POST-ROUTING ANALYTICAL MODELS FOR HOMOGENEOUS FPGA ARCHITECTURES

by

Yoon Kah Leow

A Dissertation Submitted to the Faculty of the

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

In Partial Fulfillment of the Requirements
For the Degree of

DOCTOR OF PHILOSOPHY

In the Graduate College

THE UNIVERSITY OF ARIZONA

2013

THE UNIVERSITY OF ARIZONA
GRADUATE COLLEGE

As members of the Dissertation Committee, we certify that we have read the dissertation prepared by Yoon Kah Leow entitled Post-Routing Analytical Models for Homogeneous FPGA Architectures and recommend that it be accepted as fulfilling the dissertation requirement for the Degree of Doctor of Philosophy.

Date: 30 October 2013
Ali Akoglu

Date: 30 October 2013
Susan Lysecky

Date: 30 October 2013
Roman Lysecky

Date: 30 October 2013

Date: 30 October 2013

Final approval and acceptance of this dissertation is contingent upon the candidate's submission of the final copies of the dissertation to the Graduate College.

I hereby certify that I have read this dissertation prepared under my direction and recommend that it be accepted as fulfilling the dissertation requirement.

Date: 30 October 2013
Dissertation Co-Director: Ali Akoglu

Date: 30 October 2013
Dissertation Co-Director: Susan Lysecky

STATEMENT BY AUTHOR

This dissertation has been submitted in partial fulfillment of requirements for an advanced degree at the University of Arizona and is deposited in the University Library to be made available to borrowers under rules of the Library.

Brief quotations from this dissertation are allowable without special permission, provided that accurate acknowledgement of source is made. Requests for permission for extended quotation from or reproduction of this manuscript in whole or in part may be granted by the head of the major department or the Dean of the Graduate College when in his or her judgement the proposed use of the material is in the interests of scholarship. In all other instances, however, permission must be obtained from the author.

SIGNED: Yoon Kah Leow

ACKNOWLEDGEMENTS

I would like to thank my supervisors, Professors Ali Akoglu and Susan Lysecky for their patience and guidance. Professor Ali Akoglu is truly dedicated in guiding me, from both an academia's standpoint and beyond, by providing moral support for my family woes. Despite not working directly under her lab, Professor Susan Lysecky has given me constant research insights and guidances through these four years. I am proud to say that I have learnt significantly from the both of them.

I would like to thank all my friends and colleagues in the Reconfigurable Computing Laboratory (RCL) for sharing their enjoyable moments and filling me with innovative ideas as I indulged myself in the state-of-the-art research.

I would like to thank my family, especially, my brother, Yoon Hwee Leow, and my parents-in-law for their unwavering support and trust in me for pursuing overseas graduate studies.

I would like to thank my parents for dedicating their life and time to my education and personal upbringing. I will most like to share this achievement with them.

Last but not least, to my wife for her unconditional love and support through these four years. Those times that I shall always remember, that knitted us much closer together.

DEDICATION

Dedicated to my beloved wife, Liu Qi and my adorable daughter, Hong Wen and son, Hong Jun.

TABLE OF CONTENTS

TABLE OF CONTENTS – *Continued*

LIST OF FIGURES

LIST OF FIGURES – *Continued*

LIST OF TABLES

ABSTRACT

The rapid growth in Field Programmable Gate Array (FPGA) architecture design space has led to an explosion in architectural choices that exceed well over 1,000,000 configurations. This makes searching for pareto-optimal solutions using a CAD-based incremental design process near impossible for hardware architects and application engineers. Designers need fast and accurate analytical models in order to evaluate the impact of their design choices on performance. Despite the proliferation of FPGA models, todays state-of the art modeling tools suffer from two drawbacks. First, they rely on circuit characteristics extracted from various stages of the FPGA CAD flow making them CAD dependent. Second, they lack ability to take routing architecture parameters into account. These two factors pose as a barrier for converging to the desired implementation rapidly. In this research, we address these two challenges and propose the first static power and post-routing wirelength models in academia. Our models are unique as they are CAD-independent, and they take both logic and routing architecture parameters into account. Using the static power model we are able to estimate the active and idle leakage power dissipation in homogeneous FPGAs with average correlation factor of 95% and mean percentage error of 17% over experimental results based on MCNC benchmarks. Using our wirelength model, we are able to obtain a low mean percentage error of 4.2% and an average correlation factor of 84% using MCNC and VTR benchmarks. We also show that utilizing wirelength model for architecture optimization process reduces the design space exploration time by 53% compared to the CAD-based process. We finally propose an algorithmic approach to estimate the logic density (i.e., number of LUTs) of multiplexer-based circuits, and address the problem of discrete effects in FPGA analytical models. We show that a model that generates logic density of a fundamental circuit element, such as a multiplexer, can be used to estimate

performance metrics, such as critical path delay and power.

CHAPTER 1

Introduction

1.1 Overview

Field Programmable Gate Arrays (FPGAs) are an interesting genre of Very Large Scale Integration (VLSI) circuits that require a mixture of programmable metal dominated routing interconnects and silicon based logic blocks/hard-IPs in order to support post-fabrication reconfiguration.

