be paid if the wire segment length L of the routing architecture is greater than 1:

$$segment\ length\ penalty = \frac{\lambda(L-1)}{4}\left(1 + \frac{1}{{Fc_{in}}^{\alpha_{in}}}\right) \tag{4.8.0.11}$$

Parallel to this main routing demand model, we also developed a simplicity-driven model that is similar but much simpler in form. For $Eqv = 1$ the simplicity-driven model is

Simplicity-Driven Model:

$$W_{need}(Fs, Fc_{in}, Fc_{out}, L, Eqv = 1) = W_{abs\_min} + \frac{1}{\theta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)\left(\frac{W_{abs\_min}}{Fc_{out}}\right) + \frac{\lambda(L-1)}{4} \tag{4.8.0.12}$$

with

$$W_{abs\_min} = p\frac{\lambda\overline{R}}{2} \tag{4.8.0.13}$$

where $p = 1.4$, and $\theta = 50$.

When $Eqv = 0$, $\overline{R}$ in Equation 4.6.4.2 is replaced by $\overline{R}_{NE}$, defined by:

$$\overline{R}_{NE} = \sigma * \overline{R} \tag{4.8.0.14}$$

where $\sigma = 1.166$. And Fc_{in} in Equation 4.8.0.12 is replaced by $FcNE_{in}$, defined by:

$$FcNE_{in} = \frac{Fc_{in}}{I\mu} \tag{4.8.0.15}$$

where I is the number of input pins from which logical equivalence is removed, and $\mu = 0.06$. This simplicity-driven model is on average off by 13 tracks (which is 10%) in validation for $Eqv = 1$, and 26 tracks (which is 18%) in validation for $Eqv = 0$. It presents similar intuitions, although with less insights and is less accurate, compared to the main routing demand model.

In the next chapter applications of the routing demand models, in early-stage architecture development, will be presented. While model accuracies quoted in this chapter are measured on an individual circuit basis (for each circuit, λ, $\overline{R}$ and W_{need} are determined experimentally), in the application examples we measure the accuracy of the model for predicting routing demand when no circuits are given.

Chapter 5

Applications and Quality of Model

The FPGA interconnect models developed in the last chapter are intended for use when there are no circuits or tools that the architect can employ in an experimental architecture exploration process. In this chapter we describe how that might be done in three important application areas, relevant to architecture development. They are: 1. Comparing routing demand of different logic block architectures (Section 5.1), 2. Making routing architectural tradeoffs (Section 5.2), and 3. Estimation of channel width for commercial FPGAs in early generations (Section 5.3). For each application, we also give examples that assess the quality of the models during use. We will conclude with a discussion on the limitations of our models.

5.1 Comparing Routing Demand of Different Logic Block Architectures

An important step in architecting an FPGA is choosing a logic block. When choosing a logic block among many variants, it is important that the architect can evaluate and compare logic block choices for their routing demand, as routing dominates the area and therefore cost of the FPGA. However, at this stage of architecture development no complete routing architecture is chosen; a multitude of routing architecture choices exist. Also the architect may not have the circuits or tools for experimental evaluation at this stage. Even if there are circuits and tools, there are far too many combinations of logic block and routing architecture choices to

run experiments.

For this problem of evaluating and comparing logic block architectures for routing demand in early-stage development, our FPGA interconnect models are useful for a preliminary analysis. It can be used to provide quick feedback on each of the multitude of nascent architecture choices, without employing actual circuits or experiment tools.

To use our FPGA interconnect models to evaluate a logic block's routing demand W_{need}, the FPGA architect simply needs to provide parameter values to be employed in the models. Specifically, the architect needs to:

1. Estimate values for λ and $\overline{R}$ for each logic block under consideration
2. Provide values for the routing architecture parameters Fs, Fc_{in}, Fc_{out}, L and Eqv.

The routing architecture parameters are chosen by the FPGA architect, to represent possible routing architectures that may ultimately be employed.

The parameters λ and $\overline{R}$ are a characterization of the logic block (in terms of its post-placement routing demand in a fully flexible FPGA) circuit connectivity, and placement tool capability. During the model development in Chapter 4, they are measured using actual circuits and experiment tools, individually for each circuit. However, this cannot be done during the application of the model, as λ and $\overline{R}$ must be chosen to represent a set of circuits in a target market. Furthermore, there are no circuits or experiment tools in early-stage development, to measure λ and $\overline{R}$. We will discuss how the architect can select values for λ and $\overline{R}$, and give models for them, in turn in the following subsections.

5.1.1 Selecting a Value for λ

The architect must derive the value of λ from the logical nature of the block. The architect will know the number of input pins on the logic block (presumably having determined that as part of his/her initial thinking). The value of λ is a function of the number of inputs: it is the average number of *used* inputs per logic block expected for the most difficult circuit that the architect wants to succeed in routing. An upper bound for λ would be the number of input pins, but this appears to be unduly pessimistic. The best value to choose would likely be related

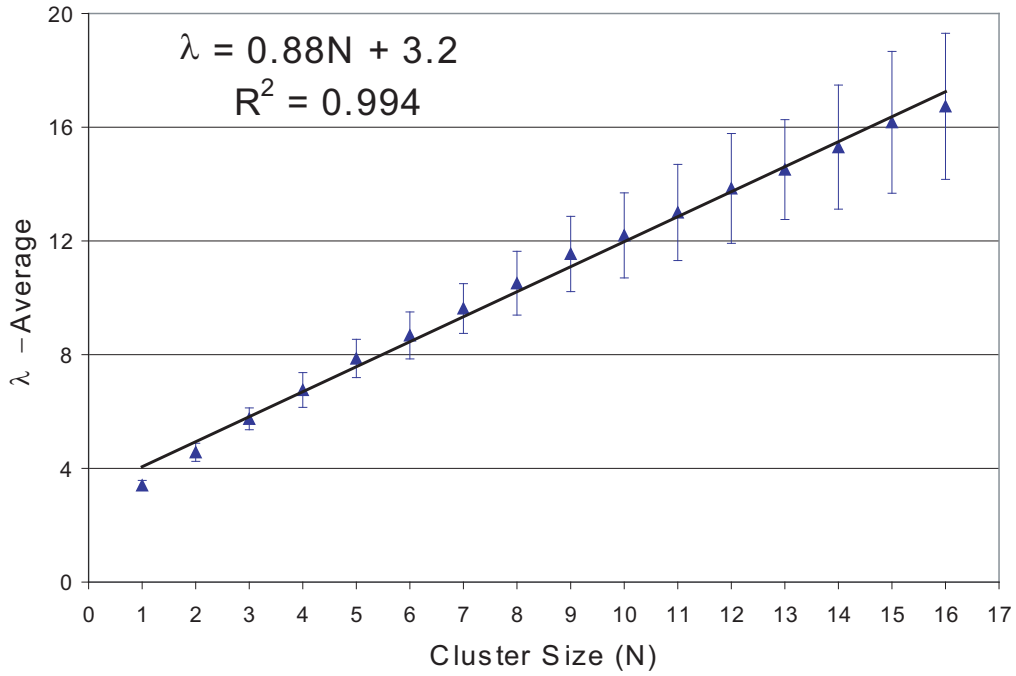


Figure 5.1: Average λ vs. cluster size (N)

to the logical nature of the block, and it is not possible to speak to the most general logic block case – for example, we believe that the most appropriate λ for a hard 20x20 multiplier as a function of its number of input pins would be rather different than the λ for a cluster of 12 5-input lookup tables, because the size of multiplier actually implemented can vary (thus number of used pins vary), while the cluster is likely as fully utilized as possible.

We *do* believe that the more common soft logic blocks would have similar behavior in the relationship between number of usable input pins and average number of used input pins. This is because they are being used to implement a common set of logic functions, regardless of what the native function of the logic block is. This assumption would not be true for highly inefficient logic blocks that don't use their available pins very often, compared to the more typical, fairly well-used pins.

To characterize this relationship, we measure λ by packing all benchmark circuits, using the synthesis flow described in Section 3.4, for different clustered logic block architectures. Figure 5.1 gives a plot of that measured average λ (y-axis), averaged over all benchmark circuits, for

clustered logic block architectures ranging from size 1 to 16 (x-axis). The y-error bars give one standard deviation of λ. From Figure 5.1, λ can be modeled as an increasing linear function of cluster size. Since the clusters are architected to use use $I = 2N + 2$ input pins (and recall from [9] this value is set to permit complete use of the internal logic of the cluster), the fitted equation in Figure 5.1 can be rewritten in terms of I by substituting $N = \frac{I-2}{2}$:

$$\lambda = 0.44I + 2.3 \tag{5.1.1.1}$$

An architect could use this model to choose λ as a function of I, the total number of inputs to a new logic block, particularly if the nature of the logic in the block is similar to LUT-based clusters. An example for which this would be true would be PLA-based logic blocks discussed in [17]. More roughly, Equation 5.1.1.1 suggests that on average λ is roughly half of the total number of input pins.

