Modeling Routing Demand for Early-Stage FPGA Architecture Development

Wei Mark Fang and Jonathan Rose
The Edward S. Rogers Sr. Department of Electrical and Computer Engineering
University of Toronto, ON, Canada
{fang,jayar}@eecg.utoronto.ca

ABSTRACT

Architecture development for FPGAs has typically been a very empirical discipline, requiring the synthesis of benchmark circuits into candidate architectures. This is difficult to do in the early stages of architecture development, however, because there is no complete architecture to synthesize circuits into. The effort required to create prototype tools for nascent architectures is far too great for every new logic block or routing architecture idea, and so it would be extremely helpful to have a simple and intuitive FPGA interconnect model to guide the architect.

In this paper we present such an interconnect model for island-style FPGAs, whose single output is the estimated routing demand (often referred to as W, the number of routing tracks per channel) for an FPGA as a function of several logic block, circuit and routing architecture parameters. The goal of this model is to be as simple as possible, while still accurate enough to be useful, to provide understanding and intuition on FPGA routing. Our methodology is empirical – we propose model forms based on empirical observations, intuition and some derivation, and then fit models to experimentally generated data.

We show the development of the model in stages, beginning with a fully flexible FPGA, and gradually proceeding to one which includes the key parameters that control the flexibility of FPGA routing, and one key parameter describing the logic block and another relating to the typical circuit. We then show how to use these models in early-stage architecture development to provide feedback on several aspects of logic block architecture. We also show how the model can be used to explore the routing architecture space itself and to provide an overall intuition for architecture development.

Categories and Subject Descriptors

B.7.1 [**Integrated Circuits**]: Types and Design Styles—*Gate arrays*; I.6.5 [**Simulation and Modeling**]: Model Development—*Modeling methodologies*

FPGA'08, February 24-26, 2008, Monterey, California, USA.

General Terms

Design, Experimentation, Theory

Keywords

FPGA, Architecture, Routing, Model

1. INTRODUCTION

As Field-Programmable Gate Arrays (FPGAs) evolve, there is interest in developing new logic block architectures with more specific functionality including computational blocks, memories and even processors [17, 2, 6]. A key part in the evaluation of new architectures, as they are proposed, is to evaluate their impact on the demand for routing wires. This is important because routing demand is a key factor determining the area, performance and power consumption of the resulting FPGA.

Traditionally, routing demand is determined empirically by synthesizing benchmark designs to proposed architectures, requiring many hours of experiments. In the absence of a tool flow and sufficient benchmark circuit designs, during early-stage architecture development, an FPGA architect requires an interconnect prediction model to take the place of the empirical method. Such a model should be simple to use, capture important detailed routing parameters, and give intuition on the tradeoffs between these parameters.

In this paper, we present such an FPGA interconnect prediction model for island-style FPGAs. The inputs to the model are a few easy-to-measure parameters that can be derived from proposed logic blocks, basic knowledge of circuits, and detailed routing architecture parameters.

The output of the model is the routing demand/track count per channel required (often referred to as W in the literature) for successful detailed routing. Our goal is to develop this model balancing simplicity, accuracy and intuition/understanding.

This paper is organized as follows. Section 2 describes the basic terminology, defines the parameters used in the model, and reviews related previous work. Section 3 describes the modeling methodology and Section 4 develops the model incrementally, moving from a fully flexible routing architecture to a less flexible one. Section 5 shows how to use the model in various aspects of early-stage architecture development, and how some intuition can be derived from the model.

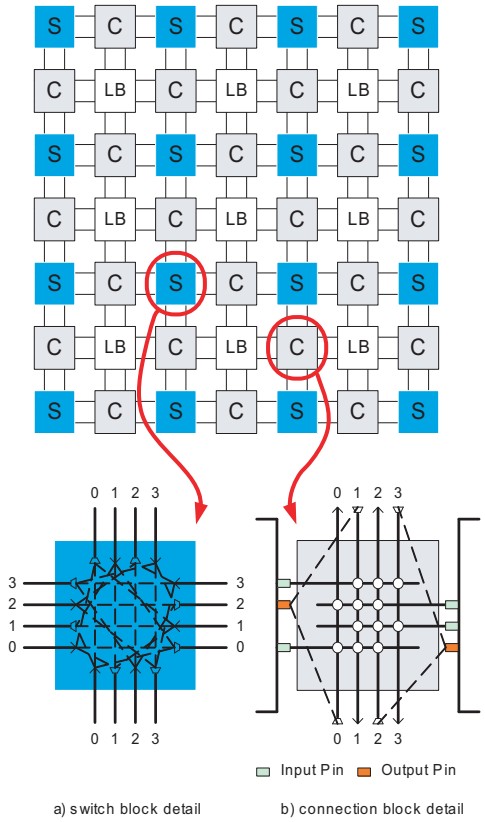

Figure 1: Island-Style FPGA Architecture [15]

2. BACKGROUND

2.1 Architecture Terminology and Parameters

The FPGA architecture modeled in this work is the island-style architecture illustrated in Figure 1. This style of FPGA consists of an array of logic blocks (LB) separated by horizontal and vertical routing channels which contain individual routing tracks. For the experimental part of this work, we will assume that logic blocks consist of clusters of N 4-input lookup table (LUT) based basic logic elements (BLE) that are fully connected as described in [3]. These clusters typically present I input pins and N output pins to the routing fabric (where I is set to be $2N + 2$ [3] to achieve good utilization of the cluster's logic). Although our experiments make use of this fairly standard soft logic cluster, this work seeks to create models that are applicable beyond this type of logic block.

The portion of a channel adjacent to a logic block is a channel segment. The channel width, W, is the number of wiring tracks in each channel segment. We assume that all channels have the same width as is commonly the case. We define the routing demand, W_{need}, of a circuit to be the minimum channel width an FPGA must have to successfully route that circuit.