Designing a new FPGA device requires an application-related suite of benchmark circuits to be synthesized to evaluate a new architecture change. Therefore, it is not surprising to find the FPGA industry and academia community relying heavily on CAD tools to design/evaluate a new FPGA device. Inevitably, this entanglement with CAD tools to complement architecture development is a burdensome process. In today's context, designing a modern FPGA architecture encompasses a huge architecture design space exploration which made searching for architectural optimal pareto configurations (i.e., more than 1,000,000 configurations) using the conventional CAD based incremental design and optimization process no longer economical. The growing gap between increasing FPGA architecture complexity and long CAD run times can only be bridged through architecture modeling. The aim is to allow hardware architects and application engineers to evaluate the impact of their design choices with model-based trend analysis and rapidly converge to a desired solution without having to launch extensive CAD-based experiments. Despite the proliferation of FPGA models, todays state-of the art modeling tools suffer from CAD tool dependencies. Furthermore, the lack of routing information prevents users from performing post-routing architecture evaluations. In this research, we address these two issues and present the first ever post-routing wirelength and static power models. Furthermore, we address an inherent deficiency in analytical

modeling targeting multiplexer circuits.

It is a well-known fact that, increasing the logic density with technology scaling has an adverse effect on static (leakage) power. Companies, such as Xilinx, has indicated that static power can be as high as dynamic power in their 28nm technology devices. In this dissertation, we present the first static power model in academia which estimates leakage power dissipation in homogeneous FPGAs. In our approach, we take both logic and routing architecture parameters into account and show that our model achieves high fidelity with an average correlation factor of 95%. In fact, the model only deviates from the actual CAD based experimental results with a mean percentage error of approximately 17%.

Modeling wirelength is a critical step since channel width can be expressed as a function of total net length in a design, which is an indicator of routability for an FPGA. Performance metrics, such as critical path delay and power consumption are also functions of net capacitance, which in turn is a function of net length. However, current wirelength models are inadequate as they operate based on an unrealistic assumption that FPGAs have unlimited routing resources to provide an optimal routing solution. Hence, in this dissertation, we introduce a post-routing wirelength model which takes into account of both logic and routing architectural parameters. Using MCNC and VTR benchmarks, we obtain a low mean percentage error of 4.2% and an average correlation factor of 84% We also show that utilizing wirelength model for architecture optimization process reduces the design space exploration time by 53% as compared to the conventional CAD tool based process.

Finally, we propose an algorithmic approach to estimate the logic density (i.e., number of LUTs) of multiplexer-based circuits and address the problem of discrete effects in FPGA analytical models. We show that a model that generates logic density of a fundamental circuit element, such as a multiplexer, can be used to estimate performance metrics, such as critical path delay and power.

In essence, enabling rapid performance evaluations is invaluable considering the increasing complexity of current applications, logic density and hardware heterogeneity (i.e., hard/soft microprocessors, Digital Signal Processing (DSP) sys-

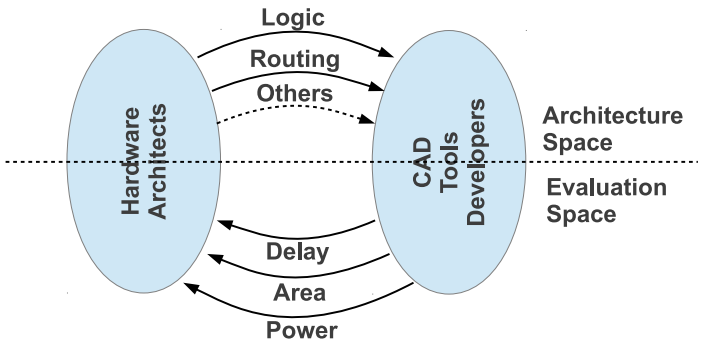
tems, Application-Specific Standard Products (ASSPs) and Application-Specific Integrated Circuits (ASICs)) of modern FPGAs. Example applications of these models are in areas of (1) expediting early architecture design space exploration, (2) evaluating application/algorithm performance specific to FPGA architecture and (3) complementing FPGA CAD tools.

1.2 Motivation

Conventionally, due to its re-programmable nature, Field Programmable Gate Array (FPGA) hardware architects are always subjected to rely on a set of established Computer Aided Design (CAD) tools to investigate the impact of their design choices on performance. Similarly, CAD tool developers rely on architecture configuration for effectively utilizing the available resources and meeting the design objectives. This symbiotic relationship between CAD tool developers and FPGA architects is illustrated in 1.1. Although CAD tool based performance evaluation is a reliable method, the tight-coupling between these domains is no longer sustainable due to the rapid increase in the logic density and complexity of the FPGA architectures as well as the increase in the size of the applications and their demand for heterogeneous set of resources.

In the following sentences, we highlight the recent progress in design space that contributed to this complexity increase. Traditional FPGA architectures (i.e., an

Figure 1.1: Tight coupling between hardware architect and CAD tool developers.



island-style FPGA which consists of a 2D-array of Configurable Logic Blocks (CLBs) connected via the routing interconnect) are relatively simple with approximately 8 major configuration parameters (i.e., logic: inputs per LUT, inputs per logic block, number of LUTs per logic block; and routing: connection-box and switch-box flexibilities, routing segment length, channel width). Despite so, for an architect deriving an optimal configuration for the target application(s) requires sweeping the entire architecture design space consisting of (8!) 40,320 permutations[1]. In modern architectures, such as the multi-bit routing architecture proposed by Ye et al. [4], to support bus-transactions in data-path related circuits, 19 architecture parameters have been identified! Following the inflation of the architecture configuration space, another evolving trend is the increasing logic density in today's reconfigurable devices. This is attributed to both the shrinking technology feature sizes and a well-known technique of integrating hardIPs into the sea of programmable CLBs. Ever since the first manufactured device by XILINX (XC2064), which only had 64 configurable logic blocks (CLBs), each with two 3-input Look-Up Tables (LUTs), today's high-end FPGAs are capable of housing over 500,000 logic cells on a single die. In the upcoming 20nm technology milestone, FPGA companies such as Altera also revealed new product features that include heterogeneous system hardware modules such as soft microprocessors, Digital Signal Processing (DSP) systems, Application-Specific Standard Products (ASSPs) and Application-Specific Integrated Circuits (ASICs) that can even enable System-on-Chip (SoC) platforms to be implemented [40]. Moreover, the desirable application genre that is supported by these heterogeneous FPGA architectures are much larger than the conventional Microelectronics Center of North Carolina (MCNC) benchmarks. The largest MCNC benchmark with 14,253 gates (*clma*) is nothing as compared with the latest TITAN23 benchmarks [36] which requires 1,859,501 logic blocks (*gaussianblur*). It is important for the selected benchmarks to reflect the modern applications of today's FPGA devices. Unfortunately, Murray et al. identified that even the most up-to-date academia available tool (VTR-7.0) is too slow for some of these benchmarks (i.e.,