The model in Equation 5.1.1.1 gives the average λ. For the purpose of comparing routing demand of different logic block architectures, without any circuits, having average λ is useful (it can be used to determine average routing demand). In [42], it was suggested that 20% above the average routing demand is a useful metric for comparing architectures. Based on this, an architect can make routing demand comparisons of logic block architectures using the average λ.

Also discussed in [42], the determination of channel width in a commercial architecture is done by a full experimental flow on a large set of circuits, representative of the target market. Furthermore, more weight should be given to the circuits with the largest routing demand – difficult-to-route circuits. However, this does not imply setting the channel width to the absolute maximum routing demand of a circuit in the benchmark set, for fear of an outlier that would unduly and dramatically increase the channel width in the chip. Therefore, to use our routing model to estimate the channel width of a commercial architecture, the architect needs select a value of λ representative of difficult-to-route circuits in the set of circuits for the target market.

Since we do not have circuits in the target market, it is difficult to model this single value of λ. However, we can create a distribution model for λ based on our benchmark circuits. With

a distribution model of λ, we can model the the value of λ representative of difficult-to-route circuits – as the value that is larger than 98% of λ's in the distribution, because the higher λ the more difficult it is to route. This modeling has the disadvantage of not being truly reflective of the target market (which we don't have any information about), but it has the advantage that the quality of the model can be compared to a real commercial architecture, later in Section 5.3.1.

To model the distribution of λ, we use statistics gathered from our benchmark circuits. The standard deviation of measured λ, of our benchmark circuits, is linearly increasing w.r.t. cluster size, as shown in Figure 5.1. This growth in standard deviation can be explained as follows. When the cluster size is small, there is little opportunity for input sharing, thus all benchmark circuits have similar connectivity λ. When the cluster size is larger, more opportunities exist for input sharing: circuits begin to distinguish themselves on their differing ability to share inputs – resulting in different λ. Those circuits that share a lot of inputs have low λ (few of I available inputs is used), and high λ otherwise (most of I is used). Therefore standard deviation of λ increases as cluster size increases. Experimental data in Figure 5.1 shows a very linear increase, with correlation $R^2 = 0.982$.

Therefore by the same approach as the average, we model the standard deviation of λ as a linear function of I:

$$\lambda_{SD} = 8.05 \times 10^{-2} I - 0.25 \tag{5.1.1.2}$$

We further model λ of circuits in the target market as normally distributed. This is a reasonable assumption given the lack of information.

Given the distribution type, average, and standard deviation, we can find the value of λ of difficult-to-route circuits – that is more difficult to route than 98% of circuits's in the distribution. For a normal distribution 98% lies below approximately 2 standard deviations above the average [39]. Therefore a difficult-to-route circuit has:

$$\begin{aligned} \lambda_{diff} &= \text{average} + 2 \text{ standard deviations} \\ &= 0.44I + 2.3 + 2 * (8.05 \times 10^{-2} I - 0.25) \\ &= 0.60I + 1.8 \end{aligned} \tag{5.1.1.3}$$

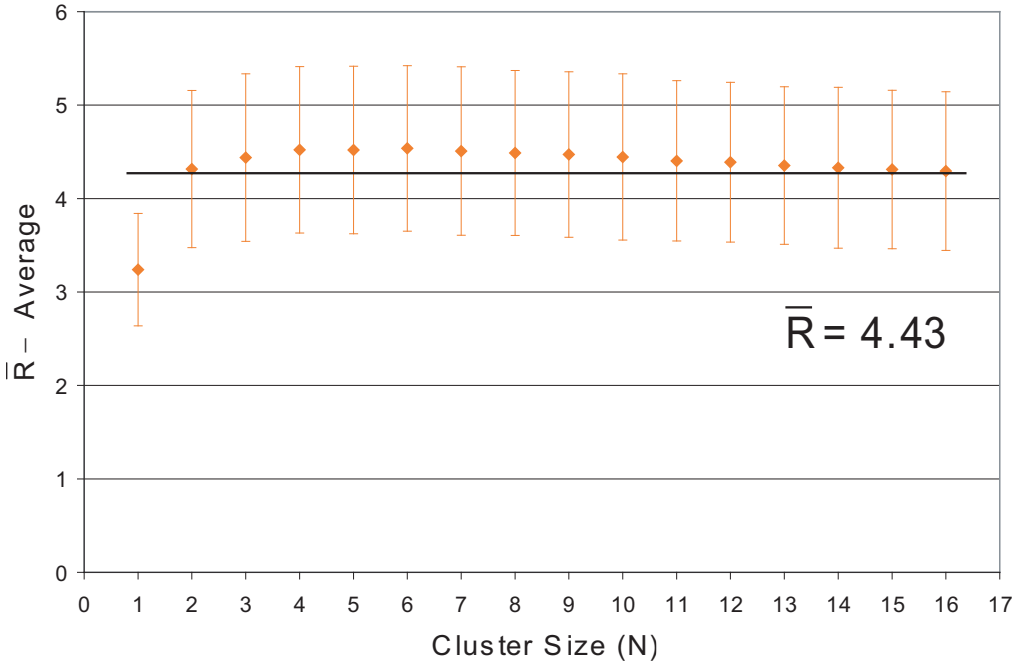


Figure 5.2: Average $\overline{R}$ vs. cluster size (N)

The model of λ of difficult circuits in Equation 5.1.1.3 will be used for estimating channel width of an commercial FPGA architecture in Section 5.3.1.

The second key parameter that the architect needs to determine, to evaluate logic block architectures is $\overline{R}$. We discuss how to select its value in the following subsection.

5.1.2 Selecting a Value for $\overline{R}$

In using our interconnect model to compare logic block routing demand the architect needs to determine $\overline{R}$, the average length of two-pin connections measured in number of logic blocks traversed.

For some insight into this metric, we measure $\overline{R}$ by packing and placing all benchmark circuits, using the synthesis flow described in Section 3.4, for different clustered logic block architectures. Figure 5.2 gives a plot of the measured average $\overline{R}$ (y-axis), averaged over all benchmark circuits, for clustered logic block architectures ranging from size 1 to 16 (x-axis). The y-error bars give standard deviation of individual circuit's $\overline{R}$ at each cluster size – we will

refer to this as the *across circuit standard deviation* ($\overline{R}_{ACSD}$).

Seen in Figure 5.2, measurements of this parameter show that $\overline{R}$ is remarkably consistent as a constant over different cluster sizes. The average of the benchmark circuits $\overline{R}$ is consistently near 4.43, its average across cluster sizes 1 to 16, with the exception of cluster size 1 where $\overline{R}$ is lower since there is no clustering. The standard deviation of average $\overline{R}$, across cluster sizes is 0.085, or 1.9% of the average 4.43; this shows a high degree of consistency. This is a somewhat surprising result: that over a wide range of granularity, the average length of connections remains relatively fixed when measured in terms of 2-dimensional array of the changing granularity. Others [40] have suggested that this is due to a some fractal nature of circuit netlists; this nature is inherent in the Rent's Rule model of circuit structure.

Since the average $\overline{R}$ of all benchmark circuits can be seen to be largely independent of the amount of logic in each logic block, an architect could estimate $\overline{R}$ as 4.43 for the average circuit, mapped to any clustered logic block. Of course, some other facet of the architecture may influence this choice – for example, this model of $\overline{R}$ as 4.43 doesn't model the effect of fixed I/O pad positions, and this is known to increase average wire length [7].

For estimating $\overline{R}$ of the difficult-to-route circuits we adopt the same strategy as used for λ. In Figure 5.2, the y-error bar shows the across circuit standard deviation of $\overline{R}$ ($\overline{R}_{ACSD}$). It shows the spread of the distribution of individual circuit's $\overline{R}$, at a cluster size. The figure shows that it is consistent at any cluster size, around its average value of 0.872, with its standard deviation (across cluster size) being 0.0218 or 2.5% of the average. Therefore we will model $\overline{R}_{ACSD}$ as constant 0.872. We further model the $\overline{R}$ distribution to be normal. Therefore, for any clustered logic block, the average circuit has $\overline{R} = 4.43$, and the difficult-to-route circuit (by our definition: more difficult to route than 98% of all circuits) has:

$$\begin{aligned} \overline{R}_{diff} &= \text{average} + 2 \text{ standard deviations} \\ &= 4.43 + 2 * (0.872) \\ &= 6.17 \end{aligned} \tag{5.1.2.1}$$

Again, the model of average $\overline{R}$ ($\overline{R} = 4.43$) will be used for comparing logic block architectures, and the model of $\overline{R}$ of difficult circuits in Equation 5.1.2.1 will be used for estimating

channel width of an FPGA commercially architected in a later section.

Having reviewed how to select input parameter values to our interconnect model, we will demonstrate examples of using the model to compare logic block architectures in early-stage development in the following subsection.

5.1.3 Comparing Clustered Logic Blocks

In this subsection, we will test the quality of our interconnect models in comparing logic block architectures' routing demand in early-stage architecture development. The method of selecting λ and $\overline{R}$ as the average (from the previous subsections) will be adopted: λ will be selected by $0.44I + 2.3$, and $\overline{R}$ will be the constant 4.43. This analysis does not use the models of λ and $\overline{R}$ for difficult-to-route circuit, but later Section 5.3.1 does.