This work will also assume that the detailed routing architecture employs the single-driver approach [15] (also known as *unidirectional* [1] and *direct drive* [16]) as this is now the common standard in industry. Here each routing wire can only be driven by a single source, chosen by a multiplexer at the beginning of the wire. A wire that spans multiple logic blocks is called a starting wire in the logical location containing its multiplexer.

A *switch block* (labelled S in Figure 1) exists at the intersection of the horizontal and vertical channels, and contains routing switches to connect wires for turning corners or extending wires farther along a channel. The flexibility of the switch block, **Fs** [20], is defined to be the total number of starting wires in this switch block each incoming wire can connect to. Routing switches are implemented as multiplexers that select the driver of a wire. The switch block at the bottom of Figure 1 uses a dashed line to indicate a routing switch between wires.

A *connection block* (C) conceptually contains the connections between the logic block and the routing segment. It is separated into the input connection block (which is a set of multiplexers selecting from the adjacent channel segments to connect to the logic block input pins) and the output connection block (which drive into the same multiplexers as the switch block). 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 on the logic block through routing switches. These routing switches are depicted in Figure 1b) as circles.

The flexibility of the output connection block, $\mathbf{Fc_{out}}$, 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 1b) uses a dashed line to indicate a routing switch between output pin and wire.

The length of a wire, **L**, is the number of logic blocks it spans without interruption by a programmable switch. We will model an FPGA in which all wires are of the same length, and leave the modeling of multiple lengths to future work.

The input and output pins on a logic block can be either logically equivalent (because each pin can be made to implement the same function, such as an AND gate) or not. Logical equivalence can also be architected into a logic block by constructing fully populated switching interconnect layer in front of the the input pins. We will refer to this architecture feature using the binary parameter **Eqv**. We will consider the input pins and output pins to either all be logically equivalent (Eqv=1) or not (Eqv=0).

These five parameters $\{\mathbf{Fs}, \mathbf{Fc_{in}}, \mathbf{Fc_{out}}, \mathbf{L}, \mathbf{Eqv}\}$ are the routing flexibility parameters that will be modeled below.

2.2 Related Work

Many interconnect models have been proposed in the past. The classical interconnect models [10, 12, 13] are analytical models built on Rent's Rule, which is an empirical observation that on average the size of a sub-circuit is proportional to the log of the number of its external connections. These models assume that routing wires have high flexibility in that they can turn and connect at any point, and hence do not take into account the inflexible nature of FPGA routing architectures. More modern extensions of the classical models [9, 22] predict wire-length and channel width distributions of gate arrays but still do not consider the impact of FPGA's inflexibility.

A number of FPGA-specific interconnect models have been proposed in literature. Chan et al. [5] applied classical models [12] and [13] to predict required channel width for a fixed FPGA architecture. Rahman et al. [19] used the modern

wire model [9] to formulate an interconnect model predicting 2D and 3D channel width requirement. However, Rahman only models wire length L and not the flexibility of the switching blocks.

The most related FPGA interconnect models to the work presented in this paper are from Kannan et al. [14] and Brown et al. [4]. In [14] Kannan models a large space of FPGA architectures using a routing resource graph generating tool and estimates the required channel width by computing wire demands from circuit placement information. This model is similar to ours in its ability to model a large space of FPGA architectures, but the main difference lays in that we require little to no circuit information and no tool flow. The FPGA interconnect model closest to the one in this paper is Brown's stochastic model in [4], which predicts the likelihood of successful routing given some of the FPGA flexibility parameters in Section 2.1, although only for $L = 1$ routing architectures. While similar in model inputs, the output of Brown's model is a probability value given channel width (W) as an input, but our model output is the required channel width for successful routing; this makes it difficult to compare models. Furthermore, our goal is somewhat different – to guide an architect in early-stage evaluation of logic blocks and early interconnect planning. For that reason, our model is built for simplicity with a willingness to sacrifice accuracy.

3. MODELING METHODOLOGY

Our goal is to determine a model of routing demand as a function of the logic block and routing architecture, in the form of simple and intuitive analytic equations. While the model is intended for use with little to no circuit information, we develop the model by analyzing experimental data on the relationship between routing demand and architecture. We generate this data by synthesizing a set of benchmark circuits into a number of different architectures. We propose candidate models based on trends observed from the data, intuition and some derivation, and then fit (train) the models to the experimental data. We use the most common method, the least squares fitting method, which minimizes the sum of the squares of the difference between model prediction and actual experimental data.

The benchmark circuits used are the largest 20 MCNC circuits [23] and in addition eight benchmark circuits obtained from open sources [18]. The latter includes various applications such as FIR filter and RS decoder design. The benchmark circuits are listed in Table 1.

Half of the benchmark circuits will be used to train the models, and half will be reserved to validate the quality of the trained model. These circuits are referred to as the "training" and "validation" sets respectively, and are listed in Table 1. The two sets are chosen by a random separation of the benchmark circuits with the constraint that the statistical properties of each set are similar to that of the combined set. This is done by measuring the average and variance of a number of circuit properties, including the BLE count and number of nets, of both training and validation sets and picking the random separation that yields the most similar average and variance between the two sets.

To measure the accuracy of the models presented, the prediction will be compared to measured results from the validation set of circuits synthesized into the modeled architecture. The accuracy metric we adopt is the Mean Absolute Percentage Error (MAPE) defined as:

Table 1: Benchmark Circuit List and Sizes in 4-Input BLEs

Training Set		Validation Set	
Circuit	# BLEs	Circuit	# 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

$$MAPE = \frac{1}{C} \sum_{\text{all C circuits}} \frac{|Predicted - Measured|}{Measured} \tag{3.0.1}$$

Although we use this as a more human-comprehensible accuracy metric, the fitting method employs the common mean-squared error metric as the optimization goal. The next section describes the details of the experimental methodology used to synthesizes benchmark circuits into different routing architectures, which are used in the model development process.