[1] Architecture configuration parameters are usually correlated to each other [19]

Out of 23 benchmarks, 6 of them require more than 48 hours and 3 benchmarks are unroutable!) [36].

With this complexity explosion and the tight coupling between the fabric architects and CAD tool designers, FPGA community faces a steep challenge for the advancement of reconfigurable technologies and CAD tools. Particularly, growing architecture design space combined with slowing CAD tool trends turn into an impetus for the FPGA community. The dire question that awaits is, what can be done to bridge this gap and re-enable architects on evaluating the performance of hypothetical architectures for the target application domain, designing and developing the next generation architecture that meets the performance requirements of emerging applications rapidly. The answer is, *Modeling*!

1.3 Model's impact on architectures, CAD tools and applications

In the following paragraphs, we justify the benefits of modeling based performance evaluation from CAD tool developers, FPGA architects and as well as application engineers perspectives.

CAD Tool Developer: 3D FPGAs open the opportunities for (1) increased logic density and (2) reduction in routing overheads as compared to 2D FPGAs with the grand challenge of improving the energy efficiency of the system architecture.

Let us take 3D FPGA architecture design space as an example. Today, there is no evidence of a large proliferation of these devices in the market yet. One of the contributing factors to this slow adoption of 3D technology in FPGA is the tight-coupling problem between architecture and CAD tools. Currently, there is no reliable source of a typical 3D-FPGA architecture that can facilitate as a baseline reference for CAD tool development. Despite large FPGA companies such as Xilinx are deploying 3D integration techniques for their latest series of Virtex-7 devices, with the lack of CAD support, they are still strictly 2.5D (i.e., having separate single die stacks as opposed to a stack of dies in a single package). In the academia domain, there are efforts in developing CAD tools for 3D-FPGAs [9][37][35][10]. However,

these are mainly meant for homogeneous 3D-FPGAs (i.e., each die takes after a conventional homogeneous FPGA that only consists of a 2D array of CLBs).

These tools are built based on the 2D principles by forcing the design to be partitioned into n-layers first and then applying 2D based CAD flow on each layer individually targeted for a homogenous FPGA architecture. Furthermore, the true capabilities of 3D integration, which is the ability to incorporate dies with different technologies on a single stack in order to bring external system memory closer to actual computation units to ameliorate communication bottlenecks can not be truly explored with these tools. This is because they do not support heterogeneous FPGA architectures that require integration of hardIPs into the general reprogrammable fabric. In short, the lack of a reliable suite of CAD tools is becoming an obstacle to develop truly novel 3D FPGA architectures. There is a dire need for tools to facilitate 3D FPGA architecture exploration and development. We postulate that, modeling-assisted architecture exploration can help to address these deficiencies. Specifically, armed with the right modeling tools, architects can start with defining the architecture configuration space (i.e., identifying the critical parameters of a target FPGA architecture) to facilitate CAD tool development. In this manner, the tight-coupling between architecture and CAD tool development will be breached.

FPGA Architects: To provide a good picture of modeling-assisted architecture development, we present two application examples referring to the state-of-the-art works based on multi-bit routing fabric [4] and 3D-FPGA switch-box architecture [1], and describe how modeling could change the design process.

Example 1: Using Bus-Based Connections to Improve FPGA Density for Implementing Datapath Circuits

Ye et al. [4] proposed a multi-bit routing interconnect fabric to handle bus-based routing for data-path circuits. The authors evaluated their architecture with a suite of data-path oriented CAD tools and demonstrated good area performance. Since both the architecture and CAD tools are designed specifically for data-path circuits, it is important to ensure that the proposed architecture is not overly-biased and scales well to support single-bit based circuits in order to minimize perfor-

mance degradations with sub-optimal (i.e., single-bit oriented) circuits. From the architect's point of view, analyzing the performance of a multi-bit routing fabric evaluated over a *general* benchmark suite (i.e., including circuits that have fewer data-path circuitries) can provide insights into how the architecture can be optimized. Unfortunately, due to the large number of architecture configuration permutations (i.e., 19! permutations) with a huge set of benchmarks to explore, it is very tedious (i.e., might require concurrent CAD tool modifications to optimize a large genre of applications with respect to the proposed multi-bit routing fabric) and time-consuming to derive the *pareto* architecture configurations of a large variety of benchmarks. Since architecture search space is extremely large, researchers are forced to conduct their investigations based on a subset of architecture configurations only. Assuming that post-routing wirelength for a heterogeneous architecture is available, the authors can start by deriving an optimal single-bit architecture configuration. Thereafter, using the derived configurations as a baseline, further fine-tuning of the multi-bit routing aspects of the architecture can be carried out. This is possible due to the tight correlation between the average wirelength of the circuit and the routing area. By doing so, the authors can ensure that the overall derived architecture configurations for the multi-bit architecture is able to cater to a much wider genre of applications!