In the first example, we will compare two cluster-based logic blocks that were not used in the training of the models: a cluster of 16 4-input LUTs and a cluster of 4 4-input LUTs. Their description are given in Table 5.1. To be clear, by using these methods to estimate λ and $\overline{R}$ we do not employ any circuits or tools, and hence this could be done at an "early stage" in architecture development. We realize that this tests the models on logic blocks quite similar in nature to the ones that the models were trained on. This has the disadvantage of not proving the models' utility for new and different blocks, but it has the advantage that the the quality of model results can be compared to experimentally measured data.

We will use the main routing demand model, repeated here in Equation 5.1.3.1, as well as the simplicity-driven model, repeated here in Equation 5.1.3.5, to predict routing demand. The results are given in Tables 5.2 and 5.3.

Logic Block Architecture	1	2
LUT Size (k)	4	4
Cluster Size (N)	4	16
Cluster Input Pins (I)	10	34

Table 5.1: Test logic block architectures for early-stage comparison

Main Routing Demand Model:

$$\begin{aligned} W_{need} = & W_{abs\_min} \\ & + \frac{1}{\beta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)^{\alpha_{in}}\left(\frac{W_{abs\_min}}{Fc_{out}}\right)^{\alpha_{out}} \\ & + \frac{\lambda(L-1)}{4}\left(1+\frac{1}{{Fc_{in}}^{\alpha_{in}}}\right) \end{aligned} \tag{5.1.3.1}$$

with

$$W_{abs\_min} = p\frac{\lambda\overline{R}}{2} \tag{5.1.3.2}$$

where, $p = 1.4$, $\beta = 3$, $\alpha_{in} = 0.5$, $\alpha_{out} = 0.25$. For $Eqv = 0$ replace $\overline{R}$ by $\overline{R}_{NE}$ and Fc_{in} by $FcNE_{in}$, defined by:

$$\overline{R}_{NE} = \sigma * \overline{R} \tag{5.1.3.3}$$

$$FcNE_{in} = \frac{Fc_{in}}{I\mu} \tag{5.1.3.4}$$

where $\sigma = 1.166$ and $\mu = 0.33$.

Simplicity-Driven Model:

$$\begin{aligned} W_{need}(Fs, Fc_{in}, Fc_{out}, L, Eqv = 1) = & W_{abs\_min} + \frac{1}{\theta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)\left(\frac{W_{abs\_min}}{Fc_{out}}\right) \\ & + \frac{\lambda(L-1)}{4} \end{aligned} \tag{5.1.3.5}$$

with

$$W_{abs\_min} = p\frac{\lambda\overline{R}}{2} \tag{5.1.3.6}$$

where $p = 1.4$, and $\theta = 50$. For $Eqv = 0$ replace $\overline{R}$ by $\overline{R}_{NE}$ and Fc_{in} by $FcNE_{in}$, defined by:

$$\overline{R}_{NE} = \sigma * \overline{R} \tag{5.1.3.7}$$

$$FcNE_{in} = \frac{Fc_{in}}{I\mu} \tag{5.1.3.8}$$

where $\sigma = 1.166$ and $\mu = 0.06$.

To employ each model, a set of routing flexibilities must be chosen. Tables 5.2 and 5.3 both give, in each horizontal row, a set of routing architecture parameters in the columns

Routing Architecture Parameters					Cluster Size N = 16			Cluster Size N = 4		
Fs	Fc_{in}	Fc_{out}	L	Eqv	Measured	Predicted	Error	Measured	Predicted	Error
3	12	4	4	1	90	94	5.1%	42	32	-24%
3	12	4	6	1	113	105	-7.0%	52	36	-30%
9	12	4	4	1	76	78	3.4%	34	29	-15%
9	12	4	6	1	94	89	-4.9%	41	33	-20%
3	20	4	4	1	85	88	3.9%	38	31	-20%
3	20	4	6	1	106	99	-6.8%	45	35	-24%
9	20	4	4	1	74	76	2.4%	32	28	-13%
9	20	4	6	1	93	86	-7.2%	40	32	-19%
9	12	8	4	0	119	118	-0.9%	43	35	-18%
9	20	4	4	0	116	112	-2.9%	42	34	-18%
					Average Absolute Error		**4.5%**	Average Absolute Error		**20%**

Table 5.2: Logic block routing demand comparison by the main routing demand model

1 to 5. Each then gives, for the "proposed" cluster-size 16 logic block the experimentally measured average routing demand W_{need} for that architecture (the average W required across all benchmark circuits), the routing demand predicted by the model (with λ and $\overline{R}$ selected as discussed above), and the percentage error defined by $\frac{predicted-measured}{measured} \times 100\%$. The bottom of the error column gives the average absolute error. Similarly the final three columns of the tables give the measured, predicted and error for a cluster of size 4.

For the main routing demand model (results in Table 5.2) the average absolute error in the cluster size 16 predictions, across the selected range of routing architectures is 4.5%, and 20% for the cluster size 4. The smaller cluster has a larger percentage error, one that is consistently low, we believe, in part because the model was trained on a larger cluster, and in part because the W's are much smaller. In general we believe the model is a success in its ability to make fairly good predictions across a wide range of routing architecture parameters. This shows that

Routing Architecture Parameters					Cluster Size N = 16			Cluster Size N = 4		
Fs	Fc_{in}	Fc_{out}	L	Eqv	Measured	Predicted	Error	Measured	Predicted	Error
3	12	4	4	1	90	88	-2.0%	42	27	-36%
3	12	4	6	1	113	97	-15%	52	30	-41%
9	12	4	4	1	76	74	-2.7%	34	26	-22%
9	12	4	6	1	94	82	-13%	41	30	-28%
3	20	4	4	1	85	79	-6.4%	38	27	-31%
3	20	4	6	1	106	88	-17%	45	30	-34%
9	20	4	4	1	74	71	-4.2%	32	26	-19%
9	20	4	6	1	93	79	-15%	40	29	-26%
9	12	8	4	0	119	87	-27%	43	30	-31%
9	20	4	4	0	116	89	-23%	42	30	-29%
					Average Absolute Error		**12%**	Average Absolute Error		**30%**

Table 5.3: Logic block routing demand comparison by the simplicity-driven model

our model enables the architect to confidently compare routing demands of logic blocks across a wide range of routing architectures, in early-stage architecture development, without circuits or experiment tools.

For the simplicity-driven model (results in Table 5.3) the same trends hold, but generally the accuracy is significantly worse than that of the main routing demand model. The simplicity came at a cost of intuition and accuracy. It is left to the architect's discretion on when to use the simplicity-driven model (probably "earliest"-stages) for its ease of use, and when to use the more insightful and accurate main routing demand.

Fs	Fc_{in}	Fc_{out}	L	Eqv
6	12	6	4	1

Table 5.4: A test routing architecture

N	Measured	Predicted	Error
4	35	29	-16%
5	40	33	-16%
6	44	38	-15%
7	48	42	-14%
8	53	46	-12%
9	56	50	-11%
10	60	55	-9.1%
11	63	59	-7.2%
12	67	63	-5.1%
13	68	68	-1.1%
14	72	72	0.3%
15	75	77	2.1%
16	78	81	3.7%
17	81	86	6.0%
18	83	90	9.0%
19	86	95	11%
20	88	100	13%
Average Absolute Error			**8.9%**

Table 5.5: Main routing demand model prediction for cluster sizes N for arch. in Table 5.4

To further validate the models, we compare routing demand predictions (using the same λ and $\overline{R}$ described above) for a single routing architecture, defined in Table 5.4, across a range of cluster sizes $N = 4$ to $N = 20$, in Tables 5.5 and 5.6, for the main routing demand model and simplicity-driven model respectively. The tables show: cluster size, measured, predicted and percentage error defined by $\frac{predicted-measured}{measured} \times 100\%$, in the four columns. The average error is quite small for this fairly flexible architecture, only 8.9% for the main routing demand model and 14% for the simplicity-driven model.

N	Measured	Predicted	Error
4	35	26	-25%
5	40	30	-25%
6	44	33	-24%
7	48	37	-23%
8	53	41	-23%
9	56	45	-20%
10	60	49	-19%
11	63	53	-17%
12	67	57	-16%
13	68	61	-11%
14	72	65	-9.8%
15	75	69	-7.7%
16	78	74	-5.6%
17	81	78	-3.5%
18	83	83	-0.2%
19	86	88	1.9%
20	88	93	5.1%
Average Absolute Error			**14%**

Table 5.6: Simplicity-driven model prediction for cluster sizes N for arch. in Table 5.4

The interconnect models developed in this work enable an FPGA architect to compare routing demand of logic blocks in an early-stage architect development. Quality predictions of routing demands of clustered logic blocks are possible by them, thus reducing the need for circuits and experiments. In addition to this application, our interconnect models also serve to provide architects intuition on routing architecture development – it captures interactions of routing parameters in simple analytical expressions. The following section explores how our models can provide quick feedback on routing architecture design and intuitions on routing

parameter tradeoff.