3.1 Experimental Methodology

The experimental flow used to generate the data relating routing demand to circuits and architecture is as follows: First, benchmark circuits are synthesized, using the FlowMap and FlowPack tools [7], into registers and lookup tables (LUT). They are then packed into logic clusters using a non-timing driven packing algorithm (VPACK) followed by wireability-driven placement and routing using a modified version of VPR [3]. (The single-driver routing architecture has been added into VPR by [24] and [11]. This work included extensive and careful creation of routing switch connectivity patterns.)

Each circuit is placed in the smallest square FPGA it can fit and routed by VPR (in routability-driven mode) into each architecture of interest. VPR determines the routing demand W_{need} required to successfully route each circuit by iteratively routing each circuit, reducing channel width of the routing architecture until it fails to route.

The logic block architecture used consists of a cluster of ten 4-input basic logic elements (BLE), each with a register and a 4-input LUT. Each logic block has $I = 22$ input pins, which are shared among all BLE inputs in the logic block. Each logic block has ten output pins, one for each BLE output.

Note that the entire flow used is non-timing driven. We believe that the routing demand in timing-driven mode is only marginally different than in non-timing driven mode because in both cases we force the router to primarily focus on routability since it is searching to find the minimum

possible track count, and so the parts of the cost function that derive from timing optimization will be relatively small. Test measurements bear this out: the routing demand ratio of timing-driven routing to routability-driven routing for a variety of different routing architectures ranges from 0.95 to 1.13 (with absolute track count ranging from 36 to 266), and has a mean of 1.02.

4. MODEL CONSTRUCTION

The goal of the model is to predict the number of routing tracks needed, W_{need}, for a given logic block and routing architecture. We will create the model in stages, dealing with three basic contributing factors: the fundamental logic block and circuit demand (which we call the absolute minimum track count), the increase in routing demand due to flexibility of the major switching stages in the routing fabric, and the effect of longer wires on routing demand. Although these three factors interact it is more intuitive to at first discuss them separately, as expressed in this high-level equation:

$$W_{need} = \begin{array}{c}\text{absolute}\\ \text{minimum}\end{array} + \begin{array}{c}\text{switching matrix}\\ \text{flexibility}\\ \text{penalty}\end{array} + \begin{array}{c}\text{segment}\\ \text{length}\\ \text{penalty}\end{array} \quad (4.0.1)$$

We will develop this model incrementally, starting with a fully flexible FPGA that provides the first term of this equation, and then incrementally generalizing it.

4.1 Model for the Fully Flexible FPGA

We begin with a fully flexible FPGA that has maximum connectivity between all routing resources – effectively a crossbar in the switch block, connection blocks, and the smallest length wires possible. Table 2 gives the routing parameter values of the fully flexible architecture. Routing in a fully flexible FPGA is very similar to ASIC routing, where routes can turn almost anywhere and have any length. For this reason we can build on El Gamal's master slice interconnect model [13] to form a model predicting W_{need}. A key result from El Gamal's model is that the average number of used wires per channel, W_{avg}, is given by:

$$W_{avg} = \frac{\lambda \cdot \overline{R}}{2} \quad (4.1.1)$$

where λ is the average number of used inputs on each logic block and $\overline{R}$ is the average point-to-point wire-length [13], measured in logic blocks traversed. These two parameters capture the routing demand of the logic block.

We will use these two parameters as inputs to our model, which characterize the routing demand of the logic block and circuits mapped and placed with that logic block. They can be measured on a circuit-by-circuit basis: λ measured for a soft-logic cluster FPGA using a packer and $\overline{R}$ measured by placement estimation. (In a later section, we will discuss how to estimate λ and $\overline{R}$ in the absence of synthesized and placed circuits)

Table 2: Fully Flexible FPGA Routing Architecture

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

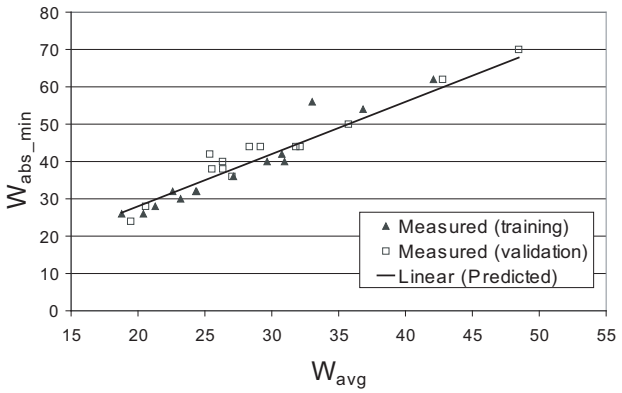

Figure 2: Prediction of W_{abs_min} for Fully Flexible FPGA

Equation 4.1.1 gives the average number of used wires per channel, but we are interested in estimating the *maximum* across all channels, which is clearly going to be larger than the average. We can derive an expression for the maximum used channel width, W_{need} by observing that FPGA routing channels have an average *utilization*, U – the fraction of wires that are actually used (similar to [21]) – given by:

$$U = \frac{W_{avg}}{W_{need}} \quad (4.1.2)$$

Re-arranging this equation to solve for W_{need} and substituting in Equation 4.1.1 for W_{avg} gives:

$$W_{need} = \frac{1}{U} \cdot \frac{\lambda \cdot \overline{R}}{2} \quad (4.1.3)$$

We will employ the inverse of the utilization term U, which we call the *peak factor*, p ($p = \frac{1}{U}$). Data from our experiments show that a peak factor $p = 1.4$ predicts W_{need} across a wide range of circuits in a fully flexible FPGA with a mean absolute percentage error (MAPE) of 6.2% for the training circuits and 5.8% for the validation circuits. (We used a measured value of λ and $\overline{R}$ from each circuit's synthesis and placement as inputs to the model) We will re-write Equation 4.1.3 and refer to the channel width requirement of a fully flexible FPGA by a special term, called W_{abs_min}, given as:

$$W_{abs\_min} = p\frac{\lambda \cdot \overline{R}}{2} \quad (4.1.4)$$

Figure 2 gives a plot of the model in Equation 4.1.4 and shows how close its prediction is to actual measured values of circuits in the training and validation sets.