Example 2: Designing a 3-D FPGA: Switch-box Architecture and Thermal Issues

Gayasan [1] proposes five types of switch-boxes, (1) subset, (2) subset-split, (3) subset-twist, (4) universal-twist and (5) universal-more. These are evaluated using the 20 MCNC benchmarks based on their performance with respect to the minimum channel width, area, delay and area-delay-product. Since the 3D-FPGA architecture is homogeneous (i.e., each die layer is identical), the academia available CAD tool, VPR, can be easily modified to incorporate the proposed switch-box architectures. Similar to the approach from Ye et al., a default set of parameters are used to evaluate the 3D switch-box architectures. However, using a default switch-box flexibility, F_S, will prevent an architect from studying the trade-off between F_S and area with delay. This is important because, changing F_S will quickly impact the

delay and area performance. Specifically, increasing the flexibility might increase the area of each switch-box (i.e., number of switches to support the connectivity), but, might decrease the routing wirelength, which implies that fewer switch-boxes are required! Furthermore, different switch-box architectures might be more suitable for different application genres. Unfortunately, to explore the full architecture design space with all flexibility possibilities is exorbitant. Even with a homogenous static power model , one can estimate the number of routing resources (i.e., tri-state gates, routing multiplexers, buffer drivers, etc) with minor modifications to the architecture layer[2]. It is anticipated that the critical path delay of a circuit will decrease with increasing routing flexibilities. Hence, one can use the post-routing wirelength model and analyse its trend by sweeping F_S The region where the average wirelength has large changes will correspond to a significant impact on the critical path delay. Then, CAD experiments can be conducted focusing only on that region of interest. With the availability of heterogeneous static and dynamic power models, one can investigate the optimal partitioning of an application among the 3D FPGA die stack to achieve the lowest power performance.

Application Engineers: For the past 5 years, the results of the ground-breaking study by Kuon et al., *Measuring the Gap Between FPGAs and ASICs* [40], lauded the approach of incorporating ASIC hardIPs into the general FPGA fabric. But recently, Dehon in his paper, *Location, Location, Location The Role of Spatial Locality in Asymptotic Energy Minimization* [21], reveal that there is a cost associated with the spatial locality of hardIPs disguised in the form of communication bottlenecks. Due to the difficulty of finding ways to take advantage of the spatial locality offered by modern FPGAs, the communication bottlenecks have strong implications on the major performance metrics (i.e., power, delay, area). The conclusion drawn from Dehon's work greatly stresses the importance of applying architecture-specific knowledge on FPGA based application mapping, and the fact that spatial locality should be taken into account amplifies the burden and complexity of this mapping process.

[2]Assume that the partitions among the die layers are relatively similar.

In this context, models can be applied to lighten the burden on application engineers. Assume that our objective is to evaluate the tradeoffs between fine-grained and coarse-grained workload partitioning with respect to power, area and delay. Fine-grained partitioning implies high-degree of parallelism through replication of the resources, and coarse-grained partitioning implies recycling individual functional blocks with more number of execution iterations. To complicate the task, the constraints could as well be contradicting. From energy consumption point of view, while one might wish for an application to dissipate as little power as possible by instantiating fewer functional blocks, increasing the amount of computation by parallelizing the workload might result in a shorter overall execution time! Using the conventional approach, locating the *pareto* configurations for these contradicting objectives will require conducting large scale CAD experiments. Assuming that area, power and critical path delay models which capture the hardIP area and block distribution parameters are readily available, application engineers can replace CAD based experiments with these models for trend analysis based on their target application. This will enable application engineers to quickly converge to an optimal design configuration without having to launch extensive CAD flows.

1.4 Model Validation

Since models are mainly constructed to target architecture development work, one might ask the question of how models can be validated without a clear definition of an architecture and a suite of suitable CAD tools in the first place? In other words, can models still help to leverage the interdependency between architecture development and CAD tools? The objective of modeling is to capture the performance trends with respect to architecture variations. The main difference from the former approach of having CAD tools to complement architecture development process directly is that, modeling is not meant to reproduce the exact empirical output of a CAD tool. In other words, instead of creating a full-fledge CAD flow, only minor modifications are required to validate the models. For example, with-

out an actual CAD flow for a hypothetical 3D-FPGA architecture, the placement constraints due to the physical distribution of Through Silicon Vias (TSVs) can be simulated by having the clustering algorithm to loosely cluster and not maximizing the occupancy of each logic block.