5.2 Routing Architectural Tradeoffs

In the development of the routing architecture, it is often necessary to explore many tradeoffs of routing parameters. Each tradeoff point may lead to different area, performance, and power metrics in the FPGA, so such exploration serves as an architecture-driven method to meet area, performance, and/or power goals. However, the effort required to empirically evaluate tradeoff points is very large, because the parameter interactions are not known, and they have to be determined by a multitude of experimental searches.

Instead of running thousands of hours of experiments for each tradeoff idea, the architect can use our interconnect model to provide: 1. quick feedback on which tradeoff points are worth further (more accurate) experimental evaluation, and 2. intuition on how the routing parameter interact and tradeoff for new architectural ideas.

To use our interconnect model for this purpose, during the routing architecture development stage, it can expected that the architect has chosen a logic block architecture. Thus, λ and $\overline{R}$ are known or can be well estimated (if no benchmark circuits to measure). The routing parameters are also known and a tradeoff is proposed – all input parameters to the model are known. The architect can then simply manipulate the analytic expression of our model to make the tradeoff. An example is given in the following subsection.

5.2.1 Trading Of Switches

During routing architecture development, suppose that an architect has chosen a routing architecture for a logic block. However, after more development in electrical design and analysis, the speed of the FPGA is is deemed too slow, and analysis indicates that it is because the multiplexer driving the single-driver wires are too big and slow. The architect proposes a tradeoff that reduces the input size of the wire-driving multiplexer at the cost of increasing the input size of input-pin-driving multiplexer.

This tradeoff has the potential to improve performance, since it speeds up the most fre-

quently used routing resource, at the expense of slowing a seldom used routing resource: a routed connection will likely go through many wire-driving multiplexers and only one input-pin-driving multiplexer.

This implies trading of switches from the switch block, reducing Fs, to the input connection block, increasing Fc_{in}. To make this tradeoff experimentally the architect must keep all other parameters fixed and run new experiments that varies only Fs and Fc_{in}. This could require modifications to experimentation tools and a day of experiments for each design iteration. As a first-order intuitive analysis to speed up the early design iterations or a guide for actual routing architecture exploration experiments the architect can use our interconnect model, as follows.

In the original routing architecture the switch block and input connection block parameters are known, i.e. $Fs = Fs_1$ and $Fc_{in} = Fc_{in1}$. The architect seeks to reduce Fs from Fs_1 to a second known value Fs_2, and the question is what is the new higher value of Fc_{in}, Fc_{in2}, that provides the same routing track demand, W_{need} (Here we assume that it is desired to keep the number of tracks the same; other kinds of calculations can be used if this is allowed to vary). This can be expressed as:

$$W_{need}(Fs_1, Fc_{in1}) = W_{need}(Fs_2, Fc_{in2}) \tag{5.2.1.1}$$

Using our main routing demand model, and keeping all other parameters constant, we arrive at the solution through a series of algebraic manipulations:

- If L = 1

$$Fc_{in2} = Fc_{in1} \cdot \left(\frac{Fs_1}{Fs_2}\right)^2 \tag{5.2.1.2}$$

- If L > 1

$$Fc_{in2} = Fc_{in1} \cdot \left\{ \frac{\frac{W_{abs\_min}{}^{1.75}}{3Fs_1 Fc_{cout}{}^{0.25}} + \frac{\lambda(L-1)}{4}}{\frac{W_{abs\_min}{}^{1.75}}{3Fs_2 Fc_{cout}{}^{0.25}} + \frac{\lambda(L-1)}{4}} \right\}^2 \tag{5.2.1.3}$$

The $L = 1$ case is the easiest to discuss – it shows that the new value of Fc_{in} increases as the square of the ratio of the reduction of Fs, which is a fairly expensive tradeoff. The $L > 1$ analytical expression is more complex but shows the tradeoff depends on other architecture parameter settings. The architect can use the analytical expressions to find how to trade Fs switches for Fc_{in} switches, as a quick first-order estimate and a guide for experiments.

Using the simplicity-driven model, the tradeoff for any L is:

$$Fc_{in2} = Fc_{in1} \cdot \left(\frac{Fs_1}{Fs_2} \right) \tag{5.2.1.4}$$

Although it is less accurate, the tradeoff derived from the simplicity-driven model reflects the insight that increase in Fc_{in} should be proportional to the loss in Fs as a ratio. The value of the simplicity-driven model is demonstrated here: the tradeoff expression in Equation 5.2.1.4 is simple for any L.

In general, the interconnect models developed in this work could be used in many different kinds of analysis of this sort. They provide FPGA architects intuitions on routing architecture tradeoff and guidance for which tradeoffs are worth further evaluation by compute-intensive and time-consuming experiments.

5.3 Channel Width Estimation of Commercial FPGAs in Early Generations

In recent years, a number of FPGA startups have failed due to lack of routing resources in their device. They architected a programmable routing fabric with a fixed channel width that is found to be insufficient for routing completion of circuits when used in the field. This problem occurs because at early stages of the startup, architects have little to no access to customer circuits to determine what is the required channel width to route their target market of circuits. Furthermore there is a lack of non-empirical method for predicting channel width requirements given different logic block architecture and routing architecture parameters. An architect for a startup has nothing to guide him/her to develop a first successful device which garners more customer circuits, reflective of the target market, for use in experiments more accurately determining the required channel width, which leads to more successful next-generation devices. This cyclic dependency means very few startups ever get off the ground.

The interconnect models developed in this work can guide an FPGA architect in designing a first generation device. They can be used to estimate the required channel width, as a complement to experiments on actual (although limited) circuits, for the goal of providing

sufficient width to route circuits in the target market not yet accessible to the startup. The "early" of this application title refers to early in the long-term time frame of generations of incremental architecture design, which is key to successful FPGA architecture development. Each new generation of Altera's and Xilinx's FPGAs are based on the knowledge gained from the preceding one.

To demonstrate this application and assess the quality of our interconnect models, we will determine how well our models could "predict" the channel width of an existing commercial FPGA. We extract from the existing commercial FPGA the following information:

1. Number of input pins emanating from the logic block

2. Values for the routing architecture parameters Fs, Fc_{in}, Fc_{out}, L and Eqv.

and treat them as the decisions made by the architect during development. They are inputs to the interconnect model.

For this analysis we will use our model of λ and $\overline{R}$ for difficult-to-route circuits in Equation 5.1.1.3 and Equation 5.1.2.1, to predict the channel width required, W_{need}. This is because modern FPGA vendors strive to make their FPGAs routable for almost all circuits in the target market, including the difficult-to-route ones, as discussed in Section 5.1.1. The predicted channel width, W_{need}, will be compared to the actual channel width of the device for assessing the quality of our models.

All of this will be done without any circuits and experiments. While we are not suggesting developing an FPGA routing architecture using these equations alone, they can be used to give a good sense of the scale of routing demand required.

The existing commercial FPGA we will use is Xilinx's Virtex 4 [65]. We describe its architecture and apply models to predict its channel width in the following subsection.

5.3.1 "Predicting" Channel Width of Virtex 4

While Xilinx's Virtex 4 FPGA certainly is not a first generation FPGA, we chose it to compare to our model because it is a recent commercially-successful FPGA. It was introduced in 2004