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 tracks than W_{abs_min}. The following sections explore models for this reduction in flexibility.

4.2 Generalizing for Switch Block

The fully flexible FPGA model was developed assuming the *switch block flexibility* Fs is set at maximum value, $Fs = 3W$. To generalize the equation for Fs, we keep all other routing flexibility parameters at maximum and reduce Fs.

We have found that the range of interest for Fs is [3,21] – first because $Fs = 3$ is considered the minimum for it gives each wire only one chance to turn in each direction, and second because $Fs = 21$ is empirically equivalent to the fully flexible $Fs = 3W$.

Clearly as Fs decreases, W_{need} will increase, suggesting an inverse power function relationship. Furthermore, we postulate that the amount of increase will be in proportion to W_{abs_min}, since more wires lose flexibility for circuits with larger W_{abs_min}. This suggests the following model form:

$$W_{need} = W_{abs\_min} + \frac{1}{\beta}\frac{W_{abs\_min}}{Fs} \qquad (4.2.1)$$

where β is the parameter we will use to fit to the data.

We fit β to experimental data from synthesizing training circuits into routing architectures that are fully flexible with the exception of the switch block – here Fs is varied over the range [3,21]. A value of $\beta = 3$ achieves an MAPE of 6.6% for the training circuits and 5.9% for the validation circuits.

In this result and other accuracy results, we noticed that our validation MAPE was better than our training MAPE, which is surprising as one expects it to be the other way around. This was checked many times, and it is correct for the selected sets, which were chosen to be as similar as possible. It appears to be due to the fact that the validation circuits are just slightly more homogeneous.

We should note that we experimented with several other model forms, particularly with different exponents for Fs. Sensitivity analysis showed that an exponent of Fs in the range of 0.8 and 1.3 are all similarly good models, so we chose an exponent of 1 for simplicity.

4.3 Generalizing for Connection Block

Equation 4.2.1 captures the effects of reduced switch block flexibility when the *connection block flexibilities* are at maximum, i.e. $Fc_{in} = W$ and $Fc_{out} = W$. To generalize the model to include Fc_{in} and Fc_{out} we observe that the switch block, input connection block and output connection block form an inter-connected *switching matrix* programmably connecting all logic block pins and interconnect 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. For example, one could achieve the same routing demand, W_{need}, by lowering Fc_{in} and raising Fs an appropriate amount.

Furthermore, when Fs and Fc_{in} have high values the architecture is sufficiently flexible that W_{need} is close to the absolute minimum W_{abs_min}, so only small decreases in W_{need} can be had by increasing Fc_{out} (a "saturation" effect).

Finally, we have observed that the effective maximum of both Fc_{in} and Fc_{out} is W_{abs_min}, because making either Fc's larger than W_{abs_min} doesn't improve routability.

These arguments suggest a model of the form given in Equation 4.3.1:

$$W_{need} = W_{abs\_min} + \frac{1}{\beta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)^{\alpha_{in}}\left(\frac{W_{abs\_min}}{Fc_{out}}\right)^{\alpha_{out}} \qquad (4.3.1)$$

Here β remains set at 3, and α_{in} and α_{out} are exponents set by fitting to the training circuits' data, at 0.5 and 0.25 respectively. Their values are rounded for simplicity. The data were generated by synthesizing training circuits to the appropriate range of routing architectures – $Fc_{in}, Fc_{out} \in [2, W_{abs\_min}]$, and $Fs \in [3,21]$.

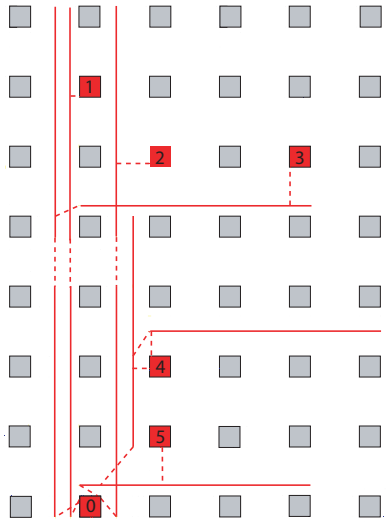

Figure 3: Routing Using Length 4 Wires

Equation 4.3.1 achieves an accuracy of 7.6% training MAPE and 6.5% validation MAPE.

The model exponents capture an important intuition about FPGA routing. The exponent $\alpha_{in} = 0.5$ is greater than $\alpha_{out} = 0.25$ reflecting the fact that each potential connection into a logic block is more important than each connection out – perhaps that the need to connect in upon "arrival" at a logic block is the last chance to complete a connection.

So far we have discussed the "absolute minimum" and "switching matrix flexibility penalty" terms described in the high-level equation, Equation 4.0.1. The remaining penalty in routing demand comes from the loss of flexibility in the interconnect wire length when using longer than unit length wires, and is discussed next.

4.4 Generalizing for Wire Length

The model up to this point has been developed for routing architectures containing interconnect wires that span only one logic block. Longer wires (where the parameter L is greater than 1) bring a different form of inflexibility, causing higher occupancy of the channels leading to higher W_{need}. Keep in mind that we assume all wires in an architecture will have the same length, L.