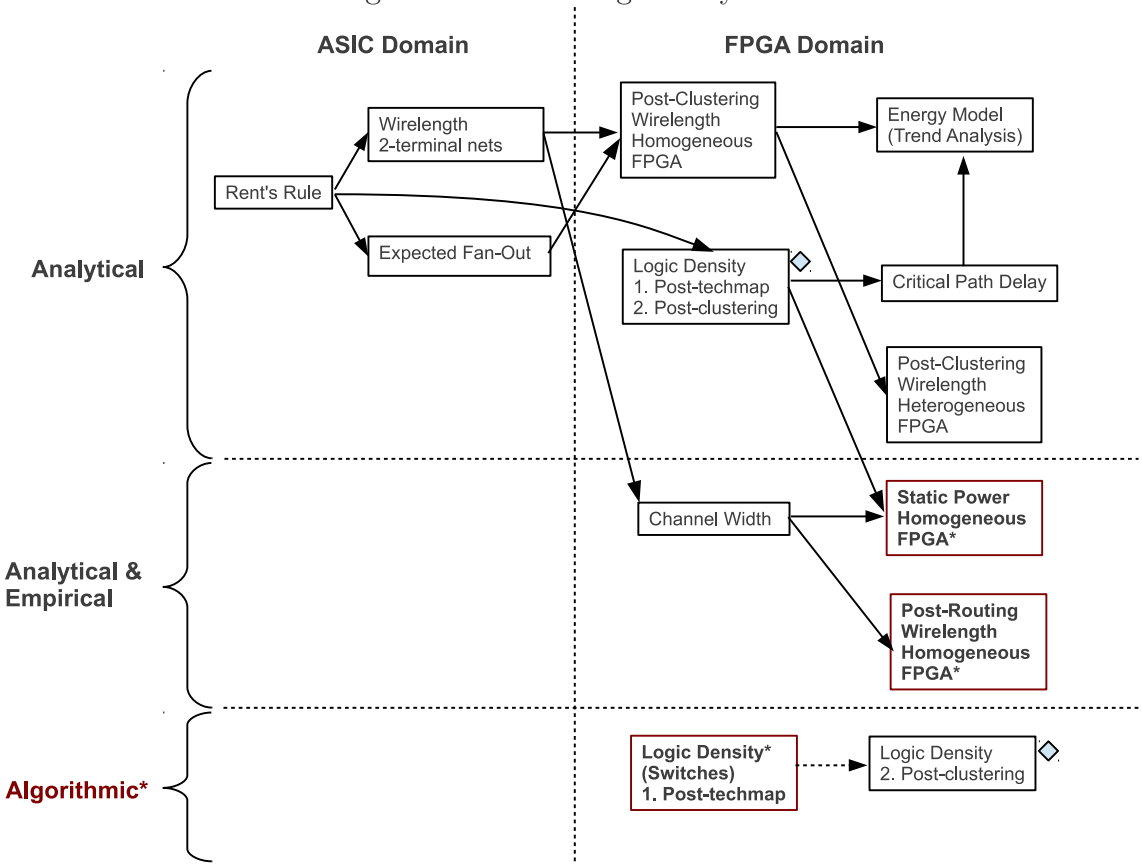
Moreover, existing models also function well as good baselines for modifications to capture architecture modifications and perform first-hand performance analysis. The benefits of this approach is tremendous. By trimming the design exploration space, so that the actual time-consuming CAD experimentations are conducted only on a much smaller hand-full of potential architectures, studies have shown that up to 80% of the design exploration time can be eliminated [19]!

1.5 Dissertation Contributions

In recent years, to aid FPGA development in the aspects of (1) architecture, (2) application porting and (3) CAD flow optimizations, numerous FPGA models (e.g., delay[28], wirelength[3], energy[55], logic density[18], routability[60][20]) have emerged from the academia domain. These models are largely derived based on analytical and/or empirical approaches. In the analytical approach, FPGA models can be traced back into the ASIC domain where models are mostly originated from the Rent's rule [42]. In the empirical approach, models establish the relationship between the architecture configuration parameters with the target performance metrics (i.e., power, delay, area) via trend analysis by conducting extensive CAD experiments.

Figure 1.2 depicts the family tree of models including the works of this dissertation. In large, the models can be categorize into the ASIC and FPGA domain. As the main focus of this dissertation is in FPGA domain models, the ASIC domain models as illustrated in figure 1.2 is not exhaustive. The main idea is to show the point of penetration when the ASIC models can be extended into FPGA models. Besides the 2 main genres of approaches (i.e., purely analytical, coupling both analytical and empirical), this dissertation also propose an additional genre, *algorithmic*

Figure 1.2: Modelling Family Tree



approach. In general, previous FPGA models suffer from two deficiencies:

FPGA models are CAD dependent. In this dissertation, CAD dependencies refer to the requirement of launching extensive CAD flows for parameter extraction purposes before the actual model can be deploy. Specifically, models use Rent's parameters that are extracted after the placement stage. Hence, these parameters will vary with different mapping, clustering, and placement tools for a same input circuit. Although parameter extraction is only required once per circuit, this process requires extensive amount of time, considering the number of circuits that need to be studied. This tight coupling limits the portability of these models into architecture evaluation tools. In this dissertation, we ensure that our models are minimally CAD dependent by validating our results using pre-technology mapped Rent's parameters. This decouples our models from the CAD flow while ensuring that the models are accurate with minimal CAD interventions.

FPGA models are based on post-placement stage. Specifically, the routing interconnect architecture is neglected in their model construction. On the contrary, due to the inherent reconfigurable nature of FPGAs, these devices need to support large number of routing channels/tracks that are connected by routing switches/multiplexers/tri-state buffers, during post-fabrication configuration. These routing switches are the main contributors of delay (i.e., loading capacitance due to pass-transistor switches/driving stage buffers, metal resistance in the channel tracks) which is often estimated in terms of a RC (i.e., Resistance and Capacitance) factor. Ye et al. quantified that for a typical channel widths in the range of 20 to 40, 55% to 65% of the total FPGA area is occupied by the routing programmable switches. Moreover, with an increase in Rent's exponent due to larger fan-ins/outs of ASIC hardIP blocks in heterogeneous FPGAs, this will also lead to an increase in routing switches to establish connectivity with the routing channel tracks. Furthermore, unlike the silicon dominated diffusion layers in CLBs, these large number of metal routing tracks are not prone to technology process scaling. Hence, routing overhead will increase with shrinking technology feature sizes. In this dissertation, models are constructed to take into account of the routing architecture parameters.

In this manner, our models can be used for both logic and routing architecture development and evaluation.