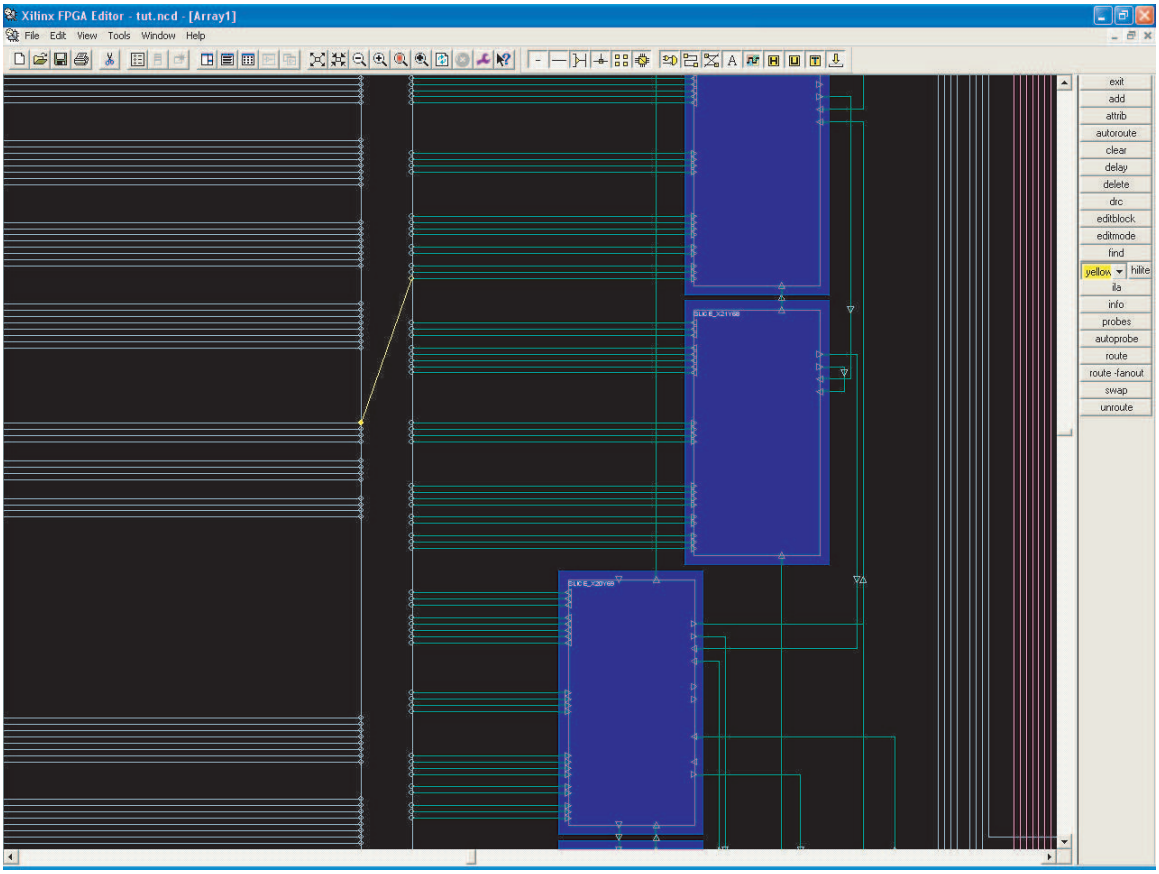


Figure 5.3: FPGA editor: Virtex 4

and manufactured on 90nm process technology. The specific device we chose in the Virtex 4 family is XC4VLX15, package SF363.

Its logic block architecture consists of 4 slices in a logic block, each slice containing two 4-input lookup tables (LUTs) and two registers [65]. To extract the architecture parameter values (input to our model) we used Xilinx's FPGA Editor 9.1i. A sample screen capture of which is shown in Figure 5.3.

The distribution of wire segments in each vertical and horizontal channel is:

1. 40 length two single-driver wires (called Double-length lines by Xilinx)

2. 120 length six single-driver wires (called HEX-length lines by Xilinx)

3. 24 chip length wires (called Long lines by Xilinx)

We will model all wire segments in this device as length six (i.e. 184 tracks of length six

wires), since our model can only model one type of wire segment length. In this case, since the majority of wire segments are single-driver length six (65%), this simplification will only cause a slight over-estimation (22% length two are modeled as length 6), which could be offset by an under-estimation of the chip length wires (13%). So $W_{FPGA} = 184$ and $L = 6$.

In the switch block, each incoming length six wire segment connects to: 2 horizontal length two wires, 2 vertical length two wires, 2 horizontal length six wires, and 2 vertical length six wires. So we will model as $Fs = 8$.

There are 32 input pins to the logic block, each connecting to 24 single-driver wire segments (half vertical, half horizontal) through a multiplexer called IMUX by Xilinx. And there are 8 output pins, each connecting to 24 single-driver wire segments through a multiplexer called OMUX by Xilinx.

Although the IMUX [65] provides depopulated switching between all input pins and adjacent wires, there is enough switching that we believe it gives logical equivalence between all input pins. Similar statement holds for OMUX.

This analysis translates to $I = 32$, $Fc_{in} = 24$, $Fc_{out} = 24$, and $Eqv = 1$.

Therefore, during development the architect of Virtex 4 made a number of architecture decisions (parameter values), which are summarized in Table 5.7. These parameter values are input into the interconnect model along with the λ and $\overline{R}$ model for difficult-to-route circuits.

Recall the model of λ and $\overline{R}$ for difficult-to-route circuits (higher than 98%) are:

$$\lambda_{diff} = 0.60I + 1.8 = 0.60(32) + 1.8 = 21 \tag{5.3.1.1}$$

$$\overline{R}_{diff} = 6.17 \tag{5.3.1.2}$$

The channel width predicted by our main routing model (in Equation 5.1.3.1) is $W_{need} = 133$. This under-predicts by 51 tracks (-28%). The breakdown of required channel width due

I	Fs	Fc_{in}	Fc_{out}	L	Eqv
32	8	24	24	6	1

Table 5.7: Virtex 4 architecture parameter values for our model

to various forms of inflexibility are summarized in Table 5.8.

	Number of Tracks	Percentage of Total
W_{abs_min}	91	68%
Switching matrix flexibility penalty	10	8.0%
Segment length penalty	32	24%
Total W_{need}	133	100%

Table 5.8: Breakdown of main routing demand model predicted track count for Virtex 4

The error in the prediction can be attributed to a number of reasons. One, due to layout constraints (such as tileability) channel width can only be set to quantized values, unlike the model which gives any real value W_{need}. This would cause the actual FPGA channel width higher than necessary, since architects prefer to err on the side of over-allocating channel width, than creating an unroutable FPGA. Two, the architectural characterization of logical equivalence $Eqv = 1$ causes under-prediction. Since there is depopulation in the IMUX and OMUX – which takes away flexibility in a logically equivalent architecture – the routing demand should be higher than W_{need} predicted for $Eqv = 1$. Finally, the biggest reason for the under-prediction is that Virtex 4 architecture was not architected to be just routable: for performance, the final channel width is typically set to around 20-30% of the minimum required channel width [1, 9], which is W_{need}. Considering this, the main routing model prediction is not as far off as the numbers indicate (-28%).

The channel width predicted by our simplicity-driven model (in Equation 5.1.3.5) is $W_{need} = 120$. This under-predicts by 64 tracks (-35%). The breakdown of required channel width due to various forms of inflexibility are summarized in Table 5.9.

The sources of error discussed for the main routing demand model applies to the simplicity-driven model. The additional error is due to the latter model not capturing some effects of inflexibility, for the sake of simplicity.

In summary, our interconnect models, using λ and $\overline{R}$ models for difficult-to-route circuits, yielded an under-prediction of the Virtex 4 FPGA channel width by 28-35%. Our models, however, could have guided the architect of Virtex 4 in finding what channel width ranges

	Number of Tracks	Percentage of Total
W_{abs_min}	91	76%
Switching matrix flexibility penalty	3	2.5%
Segment length penalty	26	21%
Total W_{need}	120	100%

Table 5.9: Breakdown of simplicity-driven model predicted track count for Virtex 4

more accurate experimental approaches should target. Our models can also provide a quick estimation of breakdown of channel width required due to various forms of inflexibility (in Table 5.8), which gives intuition to the architect on where routing flexibility (e.g. inflexibility of low percentage track count) can be traded off for other optimization goals such as performance and power.

5.4 Limitations of Model

There are a number of limitations to the models developed in this work.

The first is the modeling of λ and $\overline{R}$. These are empirical modeled using a set of benchmark circuits that may not reflect the market the architect wishes to target. As mentioned in the discussion of selecting λ in Section 5.1.1, the target market can have a dramatic effect on the modeling of λ. If the FPGA is architected for the digital signal processing application domain, where most designs all have high connectivity (i.e. higher λ), the average and deviation of λ of real circuits in the market are expected to be significantly different from our model. The same can be said for $\overline{R}$. In the future, it would greatly enhance the applicability of our interconnect model to develop a model for λ and $\overline{R}$ based purely on logic block architecture and other architecture features, and not on benchmarks.

As with all empirically developed models, this interconnect model is influenced by the benchmark circuits and experiment tools used to generate training data. However, FPGA architecture research must be specific of a set of experiment tools; it is the nature of FPGA architecture research: One cannot talk about FPGA architecture independently of the tools that

map circuits to it. On the other hand, the benchmark circuit bias is true and further validation using more recent (possibly industrial) benchmark circuits can help ease the concern.

Finally, despite use of many routing architecture parameters, the model is limited in architectural scope still. More complex/inflexible routing architectures include switch block internal population [9], mixtures of different types of wire segment lengths, and routing structures such as nearest neighbor connections [50].

5.5 Summary

In this chapter the interconnect model developed in this work was applied to three application areas in FPGA architecture development. First models for two input parameters, λ and $\overline{R}$, were discussed. They were input into the interconnect model for comparing clustered logic block architectures, in terms of routing demands. Results showed good accuracy of model for predicting large cluster architectures, and worse for small cluster architectures. Next, an example of trading of routing architecture parameters, trading of switches, is demonstrated using the interconnect model. The final application area is using the interconnect model and models for difficult circuit λ and $\overline{R}$, to predict channel width of a commercial FPGA. Very promising results were obtained for a Xilinx's Virtex 4 FPGA. Finally, we overviewed the limitations of our models.

Chapter 6

Conclusions

The objective of this research was to develop a simple and intuitive interconnect model for island-style FPGAs, for use in early-stage architecture development. The model is intended to guide an architect in evaluating new logic block architectures and routing architecture exploration, in the absence of empirical tools and circuits.