We have identified two major effects that increase routing demand: *segmentation waste*, and Fc_{in} *reduction* which we discuss in turn in the next two subsections.

4.4.1 Segmentation Waste

Segmentation waste arises when a wire segment is only used for a portion of its length, and so the remaining portion is wasted. This occurs either at horizontal-to-vertical (or vice-versa) turns (called turn waste), or wire-to-input pin connections (called pin waste), as illustrated in Figure 3, which shows connections from source block 0 to sink blocks 1 to 5 using $L = 4$ wires. For example, the waste associated

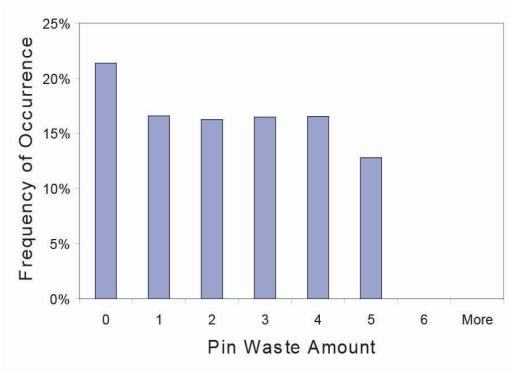

Figure 4: Pin Waste Distribution for $L = 6$ (circuit: *clma*)

with the connection from 0 to 3 is turn waste, and the waste associated with the connection from 0 to 1 is pin waste.

Waste doesn't occur at output pin connections because, in a single-driver routing architecture all output pins connect at the beginning of a wire by definition.

We found that pin waste is approximately uniformly distributed from 0 to $L-1$, by extracting the pin waste amount at each used input pin in the routed circuit. An example of this is the distribution of pin waste in the largest training circuit *clma* for $L = 6$, in Figure 4, where the x-axis indicates pin waste amount and y-axis indicates the frequency of occurrence. This distribution is typical of all circuits for any L. This is not unexpected since when a length L wire is used to connect to an input pin, the possible amounts of waste are $0, 1, 2, ..., L-1$, and since all wires in the routing fabric are of the same length each waste amount is equally likely to occur.

Our experimental data further suggest that the total segmentation waste is dominated by pin waste (with minor contribution from turn waste), so we will model total segmentation waste simply as uniformly distributed pin waste from 0 to $L-1$. This model enables the derivation of a simple analytic model of routing demand.

Using this model of segmentation waste, we can calculate the expected value of waste for each used input pin as:

$$\begin{aligned}\text{Segmentation Waste Per Pin} =& \frac{1}{L}(0) + \frac{1}{L}(1) + ... + \frac{1}{L}(L-1) \\ =& \frac{L-1}{2}\end{aligned} \tag{4.4.1}$$

The number of used input pins per channel segment is $\frac{\lambda}{2}$, since there are two channel segments per logic block (while there are four channel segments adjacent to a logic block, only one channel segment in each horizontal and vertical direction is owned by that logic block). The product of the segmentation waste per pin times the number of pins per channel segment gives the extra wire-length required in each channel segment, on average, which is the amount of routing demand increase needed:

$$\text{Segmentation Waste Per Channel Segment} = \frac{\lambda(L-1)}{4} \tag{4.4.2}$$

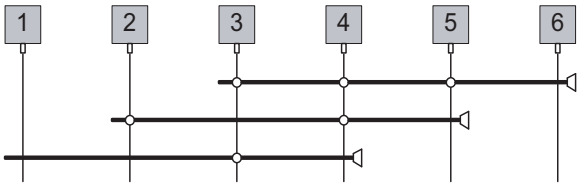

Figure 5: Fc_{in} Reduction Effect

4.4.2 Fc_{in} Reduction

The second cause of routing demand increase comes from an interaction between the Fc_{in} parameter and the segmentation waste. This effect is illustrated in Figure 5, which shows an architecture with three length four wires passing by six input pins. The wire-to-input-pin 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 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.

There is empirical evidence supporting this argument. Figure 6 is a plot of the average difference $W_{need}(L = 4) - W_{need}(L = 1)$ measured across all training benchmark circuits for a range of Fs and Fc_{in}. This figure shows that there are significantly more tracks required for longer-than-unit length architectures when Fc_{in} is small.

To account for the increased routing demand due to this effect, we observe that it occurs as a function of segmentation waste. Therefore we include a penalty term multiplier, that is a function of Fc_{in}, onto the segmentation waste term from Equation 4.4.2. Clearly this penalty term should act inversely to Fc_{in} – as Fc_{in} decreases, the penalty should become larger. We experimented with several forms of the

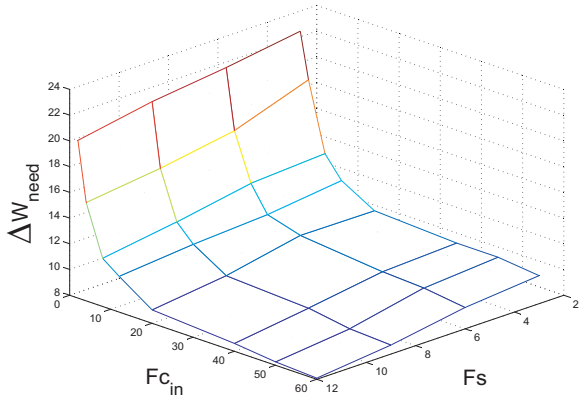

Figure 6: Average Increase in W_{need} Going From $L = 1$ to $L = 4$.

term to represent this effect and the one that provides the best balance between accuracy and simplicity is $(1+\frac{1}{Fc_{in}^{0.5}})$. The "1" in the term says that even when Fc_{in} is at maximum flexibility (where Fc_{in} reduction effect has little impact on routing demand) there is still routing demand increase purely as a result of segmentation waste.