This dissertation focuses on three overarching themes:

1. Early prediction of static power consumption in homogeneous FPGAs.
2. Post-routing wirelength estimation of final netlist in homogeneous FPGAs.
3. Alternative modelling approach based on multiplexer-based circuits to address current modelling deficiencies.

Chapter 2 presents the first static power model in academia which estimates the leakage power dissipation in homogeneous FPGAs. An infamous effect of shrinking technology device sizes is the increase in static (leakage) power. Leakage power is the power dissipated when the transistor is in an idle (quiescent) state. When the channel length is long (>130nm), majority of the leakage power is composed of sub-threshold leakage component. As the device gets smaller (>90nm), the gate tunnelling effect gets amplified and needs to be taken into account. Going to even smaller device sizes (<90nm), the junction leakage also needs to be considered. Furthermore, it is inherent for a FPGA device to be prone to logic redundancies. Since the leakage power is proportional to the number of transistors in a chip, FPGA is highly susceptible to a large leakage power and this emphasizes the importance of keeping track of the static power when designing a new application or evaluating a new architecture.

Chapter 3 presents the first post-routing wirelength model for a homogeneous FPGA architecture. In this dissertation, the static power model accounts for the number of quiescent transistors in the routing resources (i.e., connection-boxes and switch-boxes) using an estimation strategy based on the number of utilized *logic blocks* [3]. This is under the assumption that the FPGA has excess routing resources to provide an optimal routing solution. However, occasionally, this is not the case

[3] *Logic blocks* refers to configurable logic blocks (CLBs) in the case of homogeneous FPGAs. In general, they can also refer to ASIC-style hard-ips in the case of heterogeneous FPGAs.

in a realistic/commercial FPGA. Hence, a more scalable approach to estimate non-utilized routing resources is to estimate the post-routing wirelength of the implementation. The post-routing wirelength model captures the true wirelength of a circuit's final netlist taking into account of routing congestion due to a constrained routing interconnect architecture. Analyzing the post-routing wirelength allows a designer to evaluate an architecture's routing capability. Since the routing fabric constitutes a major portion of the circuit delay [30], it can also shed insight into the overall circuit delay, power and area of an implementation with respect to an architecture.

Chapter 4 considers the deficiency in current modelling approaches and propose a new *algorithmic* modelling approach that addresses the problem of *discrete effects* for logic density estimation in multiplexer-based circuits. Current models are expressed in mathematical equations which comprise of the architecture parameters to capture the expected performance trends. These functions provides a best fit *continuous* curve to describe a FPGA architecture that is composed of a limited finite number of *discrete* circuit elements (i.e., multiplexers, buffers, tri-state gates, level-restorer, etc). Therefore, due to the *discrete* nature of the FPGA, it is inevitable to have deviations between the model and actual experiment outputs. Das et al. refer to these deviations as the *discrete effects* [19]. One of the circuits that are susceptible to *discrete effects* are multiplexers.

Multiplexers are common logic primitives that are found in a wide number of circuits. A study conducted by Altera revealed, in 120 commercial benchmark circuits, an average of 25% of these circuits are composed of multiplexers [14]. Furthermore, with the proliferation of porting large arithmetic kernels targeting floating-point type computations (e.g., 32, 64-bits) on FPGA devices, it is inevitable for a circuit data-path to support wide-bus multiplexing. These large multiplexers are notorious for occupying a large amount of logic resources. Therefore, a new modelling approach, *algorithmic*, based on algorithms is proposed to estimate the number of utilized LUTs (i.e., post technology-mapping stage) with respect to multiplexer circuits. The algorithm's fidelity is demonstrated by applying the technique in large

crossbar switches. This multiplexer logic density estimation model can also be extended by incorporating it into other estimation models (e.g., static power, critical path delay) in the family.

Chapter 5 presents the concluding remarks and potential research extensions for future work.

CHAPTER 2

Homogeneous FPGA Static Power Model

2.1 Introduction

Studies have shown that FPGAs consume 7∼14X more power than their ASIC counterparts [40]. Historically, the dynamic power has been the focus as it has dominated the total power consumption. Recent studies have shown that the static power is fast approaching the dynamic power in sub-micron devices [58]. In fact, the static power can be as high as dynamic power in 28nm technology devices [29]. Therefore, as the process technology shrinks, static power consumption has become a critical performance concern during system design [74].

In this chapter, we propose an analytical static power model for evaluating homogeneous island-style FPGAs by taking both logic and routing architecture parameters into account. We first estimate the logic and routing resource utilizations based on the circuit characteristics using existing models. We then devise a method to map the total resource utilization to the amount of static power dissipated for a given architecture. We keep resource utilization calculations in an abstract form independent from the device technology. This allows our model to be reused for other devices. Complete independence from the CAD tools and simplicity of the model make our approach portable for architecture evaluation environments. We study the quality of our model by sweeping all the architectural parameters individually and calculating the Correlation Ratio (C-Ratio) and the Minimum Absolute Percentage Error (MAPE) metrics with respect to the experimental results generated by the Versatile Placement Routing (VPR) tool [47]. We evaluate the effectiveness of our model using the MCNC benchmarks, and demonstrate high fidelity with an average correlation factor of 95%. In fact, the model only deviates from the actual empirical value with a mean percentage error of approximately 17%.

The chapter is organized as follows. In section 2.2, we present the related work on FPGA power models. In section 2.3, we describe the proposed static power model and our development approach. In section 2.4, we describe the methodology that is used to validate the model, while section 2.5 presents the performance analysis.