By a guided empirical modeling approach, in combination with derivations from intuitive observations, we have created a series of intermediate models predicting channel width required for successful routing, given some logic block architecture and circuit characterization and a set of well-known routing architecture parameters. We began with a simple and intuitive model for the required channel width (W_{abs_min}) of a routing architecture of extreme overpopulation of connectivity – the fully flexible FPGA. Then we incrementally reduced the flexibility of the modeled routing architecture, by reducing flexibility in routing architecture parameters. These parameters are: switch block flexibility Fs, connection block flexibilities Fc_{in} & Fc_{out}, wire segment length L, and logical equivalence of logic block pins Eqv. For each parameter, the interconnect model is generalized to capture its effect on the required channel width, creating a series of intermediate models where each is a superset of the previous. The final interconnect model is powerful enough to predict channel width requirement for the large space of architectures described by the five routing architecture parameters, given a logic block.

This gradual and incremental modeling approach gives insights into the fundamentals of FPGA routing and its interactions with logic block architecture, which is key for providing

guidance to FPGA architects in early-stage architecture development. We have developed the model for a balance of simplicity, giving intuition, and accuracy. Results showed that model is accurate on average within 15 tracks, or 14%, on our set of validation benchmark circuits.

Along the way, we have also created a simplicity-driven version of the routing demand model. The simplicity-driven model has value in its ease of use in expressing tradeoff equations. Its simplicity came at the cost of some insights and accuracy. It is accuracte on average within 26 tracks, or 18%, on our set of validation benchmark circuits.

This interconnect model is a contribution to the area of modeling of FPGA routing and its interaction with logic block architecture, that has been largely neglected in recent years. Furthermore it models modern FPGA single-driver routing architectures, which now dominate the industrial routing architectures. These have never been modeled before.

In addition to contributing to the modeling of FPGA routing, we have laid a first foundation on how to use the interconnect model to guide an FPGA architect in early-stage architecture development. In the process, models for characterizing circuits in clustered logic blocks were created; these are necessary for the application of the interconnect model. Three application examples were given. First, two clustered logic block were compared, for their average routing demand. Results showed the model predicted with an average error of 4.5% for cluster size 16 and 20% for cluster size 4, averaged across a number of different routing architectures. Also results showed the model predicted with an average error of 8.9% for a fixed routing architecture, across cluster sizes 4 to 20. Second, the interconnect model was used in exploring tradeoffs of routing architecture parameters. Specifically, the example of trading switch block for input connection block switches was given. The simplicity-driven version easily yielded very simple expressions of tradeoffs. Finally, the interconnect model was used in estimation of channel width for the commercial FPGA, Xilinx's Virtex 4. Results showed the model under-predicted by 51 tracks, which is -28% of the actual 184 tracks of the commercial FPGA.

6.1 Suggestions for Future Research

A number of directions exist for future research.

The generation of an FPGA area model to complement to the interconnect model of this work would greatly help an FPGA architect. Such an area model would take as input the same routing parameters and required channel width, as predicted by the interconnect model, to give analytic functions for FPGA area that give architects insights on how to optimize routing architecture for area and possibly can be optimized by derivation. The area model in combination with the interconnect model in this work can guide the FPGA architect in choosing logic blocks and optimizing routing architecture for area, in early-stage development.

Similarly, models for performance and power consumption in analytic expression, as a function of the same routing parameters, would be helpful in guiding the FPGA architect in early-stage development.

More work is needed in validating and improving the model for logic blocks that don't have the same properties of a typical lookup-table-based logic clusters. For logic blocks such as memory, and multiplier blocks etc. and mixtures of such logic blocks with lookup-table-based logic clusters, the interconnect model needed to be validated further. Certainly, for such logic blocks new models for λ and $\overline{R}$ must be created to reflect the logical and connective nature of the logic block. Once created, they should be tested for validation. After validated and improved, the interconnect model can guide architects in architecting next generation FPGAs, such as FPOAs [44].

Other aspects of this interconnect model should be further developed. Currently it can only model routing architectures of a single wire segment length. Mixtures of wire segment lengths should be modeled, as it is the choice of latest and future FPGA architectures, for it allows better utilization of routing wires (resulting in less segmentation waste) and improves performance.

Finally, partial logical equivalence is another aspect to add to this interconnect model. The current model assumes all input pins and output pins are either all logically equivalent or not. With the ability to model partial logical equivalence – where only some pins are logically equivalent – the interconnect model will have greater applicability, particularly to hard blocks such as memory block which has partial logical equivalence.

Appendix A

Additional Tables and Figures

	Measured							Predicted						
Fc_{in}/Fc_{out}	4	8	12	24	38	50	62	4	8	12	24	38	50	62
4	128	116	110	106	104	104	102	108	100	96	90	87	85	84
8	102	92	90	88	82	82	82	94	88	85	81	79	77	76
12	94	88	86	82	76	76	76	87	83	81	77	75	74	73
24	86	80	80	76	74	74	74	79	76	74	72	70	70	69
38	84	80	76	74	72	70	70	75	72	71	69	68	67	67
50	82	76	74	72	70	70	70	73	71	70	68	67	66	66
62	82	76	74	72	70	70	70	71	69	68	67	66	66	65
	Error							Error %						
Fc_{in}/Fc_{out}	4	8	12	24	38	50	62	4	8	12	24	38	50	62
4	-20	-16	-14	-16	-17	-19	-18	-16	-14	-12	-15	-16	-18	-18
8	-8	-4	-5	-7	-3	-5	-6	-8.1	-4.1	-5.1	-7.8	-4.0	-5.6	-6.7
12	-7	-5	-5	-5	-1	-2	-3	-7.1	-5.9	-6.4	-6.0	-1.2	-2.6	-3.6
24	-7	-4	-6	-4	-4	-4	-5	-8.1	-5.2	-7.3	-5.6	-4.9	-5.9	-6.7
38	-9	-8	-5	-5	-4	-3	-3	-11	-9.6	-6.5	-6.6	-5.5	-3.7	-4.3
50	-9	-5	-4	-4	-3	-4	-4	-11	-7.1	-6.1	-5.8	-4.5	-5.2	-5.8
62	-11	-7	-6	-5	-4	-4	-5	-13	-8.6	-7.5	-7.1	-5.7	-6.3	-6.8

Table A.1: Switching matrix model accuracy for training circuit *clma* $Fs = 3$

	Measured							Predicted						
Fc_{in}/Fc_{out}	4	8	12	24	38	50	62	4	8	12	24	38	50	62
2	96	80	80	78	76	76	76	94	88	85	81	79	77	76
4	88	78	76	76	76	72	72	84	80	78	75	73	72	71
8	80	72	72	70	70	68	68	76	74	72	70	69	68	68
12	76	70	70	68	68	68	68	73	71	70	68	67	66	66
24	74	68	68	68	68	68	68	69	67	67	65	65	64	64
38	74	68	66	66	66	66	64	67	66	65	64	63	63	63
50	74	66	66	66	66	64	64	66	65	64	63	63	63	62
62	72	66	66	64	64	64	64	65	64	64	63	62	62	62
	Error							Error %						
Fc_{in}/Fc_{out}	4	8	12	24	38	50	62	4	8	12	24	38	50	62
2	-2	8	5	3	3	1	0	-2.4	10	6.7	4.1	3.6	1.9	0.6
4	-4	2	2	-1	-3	0	-1	-5.1	2.1	2.1	-1.8	-4.0	0.0	-0.9
8	-4	2	0	0	-1	0	0	-4.6	2.2	0.2	0.1	-1.7	0.3	-0.4
12	-3	1	0	0	-1	-2	-2	-3.8	1.2	-0.4	0.0	-1.4	-2.2	-2.8
24	-5	-1	-1	-3	-3	-4	-4	-6.8	-0.9	-2.1	-3.9	-4.9	-5.5	-5.9
38	-7	-2	-1	-2	-3	-3	-1	-9.6	-3.5	-1.5	-3.0	-3.8	-4.3	-1.6
50	-8	-1	-2	-3	-3	-1	-2	-11	-1.9	-2.7	-4.0	-4.7	-2.1	-2.5
62	-7	-2	-2	-1	-2	-2	-2	-9.5	-2.8	-3.5	-1.7	-2.4	-2.7	-3.0

Table A.2: Switching matrix model accuracy for training circuit *clma* $Fs = 6$

	Measured							Predicted						
Fc_{in}/Fc_{out}	4	8	12	24	38	50	62	4	8	12	24	38	50	62
2	88	74	74	72	70	70	70	82	78	77	74	72	71	71
4	72	68	68	68	68	68	68	75	73	71	69	68	68	67
8	70	68	66	66	66	66	66	71	69	68	66	66	65	65
12	68	68	66	66	66	66	64	68	67	66	65	64	64	64
24	68	66	64	64	64	64	64	66	65	64	63	63	62	62
38	68	66	64	64	64	64	64	64	63	63	62	62	62	62
50	68	64	64	64	64	64	64	64	63	62	62	62	61	61
62	66	64	64	64	64	64	64	63	62	62	62	61	61	61
	Error							Error %						
Fc_{in}/Fc_{out}	4	8	12	24	38	50	62	4	8	12	24	38	50	62
2	-6	4	3	2	2	1	1	-6.7	6.0	3.5	2.4	3.1	1.8	0.9
4	3	5	3	1	0	0	-1	4.6	6.9	5.0	2.1	0.4	-0.5	-1.2
8	1	1	2	0	0	-1	-1	0.8	1.0	2.6	0.5	-0.7	-1.4	-1.9
12	0	-1	0	-1	-2	-2	0	0.6	-1.6	0.2	-1.6	-2.5	-3.1	-0.5
24	-2	-1	0	-1	-1	-2	-2	-3.5	-2.2	0.0	-1.2	-2.0	-2.4	-2.7
38	-4	-3	-1	-2	-2	-2	-2	-5.5	-3.9	-1.6	-2.6	-3.2	-3.5	-3.7
50	-4	-1	-2	-2	-2	-3	-3	-6.5	-1.8	-2.4	-3.3	-3.8	-4.1	-4.3
62	-3	-2	-2	-2	-3	-3	-3	-4.4	-2.5	-3.0	-3.8	-4.2	-4.5	-4.7