This gives rise to an augmented version of Equation 4.3.1 in Equation 4.4.3, which contains the additional terms for segmentation waste and reduced Fc_{in}:

$$W_{need} = W_{abs\_min} + \frac{1}{3}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)^{0.5}\left(\frac{W_{abs\_min}}{Fc_{out}}\right)^{0.25} + \frac{\lambda(L-1)}{4}\left(1+\frac{1}{Fc_{in}^{0.5}}\right) \quad (4.4.3)$$

The accuracy for Equation 4.4.3 is given for values of $L \in \{1,2,4,6,8\}$ and Fs, Fc_{in} and Fc_{out} varied over their range as before (in Section 4.3). The training MAPE is 7.9% and validation MAPE is 6.5%. Note that this model does not introduce any new fitting parameter. The breakdown of the validation at longer wire lengths is in Table 3.

4.5 Impact of Logical Equivalence

The final routing flexibility parameter we consider is logical equivalence Eqv. When $Eqv = 1$ all input pins are logically equivalent and all output pins are logically equivalent. When $Eqv = 0$ all pins are distinct. We assume that the pins are evenly spread on all four sides of the logic block. The effect of removing logical equivalence is primarily to increase the distance each connection must travel (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.

To account for the first effect, we model that the average wire-length, $\overline{R}$, increases. We empirically measured that the total wire-length increases by a factor of 1.166, on average across all training circuits (consistently), when logical equivalence is removed. We model this by replacing $\overline{R}$ by $\overline{R}_{NE}$ in the model when $Eqv = 0$, where $\overline{R}_{NE}$ is defined by:

$$\overline{R}_{NE} = \sigma \cdot \overline{R} \quad (4.5.1)$$

and $\sigma = 1.166$. This parameter is not rounded up because it is sensitive. The knowledge embedded in this model is that, when logical equivalence is removed the average distance each point-to-point connection must travel increases by about 17%.

To account for the second effect (an effective reduction in Fc_{in}), 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}. The loss of those I pins should reduce the effective Fc_{in} by perhaps a constant amount each, and so we propose that Fc_{in} be replaced by:

Table 3: Breakdown of Accuracy of Equation 4.4.3

L	2	4	6	8
MAPE	6.2%	4.3%	4.1%	12%

$$FcNE_{in} = \frac{Fc_{in}}{I\mu} \quad (4.5.2)$$

where μ is a parameter fit with experimental data. The best fit value of μ is 0.33 (after rounding), for a training MAPE of 15% and a validation MAPE of 14%.

Fc_{out} could also be reduced but its effect is so small that it's ignored for model simplicity. Thus the impact of logical equivalence can be modeled using Equation 4.4.3 by increasing $\overline{R}$ by a factor $\sigma = 1.166$ and replacing Fc_{in} with $FcNE_{in}$ from Equation 4.5.2.

4.6 Model Summary

The FPGA routing demand model can be summarized as follows:

$$W_{need} = W_{abs\_min} + \frac{1}{\beta}\left(\frac{W_{abs\_min}}{Fs}\right)\left(\frac{W_{abs\_min}}{Fc_{in}}\right)^{\alpha_{in}}\left(\frac{W_{abs\_min}}{Fc_{out}}\right)^{\alpha_{out}} + \frac{\lambda(L-1)}{4}\left(1+\frac{1}{Fc_{in}^{\alpha_{in}}}\right) \quad (4.6.1)$$

with

$$W_{abs\_min} = p\frac{\lambda\overline{R}}{2} \quad (4.6.2)$$

Here, $p = 1.4$, $\beta = 3$, $\alpha_{in} = 0.5$, and $\alpha_{out} = 0.25$. Finally, for the case of no logical equivalence, $\overline{R}$ should be replaced with $\overline{R}_{NE}$ as given in Equation 4.5.1 and Fc_{in} should be replaced with $FcNE_{in}$ as given in Equation 4.5.2, with $\sigma = 1.166$ and $\mu = 0.33$.

5. MODEL QUALITY AND APPLICATION

The goal of this work is to create a model that can be used in early-stage architecture development, when there are no circuits or tools that the architect can employ. For that to be possible, some of the parameters required to use the model in Equation 4.6.1 must be estimated. We discuss how that might be done in the first section below, and provide some example results. In the subsequent section we give an overview of the kind of accuracy the model provides for the soft-logic cluster blocks for which we can generate measured data. Finally, we show another way in which the model can be used: to gain some intuition and understanding on routing architecture tradeoffs.

5.1 Comparing Logic Block Architectures

An important step in architecting an FPGA is choosing a logic block, as it can have a dramatic effect on the need for routing in the FPGA, which tends to dominate the area and therefore cost of the FPGA.

To determine the logic block's routing demand W_{need} using Equation 4.6.1 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 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

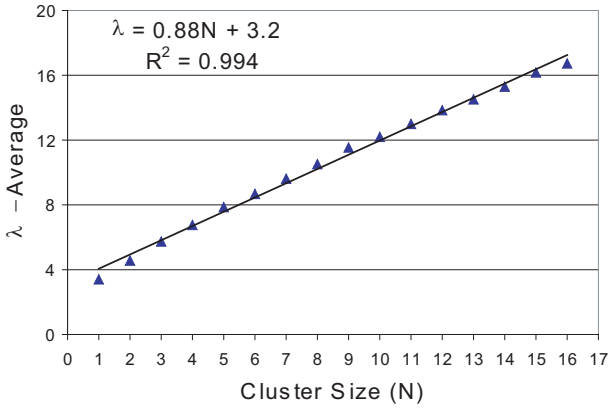

Figure 7: Average λ vs. Cluster Size (N)