2.2 Literature Review

The origins of FPGA power modeling can be traced back to VLSI power estimation techniques. Back in the earlier device generations, when the meagre leakage power was considered insignificant compared to the dynamic power, researchers mainly focused on approximating the input signal transitions in order to predict circuit activity. Najm et al. summarize these techniques [50], and categorize them as probabilistic techniques (i.e., Signal Probability [15], CREST [48], DENSIM [49], BDD [26], Correlation Coefficients [68]) and statistical techniques (i.e., McPower [8], MED [75]).

In the probabilistic techniques, sequential circuits are neglected, and only combinational circuits are taken into account. The user defines the probabilistic signals/waveforms for the nodes of a circuit, and these signals are propagated through the individual circuit nodes. This propagation based signal flow is then used to generate an estimation of the likelihood of a signal transition. Then, the total power is estimated by accumulating the average power calculated for each node. One example of such a technique is, DENSIM [49]. It makes use of *transition density*, which predicts the average number of signal transitions per unit time for a single node. In the statistical techniques, simulators are used to model the execution of the circuits based on a combination of input vectors. Thereafter, the simulations are monitored to retrieve the power values. This approach requires the simulators to run extensively until the measured power converges into the average power.

In the FPGA domain, probabilistic techniques are preferred over the statistical techniques attributed to their computational efficiency. The first FPGA power model proposed by Poon et al., employs the *transition density* signal model. In

Poon's approach, the architecture description file, the user circuit, along with outputs of post-placement, and routing stages' outputs [54] are used to derive the power model. The model, which is integrated into the VPR CAD tool, is capable of estimating power for a large variety of island-style based FPGA architectures. In this method, total FPGA power is defined in terms of logic, routing, and clock power. Li et al. in [44] introduces an FPGA architecture evaluation framework which is capable of providing cycle-accurate power simulations for multiple FPGA architectures. This framework includes detailed signal transition studies by taking signal glitches into account. A combination of switch-level models for interconnects, and macro models for LUTs are employed. The model takes in a decorated netlist consisting of circuit capacitances, and delays. This netlist is generated based on a suite of CAD tools consisting of VPR [7], RASP [13] and SIS [64]. Despite the flexibility, and accuracy of these models, one drawback they carry is their dependency on the CAD flow. Tight coupling between long running CAD experiments and the power estimator makes it harder for the designers who need to conduct design space exploration of large set of hypothetical architectures or to narrow down the architecture parameter search space for detailed optimization studies.

One area of work which can also decrease the CAD tool runtime is incremental CAD flows. These tools enable the original netlist to be modified with specific to the modifications and periphery that is impacted by these changes. By doing so, the amount of CAD flow runtime is greatly reduced. One example of such an approach is incremental placement, proposed by Singh et al. [65]. Their work reduces the placement complexity with respect to circuit restructuring techniques by addressing the blocks that resides on the critical path. Hence, these tools are very useful when modifications to the netlist are required. Despite that incremental synthesis only requires a minimal amount of runtime during the logic synthesis stage to reflect the modifications, it is best applied when consecutive minor changes are expected targeting the same application. Our work differs by focusing on an analytical approach, geared towards analysing trends of static power with respect to architecture modifications through estimation means. Hence, our goal strives by

providing fast estimations independent of CAD flows.

In [55], Rajavel et al. integrates the wirelength, and critical path delay models for FPGA energy modeling. By utilizing the heterogeneous wirelength model based on Davis' [31][32] and Das' [18] work, this model is capable of approximating the energy of a heterogeneous architecture. Furthermore, the authors eliminate the CAD flow dependencies by pre-calculating the Rent's exponents based on the pre-technology mapped netlist. Even though the model is able to capture the energy efficiency of a set of hypothetical architectures with respect to each other, it is unable to produce good accuracies, since the model ignores the actual circuit components, signal switching activities and supply voltages during estimation.

In this work, we integrated existing FPGA channel width, and wirelength models into Poon's power model to enable fairly accurate and CAD independent power estimation. Using an area model that approximates the total number of minimum width transistors, we calculate the logic and routing power for a given circuit. We show that the results highly correlate with the original Poon's power model, while greatly increasing computation efficiency, as we eliminate the need for information from the placement and routing stages.

2.3 Static Power Modelling for Homogeneous FPGAs

2.3.1 Model Development Overview

Island-based, homogeneous FPGA architecture by Xilinx and Altera is the target of our modelling work. The total static power is estimated by modelling the leakage power dissipated by both logic (i.e., configurable logic blocks), and routing resources (i.e., connection-boxes, and switch-boxes). We take routing switches that reside within the logic blocks into account when modelling the logic block static power. Many of the main components in a FPGA device such as the LUT and routing switches (i.e., connection or switch boxes) takes after a multiplexer circuit. In the model, we assume a pass transistor based architecture. However, existing patents have revealed that commercial FPGAs from companies such as Xilinx [52]

are adopting a transmission gate approach to design their device components for increased reliability, especially for devices in smaller process nodes. We believe that our model is adequate, as it is split into two layers (i.e., architecture and device specific). Hence, the change from pass transistor to transmission gate can be easily incorporated into our model. Similarly, although we are using the 180nm technology device specifications for our modelling purposes, our model's abstraction layers enable modifications on the device layer without affecting the architecture layer. Our model incorporates the SRAM leakage from configuration bits in the model estimation based on a SRAM architecture with 6 transistors (i.e., 4 NMOS and 2 PMOS transistors). Evaluating the contribution of SRAM cells to the static power and evaluating the full impact of state-of-the-art SRAM circuit techniques with respect to FPGA architectures would be one of the extensions of the work propose in this dissertation.