Table A.3: Switching matrix model accuracy for training circuit *clma* $Fs = 9$

	Measured							Predicted						
Fc_{in}/Fc_{out}	4	8	12	26	42	56	70	4	8	12	26	42	56	70
4	150	140	126	118	116	116	114	131	121	116	107	103	100	99
8	134	110	104	96	96	96	94	112	105	102	96	93	91	90
12	128	102	98	94	94	88	88	104	98	95	91	88	87	86
26	114	90	88	84	82	82	82	93	89	87	83	82	81	80
42	110	88	86	84	82	82	82	87	84	83	80	79	78	77
56	110	88	84	80	80	80	80	85	82	81	78	77	77	76
70	104	86	84	80	80	80	80	83	81	79	77	76	76	75
	Error							Error %						
Fc_{in}/Fc_{out}	4	8	12	26	42	56	70	4	8	12	26	42	56	70
4	-19	-19	-10	-11	-13	-16	-15	-13	-14	-8.2	-9.1	-11	-13	-13
8	-22	-5	-2	0	-3	-5	-4	-16	-4.3	-2.2	-0.3	-3.6	-5.3	-4.7
12	-24	-4	-3	-3	-6	-1	-2	-19	-3.5	-2.6	-3.6	-6.3	-1.5	-2.7
26	-21	-1	-1	-1	0	-1	-2	-19	-1.5	-1.6	-0.8	-0.5	-1.7	-2.5
42	-23	-4	-3	-4	-3	-4	-5	-21	-4.3	-3.9	-4.7	-4.1	-5.0	-5.7
56	-25	-6	-3	-2	-3	-3	-4	-23	-6.8	-4.0	-2.0	-3.5	-4.3	-4.9
70	-21	-5	-5	-3	-4	-4	-5	-20	-6.4	-5.6	-3.4	-4.7	-5.5	-6.0

Table A.4: Switching matrix model accuracy for validation circuit *pdc* $Fs = 3$

	Measured							Predicted						
Fc_{in}/Fc_{out}	4	8	12	26	42	56	70	4	8	12	26	42	56	70
2	120	94	94	90	88	86	86	112	105	102	96	93	91	90
4	94	86	86	86	82	82	82	99	94	92	88	85	84	83
8	90	82	82	80	80	80	80	90	87	85	82	80	79	79
12	86	82	80	80	78	78	78	86	83	82	79	78	77	77
26	82	78	76	76	76	76	76	80	78	77	76	75	74	74
42	80	76	76	74	74	74	74	78	76	75	74	73	73	73
56	80	76	76	74	74	74	74	76	75	74	73	73	72	72
70	80	76	76	74	74	74	74	75	74	74	73	72	72	72
	Error							Error %						
Fc_{in}/Fc_{out}	4	8	12	26	42	56	70	4	8	12	26	42	56	70
2	-8	11	8	6	5	5	4	-6.3	12	8.2	6.4	5.2	5.7	4.2
4	5	8	6	2	3	2	1	5.7	9.7	6.7	1.8	4.1	2.6	1.5
8	0	5	3	2	0	-1	-1	0.1	5.6	3.4	2.2	0.3	-0.8	-1.6
12	0	1	2	-1	0	-1	-1	0.0	1.4	2.1	-1.0	-0.1	-1.0	-1.6
26	-2	0	1	0	-1	-2	-2	-2.2	0.3	1.6	-0.6	-1.7	-2.3	-2.8
42	-2	0	-1	0	-1	-1	-1	-3.0	0.0	-1.0	-0.1	-1.0	-1.5	-1.9
56	-4	-1	-2	-1	-1	-2	-2	-4.7	-1.4	-2.3	-1.2	-2.0	-2.4	-2.8
70	-5	-2	-2	-1	-2	-2	-2	-5.8	-2.4	-3.2	-2.0	-2.7	-3.1	-3.3

Table A.5: Switching matrix model accuracy for validation circuit *pdc* $Fs = 6$

	Measured							Predicted						
Fc_{in}/Fc_{out}	4	8	12	26	42	56	70	4	8	12	26	42	56	70
2	90	84	84	80	80	80	80	98	93	90	86	84	83	82
4	84	82	80	78	76	76	76	89	85	84	81	80	79	78
8	80	78	76	76	76	76	74	83	80	79	77	76	76	75
12	78	76	76	76	74	74	74	80	78	77	75	75	74	74
26	78	74	74	74	74	72	72	76	75	74	73	72	72	72
42	76	74	74	72	72	72	72	74	73	73	72	71	71	71
56	76	74	74	72	72	72	72	73	73	72	71	71	71	71
70	76	74	74	72	72	72	72	73	72	72	71	71	70	70
	Error							Error %						
Fc_{in}/Fc_{out}	4	8	12	26	42	56	70	4	8	12	26	42	56	70
2	8	9	6	6	4	3	2	8.4	10	7.6	8.0	5.4	4.0	3.0
4	5	3	4	3	4	3	2	5.8	4.3	4.7	3.8	4.6	3.5	2.8
8	3	2	3	1	0	0	1	3.4	3.0	4.1	1.5	0.1	-0.6	1.5
12	2	2	1	-1	1	0	0	2.5	2.7	1.4	-0.7	0.8	0.1	-0.3
26	-2	1	0	-1	-2	0	0	-2.5	1.0	0.1	-1.4	-2.1	0.1	-0.2
42	-2	-1	-1	0	-1	-1	-1	-2.2	-1.0	-1.7	-0.1	-0.8	-1.1	-1.4
56	-3	-1	-2	-1	-1	-1	-1	-3.4	-1.9	-2.6	-0.9	-1.4	-1.7	-2.0
70	-3	-2	-2	-1	-1	-2	-2	-4.1	-2.6	-3.2	-1.4	-1.9	-2.2	-2.4

Table A.6: Switching matrix model accuracy for validation circuit *pdc* $Fs = 9$

Bibliography

[1] E. Ahmed and J. Rose, "The effect of LUT and cluster size on deep-submicron FPGA performance and density," *IEEE Transactions on VLSI*, vol. 12, no. 3, pp. 288–298, Mar. 2004.

[2] G. M. Allenby, "Cross-validation, the bayes theorem, and small-sample bias," *Journal of Business and Economic Statistics*, vol. 8, no. 2, pp. 171–178, 1990.

[3] A. Alsolaim, J. Becker, M. Glesner, and J. Starzyk, "Architecture and application of a dynamically reconfigurable hardware array for future mobile communication systems," in *IEEE Symposium on Field-Programmable Custom Computing Machines*, 2000, pp. 205–214.

[4] Altera Corporation, *Stratix II Device Handbook*, Oct. 2004.

[5] J. Anderson, S. Sheth, and K. Roy, "A Coarse-Grained FPGA Architecture for High-Performance FIR Filtering," in *ACM International Symposium on Field-Programmable Gate Arrays*, 1998, pp. 234–244.

[6] J. Anderson and F. Najm, "Power-aware technology mapping for lut-based fpgas," in *IEEE International Conference on Field Programmable Technology*, 2002.

[7] V. Betz and J. Rose, "Directional bias and non-uniformity in fpga global routing architectures," in *IEEE/ACM International Conference on Computer Aided Design*, 1996, pp. 652–659.

[8] ——, "Fpga routing architecture: Segmentation and buffering to optimize speed and density," in *ACM International Symposium on Field-Programmable Gate Arrays*, 1999.

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

[10] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic, *Field-Programmable Gate Arrays*. Kluwer Academic Publishers, 1992.

[11] P. K. Chan, M. D. F. Schlag, and J. Y. Zien, "On Routability Prediction for Field-Programmable Gate Arrays," in *IEEE/ACM Design Automation Conference*, 1993, pp. 326–330.

[12] T. Chan, J. Cong, and K. Sze, "Multilevel generalized force-directed method for circuit placement," in *International Symposium on Physical Design*, 2005, pp. 185–192.

[13] Y. Chang, S. Thakur, K. Zhu, and D. Wong, "A new global routing algorithm for fpgas," in *IEEE/ACM International Conference on Computer Aided Design*, 1994, pp. 356–361.

[14] C.-L. E. Cheng, "RISA: Accurate and efficient placement routability modeling," 1994, pp. 690–695.