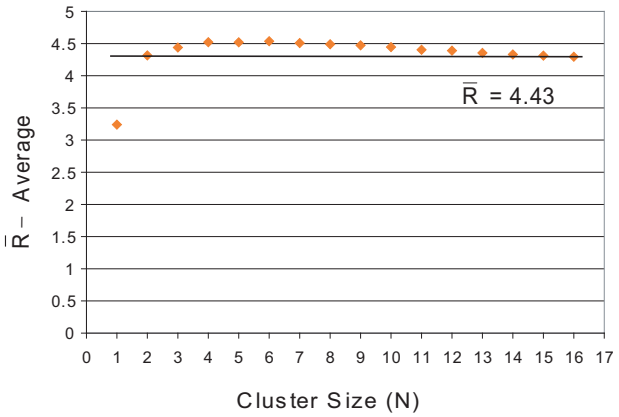

Figure 8: Average $\overline{R}$ vs. Cluster Size (N)

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 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.

We *do* believe that the more common soft logic blocks would have similar behavior in the relationship between the number of usable input pins and average number of used input pins. Figure 7 gives a plot of that measured average λ, averaged over all our benchmark circuits, for clustered logic block architectures ranging from size 1 to 16 consisting of 4-input LUT-based BLEs. It can be modeled as an increasing linear function of cluster size as given in the figure. Since the clusters are architected to use $I = 2N + 2$ input pins (and recall from [3] this value is set to permit complete use of the internal logic of the cluster), we could re-write the fitted equation in Figure 7 in terms of I by subsituting $N = \frac{I-2}{2}$:

$$\lambda = 0.44I + 2.3 \qquad (5.1.1)$$

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 [8].) More roughly, equation 5.1.1 suggests that λ is roughly half of the total number of input pins.

A second key parameter that the architect needs to determine is $\overline{R}$, the average length of two-pin connections measured in number of logic blocks traversed. Our experience with measurements of this parameter (and a number of other measurements in other contexts) is that $\overline{R}$ is remarkably consistent as a constant over different cluster sizes, when averaged over a large number of circuits. 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. It can be seen empirically in Figure 8, where the measured $\overline{R}$, averaged over all the benchmark circuits for clustered logic block architectures from size 1 to 16 consisting of 4-input LUT-based BLEs, are shown. The average of all circuits $\overline{R}$ can be seen to be largely independent of the amount of logic in each logic block: $\overline{R}$ is consistently around 4.43 for different cluster sizes N (with the exception of $N = 1$, where $\overline{R}$ is lower since there is no clustering). So, the architect could reasonably choose to estimate $\overline{R}$ as 4.43 for any logic block, and not be far wrong. Of course, some other facet of the architecture may influence this choice – for example, if this was a heterogenous logic block spread out among other blocks, the distance between the heterogenous blocks would likely increase $\overline{R}$.

5.1.1 *Quality of Model in Early-Stage Architecture Development*

To test the quality of the model given in Equation 4.6.1 in early-stage architecture development, we will use the above methods of selecting λ (with Equation 5.1.1) and $\overline{R}$ (as a constant 4.43) to compare two cluster-based logic blocks that were not used in the training of the model: a cluster of 16 4-input LUT-based BLEs and a cluster of 4 4-input LUT-based BLEs. 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 model on logic blocks quite similar in nature to the ones that the model was trained on. This has the disadvantage of not proving the model's utility for new and different logic blocks, but it has the advantage that the quality of the results can be compared to experimentally measured data.

To employ the model embodied in Equation 4.6.1, a set of routing flexibilities must be chosen. Table 4 gives, in each horizontal row, a set of routing architecture parameters in the columns 1 to 5. It 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 by benchmark circuits in Table 1), the routing demand predicted by the model, and the percentage error. The bottom of the error column gives the average absolute error. Similarly the final three columns of the table give the measured, predicted and error for a cluster of size 4. 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 model consistently under-predicts

Table 4: Routing Demand Prediction vs. Measured Across Routing Architectures

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: Routing Demand Prediction vs. Measured Across Cluster Sizes (under routing architecture $Fs = 6$, $Fc_{in} = 12$, $Fc_{out} = 6$, $L = 4$, and $Eqv = 1$)

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%**

the smaller cluster leading to a larger percentage error, 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.

To further validate the model, we compared routing demand predictions (using the λ and $\overline{R}$ method described above) for a single routing architecture ($Fs = 6$, $Fc_{in} = 12$, $Fc_{out} = 6$, $L = 4$, and $Eqv = 1$) across a range of cluster sizes $N = 4$ to $N = 20$, in Table 5. Looking at Table 5, there is a possible systematic error: the prediction error is not random, and it is increasing with cluster size. Further validation of the model on other logic blocks in future works should uncover this error and further improve the accuracy of the model. Despite this, the average error is quite small, only about 9%, which shows the potential of the model.

5.2 Architectural Intuition and Tradeoffs

An exciting advantage of the model developed in this work is that it can be analytically manipulated to provide insights into routing architecture tradeoffs. For example, the model provides a way to maintain a fixed amount of routability while exploring different routing parameters that might lead to different FPGA performance, power or other metrics.

Suppose that an architect has chosen a routing architecture for a logic block. However, after more development in electrical design, the speed of the FPGA is deemed too slow, and analysis indicates that it is because the muxes driving the single-driver wires are too big and slow. The architect proposes a tradeoff that reduces the input size of the wire-driving muxes at the cost of increasing the input size of input-pin-driving muxes. 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 to speed up actual architecture exploration experiments, the architect can use the model developed here as follows.

Suppose that in the original routing architecture the switch 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} that provides the same routing demand. This can be expressed as:

$$W_{need}(Fs_1, Fc_{in1}) = W_{need}(Fs_2, Fc_{in2}) \qquad (5.2.1)$$

Using Equation 4.6.1, and keeping all other parameters constant, we arrive at the solution through a series of algebraic manipulations:

- If L = 1

$$Fc_{in2} = Fc_{in1} \cdot \left(\frac{Fs_1}{Fs_2}\right)^2 \qquad (5.2.2)$$

- If L > 1

$$Fc_{in2} = Fc_{in1} \cdot \left(\frac{\frac{W_{abs\_min}{}^{1.75}}{3Fs_1 Fc_{cout}{}^{0.25}} + \frac{\lambda(L-1)}{4}}{\frac{W_{abs\_min}{}^{1.75}}{3Fs_2 Fc_{cout}{}^{0.25}} + \frac{\lambda(L-1)}{4}}\right)^2 \qquad (5.2.3)$$

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 in Fs, which is a fairly expensive tradeoff! In general, our analytic model could be used in many different kinds of analysis of this sort.

6. CONCLUSION

We have proposed an island-style FPGA routing demand model in the form of simple and intuitive equations, to guide an FPGA architect in early-stage architecture development. We have shown how this model can be used for preliminary feedback on logic block architecture evaluations and understanding the nature of tradeoffs between routing flexibility parameters, all in the absence of circuits and prototype tools. We have also shown the model to produce fairly good accuracy, at least in modeling the more well-known types of soft logic blocks.

We see a number of new lines of research that could be pursued: the generation of an area model that predicts area as a function of the routing parameters described above could lead to further analytic insights on how to optimize routing architecture for area. Also, an attempt to model performance (and perhaps power consumption) in addition to these routing demand equations would be helpful, to the early-stage architect. We also need to find ways to validate and improve the model for logic blocks that don't have the same properties as typical soft-logic clusters. Also, while we have provided physical interpretations for some parameters in the model (such as p and σ), more work is needed in providing physical explanations of the remaining parameters. Finally, we need to develop certain aspects of this model further – to support mixtures of wire lengths, and to understand the effect of partial logical equivalence – where only some pins are logically equivalent.

7. ACKNOWLEDGEMENTS

Wei Mark Fang received financial support from NSERC and this research project was also supported by a NSERC Discovery Grant.

8. REFERENCES

[1] Xilinx Inc. The Programmable Logic Data Book. 1999.

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

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

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

[5] P. K. Chan, M. D. F. Schlag, and J. Y. Zien. On Routability Prediction for Field-Programmable Gate Arrays. In *Proceedings of the 30th International Conference on Design Automation*, pages 326–330, 1993.

[6] D. Chen and J. Rabaey. A Reconfigurable Multiprocessor IC for Rapid Prototyping of Algorithmic Specific High Speed DSP Data Paths. *IEEE Journal of Solid State Circuits*, 27(12):1895–1904, 1992.

[7] J. Cong and Y. Ding. FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table Based FPGA Designs. *IEEE Trans. on CAD*, pages 1–12, Jan. 1994.

[8] J. Cong, H. Huang, and X. Yuan. Technology Mapping and Architecture Evaluation for k/m-Macrocell-based FPGAs. *Transactions on Design Automation of Electronic Systems*, 10:3–23, Jan. 2005.

[9] J. A. Davis, V. K. De, and J. D. Meindl. A Stochastic Wire-Length Distribution for Gigascale Integration: Part I: Derivation and Validation. *IEEE Trans. on Electron Devices*, 45(3):580–589, March 1998.

[10] W. E. Donath. Placement and Average Interconnection Lengths of Computer Logic. *IEEE Trans. on Circuits and Systems*, CAS-26(4):272–277, 1979.

[11] W. M. Fang. *M.A.Sc Thesis: Modeling Routing Demand for Early-Stage FPGA Architecture Development*. University of Toronto, Dec. 2007.

[12] M. Feuer. Connectivity of Random Logic. *IEEE Trans. on Computers*, C-31(1):29–33, Jan 1982.

[13] A. E. Gamal. Two-Dimensional Stochastic Model for Interconnections in Master Slice Integrated Circuits. *IEEE Trans. on Circuits and Systems*, CAS-28(2):127–138, Feb 1981.

[14] P. Kannan and D. Bhatia. Interconnect Estimation for FPGAs. *IEEE Trans. on CAD*, 25(8):1523–1534, Aug 2006.

[15] 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*, pages 41–48, Dec. 2004.

[16] D. Lewis and et al. The Stratix Routing and Logic Architecture. In *International Symposium on Field Programmable Gate Arrays*, pages 12–20, Feb. 2003.

[17] Mathstar. reference: http://www.mathstar.com/.

[18] OpenCores. reference: http://www.opencores.org/. 2007.

[19] A. Rahman, S. Das, A. P. Chandrakasan, and R. Reif. Wire Requirement and Three-Dimensional Integration Technology for Field Programmable Gate Arrays. *IEEE Trans. on VLSI*, 11(1):44–54, Feb. 2003.

[20] J. Rose and S. Brown. Flexibility of Interconnection Structures for Field Programmable Gate Arrays. *IEEE Journal of Solid-State Circuits*, 26(3):277–282, 1991.

[21] J. Swartz, V. Betz, and J. Rose. A Fast Routability-Driven Router for FPGAs. In *International Symposium on Field Programmable Gate Arrays*, pages 140–149, 1998.

[22] P. Verplaetse, D. Stroobandt, and J. V. Compenhout. A Stochastic Model for the Interconnect Topology of Digital Circuits. *IEEE Trans. on VLSI*, 9(6):938–942, 2001.

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

[24] A. Ye. *Single Driver Code in VPR*. Private Communication, 2005.