[15] T. Coleman and Y. Li, "On the convergence of reflective newton methods for large-scale nonlinear minimization subject to bounds," *Mathematical Programming*, vol. 67, no. 2, pp. 189–224, 1994.

[16] ——, "An interior, trust region approach for nonlinear minimization subject to bounds," *SIAM Journal on Optimization*, vol. 6, pp. 418–445, 1996.

[17] J. Cong, H. Huang, and X. Yuan, "Technology Mapping and Architecture Evaluation for k/m-Macrocell-based FPGAs," *TODAES*, vol. 10, Jan.

[18] J. Cong and Y. Ding, "On area/depth trade-off in LUT-based FPGA technology mapping," in *Design Automation Conference*, 1993, pp. 213–218. [Online]. Available: citeseer.ist.psu.edu/cong94areadepth.html

[19] ——, "An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA designs," *IEEE Transactions on Computer-Aided Design*, vol. 13, no. 1, pp. 1–13, Jan. 1994.

[20] J. Cong and Y.-Y. Hwang, "Simultaneous depth and area minimization in LUT-based FGPA mapping," in *FPGA*, Feb. 1995, pp. 68–74.

[21] J. Davis, V. De, and J. Meindl, "A stochastic wire-length distribution for gigascale integration (GSI) – part i: Derivation and validation," *IEEE Transactions on Electron Devices*, vol. 45, no. 3, pp. 580–589, March 1998.

[22] M. Dehkordi and S. Brown, "The effect of cluster packing and node duplication control in delay driven clustering," in *IEEE International Conference on Field Programmable Technology*, 2002.

[23] W. Donath, "Placement and average interconnect lengths of computer logic," *IEEE Transactions on Circuits and Systems*, vol. CAS-26, pp. 272–277, 1979.

[24] ——, "Wire length distribution for placements of computer logic," *IBM Journal of Research and Development*, vol. 25, pp. 152–155, 1981.

[25] E.Bozorgzadeh and et al., "Routability-driven packing: Metrics and algorithms for cluster-based fpgas," *IEEE Journal of Circuits, Systems, and Computers*, vol. 13, no. 1, pp. 77–100, 2004.

[26] H. Eisenmann and F. Johannes, "Generic global placement and floorplanning," in *Proceedings of the 35th Annual Conference on Design Automation*, 1998, pp. 269–274.

[27] M. Feuer, "Connectivity of Random Logic," *IEEE Transaction on Computers*, vol. C-31, no. 1, pp. 29–33, Jan 1982.

[28] A. E. Gamal, "Two-Dimensional Stochastic Model for Interconnections in Master Slice Integrated Circuits," *IEEE Transactions on Circuits and Systems*, vol. CAS-28, no. 2, pp. 127–138, Feb 1981.

[29] L. Hagen, A. Kahng, F. Kurdahi, and C. Ramachandran, "On the intrinsic rent parameter and spectra-based partitioning methodologies," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 1, pp. 27–37, 1994.

[30] L. Hagen and A. Kahng, "Combining problem reduction and adaptive multi-start: A new technique for superior iterative partitioning," *IEEE Transactions on Computer Aided Design*, pp. 709–717, 1997.

[31] D. Huang and A.B.Kahng, "Partitioning-based standard-cell global placement with an exact objective," in *ACM Symposium on Physical Design*, 1997, pp. 18–25.

[32] D. Huang and A. Kahng, "When clusters meet paritions: New density-based methods for circuit decomposition," in *IEEE European Design and Test Conference*, 1995, pp. 60–64.

[33] P. Kannan and D. Bhatia, "Interconnect Estimation for FPGAs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 8, pp. 1523–1534, Aug. 2006.

[34] A. Kennings and I. Markov, "Analytical minimization of half-parameter wire length," in *IEEE Asia and South Pacific Design Automation Conference*, 2000, pp. 179–184.

[35] Y. Kwon and et al., "A 3-D FPGA wire resource prediction model validated using a 3-D placement and routing tool," in *International Workshop on System-Level Interconnect Prediction*, 2005, pp. 140–149.

[36] B. Landman and R. Russo, "On a pin versus block relationship for partitions of logic paths," *IEEE Transaction on Computers*, vol. C-20, pp. 1469–1479, 1971.

[37] G. Lemieux, S. Brown, and D. Vranesic, "On two-step routing for fpgas," in *International Symposium on Physical Design*, 1997, pp. 60–66.

[38] G. Lemieux, E. Lee, M. Tom, and A. Yu, "Directional and single-driver wires in fpga interconnect," in *IEEE International Conference on Field Programmable Technology*, 2004, pp. 41–48.

[39] A. Leon-Garcia, *Probability and Random Processes for Electrical Engineering.* Addison-Wesley Publishing Company, 1994.

[40] D. Lewis. Private Communication.

[41] D. Lewis and et al., "The Stratix Routing and Logic Architecture," in *ACM International Symposium on Field-Programmable Gate Arrays*, Feb. 2003, pp. 12–20.

[42] ——, "New FPGA Architectures: The Stratix II Logic and Routing Architecture," in *ACM International Symposium on Field-Programmable Gate Arrays*, Feb. 2005, pp. 12–20.

[43] A. Marquardt, V. Betz, and J. Rose, "Using cluster-based logic blocks and timing-driven packing to improve fpga speed and density," in *ACM International Symposium on Field-Programmable Gate Arrays*, 2000.

[44] Mathstar:, "reference: http://www.mathstar.com/overview.html."

[45] L. McMurchie and C. Ebeling, "Pathfinder: A negotiation-based performance-driven router for FPGAs," in *ACM International Symposium on Field-Programmable Gate Arrays*, 1995, pp. 111–117.

[46] R. Murgai, R. Brayton, and A. Sangiavanni-Vincentelli, "On clustering for minimum delay/area," in *IEEE/ACM International Conference on Computer Aided Design*, 1991.

[47] M. J. Myjak and J. G. Delgado-Frias, "A two-level reconfigurable architecture for digital signal processing," *Microelectronic Engineering*, vol. 84, no. 2, pp. 244–252, 2007.

[48] O. Organization, "reference: http://www.opencores.org. 2007."

[49] A. Rahman, S. Das, A. P. Chandrakasan, and R. Reif, "Wire Requirement and Three-Dimensional Integration Technology for Field Programmable Gate Arrays," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 1, pp. 44–54, Feb. 2003.

[50] A. Roopchansingh and J. Rose, "Nearest neighbour interconnect architecture in deep submicron FPGAs," in *IEEE Custom Integrated Circuits Conference*, 2002, pp. 59–62.

[51] J. Rose and S. Brown, "Flexibility of Interconnection Structures for Field Programmable Gate Arrays," *IEEE Journal of Solid-State Circuits*, vol. 26, no. 3, pp. 277–282, 1991.

[52] S. Sastry and A. Parker, "Stochastic models for wireability analysis of gate arrays," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 5, no. 1, pp. 52–65, Jan. 1986.

[53] SCU-RTL, "reference: http://www.engr.scu.edu/mourad/benchmark/rtl-bench.html. 1998."

[54] C. Sechen, "Chip-planning, placement, and global routing of macro/custom cell integrated circuits using simulated annealing," in *IEEE Conference on Design Automation*, 1988, pp. 73–80.

[55] A. Singh and M. Marek-Sadowska, "FPGA interconnect planning," in *International Workshop on System-Level Interconnect Prediction*, 2002, pp. 23–30.

[56] R. Snee, "Validation of regression models: Methods and examples," *Technometrics*, vol. 19, pp. 415–428, 1977.

[57] X. Song and et al., "Wire space estimation and routability analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 5, pp. 624–628, May 2000.

[58] M. Stone, "Cross-validating choice and assessment of statistical predictions (with discussion)," *Journal of the Royal Statistical Society*, vol. B36, pp. 111–147, 1974.

[59] J. S. Swartz, *M.A.Sc Dissertation: A High-Speed Timing-Aware Router*. University of Toronto, 1998.

[60] J. S. Swartz, V. Betz, and J. Rose, "A fast routability-driven router for FPGAs," in *ACM International Symposium on Field-Programmable Gate Arrays*, 1998, pp. 140–149.

[61] Texas-97, "reference: http://www-cad.eecs.berkeley.edu/respep/research/vis/texas-97/. 1997."

[62] B. Tseng, J. Rose, and S. D. Brown, "Improving FPGA routing architectures using architecture and CAD interactions," in *International Conference on Computer Design*, 1992, pp. 99–104.

[63] S. Wilton, C. Ho, P. Leong, W. Luk, and B. Quinton, "A synthesizable datapath-oriented embedded FPGA fabric," in *ACM International Symposium on Field-Programmable Gate Arrays*, 2007, pp. 33–41.

[64] Xilinx Corporation, "Virtex-ii platform FPGAs: Complete data sheet," June 2004.

[65] ——, "Virtex-4 Family Overview." 2005.

[66] S. Yang, "Logic Synthesis and Optimization Benchmarks, Version 3.0," in *Tech. Report, Microelectronics Centre of North Carolina*, 1991.