Leakage power consists of the three main components, sub-threshold, reverse-bias and gate tunnelling leakages. Since our model is based on the 180nm process node, majority of the leakage power is composed of the sub-threshold leakage [2]. Currently, the reverse-bias and gate tunnelling leakages are considered negligible. Similar to Poon's power model [54], the sub-threshold leakage is estimated using a first-order estimation model [61]. The average error between Kang's model and HPSICE simulation for individual transistor (i.e. NMOS or PMOS) is estimated to be approximately 13.4% [54]. Future work will include integrating other leakage components (i.e., reverse bias and gate tunnelling leakages) into the model's device layer to add support for different technology nodes.

We extend upon Das' area equations [18] and Fang's channel width model [23] to estimate the logic and routing resource utilizations respectively. After the total routing resource utilizations, we estimate the static power analytically.

Sections 2.3.2, and 2.3.3 will provide detailed analysis on the formulation of the static power of routing and logic resources.

Logic Density (Lam) & Channel Width (Fang) Model Parameters	
p	Rent's exponent
γ	Average number of unused inputs of a LUT
n_2	Number of 2-input gates required for a given circuit
n_k	Number of K-LUTs required for a given circuit
n_c	Number of clusters required for a given circuit
c	Average number of LUTs packed in a cluster ($c = \frac{n_k}{n_c}$)
ρ	Inverse channel width utilization ratio ($\rho = \frac{W_{avg}}{W_N}$)
λ	Average number of inputs used in a logic block
i	Average number of inputs used in a cluster
o	Average number of outputs used in each cluster
f_{avg}	Average fanout of all nets for a given circuit
W_{avg}	Average number of used wires per channel
W_N	Minimum required channel width to route a given circuit
W_{am}	Absolute minimum channel width for a given circuit
FPGA Architecture Parameters, Range [Default]	
K	LUT size, 2∼7 [4]
N	Logic block size, 2∼20 [10]
I	No. of logic block inputs, 4∼40 [22]
$F_{C_{in}}$	Input connection box flexibility, 0.1∼1.0 [0.25]
$F_{C_{out}}$	Output connection box flexibility, 0.1∼1.0 [1.0]
W	Channel width, $W = W_N \times (1 + CW_I)$, $CW_I \in 10 \sim 200\%$ [10%]
L	Segment length, 1∼6 [4]
F_S	Switch box flexibility, [3]
Eqv	Logical Equivalence, [1]

Table 2.1: Power Model Parameters

2.3.2 FPGA Routing Static Power

Our routing static power model depends on finding:

- the average number of connection-boxes and switch-boxes,
- the average resource utilization of routing multiplexers, buffers and SRAM configuration bits within the switch-boxes and connection-boxes.

Our model operates based on the following assumptions:

- Routing switches are composed of multiplexers, buffers, and SRAM configuration bits as described in [43].
- The minimal sized FPGA with respect to the circuit is always available to the user.
- The router searches for a path within the bounding box instead of an exhaustive search. This is in line with VPR's execution model to reduce the CAD tool runtime.
- The generic connection-boxes and switch-boxes are in the form of either single-driver or directional wiring architectures.

In the following two subsections, we give an overview of Fang's and Lam's models. In Table 2.1, we give a summary of the model parameters and their definitions which we will refer in each stage of the model development.

Overview of Lam's Area Equations

Lam et al. estimate the total number of K-LUTs (n_k) for the technology-mapped circuit using equation 2.1.

$$n_k = n_2 . \sqrt[p]{\frac{3}{K+1-\gamma}} \tag{2.1}$$

Given the set of logic architectural parameters, (K, N, and I), Lam substitute n_k in equation 2.3 with the expression in equation 2.1 to calculate the expected number of CLBs (n_c) based on the N-limited architecture.

$$i = \frac{(K+1-\gamma).N^p}{1+\frac{1}{f_{avg}}} \tag{2.2}$$

$$n_c = \frac{n_k}{N} \tag{2.3}$$

Overview of Fang's Channel Width Model

Equation 2.4 depicts Feuer's wirelength model [24]. This model expresses the point-to-point expected wirelength ($\bar{R}$) in terms of the Rent's exponent (p).

$$\bar{R} = \frac{2\sqrt{2}\,(3+3p)}{(1+2p)\,(2+2p)} n_c^{(p-0.5)} \tag{2.4}$$

Equation 2.5 depicts El Gamal's master slice interconnect model [25]. This model establishes the relationship between the absolute minimum channel width (W_{am}) and the point-to-point expected wirelength ($\bar{R}$) for the target FPGA with fully flexible routing architecture.

$$W_{am} = \rho\frac{\lambda.\bar{R}}{2} \tag{2.5}$$

Feuer's wirelength model expresses average wirelength in terms of n_c and Rent's exponent. Fang's channel width model, equation 2.6, expresses the minimum required channel width (W_N) in terms of routing architecture parameters and the absolute minimum channel width (W_{am}) from El Gamal's model where α_{in}, α_{out} and β are curve fitting constants.

$$W_N = W_{am} + \frac{1}{\beta}\left(\frac{W_{am}}{F_S}\right)\left(\frac{W_{am}}{F_{C_{in}}}\right)^{\alpha_{in}}\left(\frac{W_{am}}{F_{C_{out}}}\right)^{\alpha_{out}} + \frac{\lambda\,(L-1)}{4}\left(1+\frac{1}{F_{C_{in}}^{\alpha_{in}}}\right) \tag{2.6}$$

For the W_{am} term in equation 2.6, Fang uses equation 2.7 by replacing the $\bar{R}$ in equation 2.5 with the expression in equation 2.4.

$$W_{am} = \rho\lambda\frac{\sqrt{2}\,(3+3p)}{(1+2p)\,(2+2p)}n_c^{(p-0.5)} \tag{2.7}$$

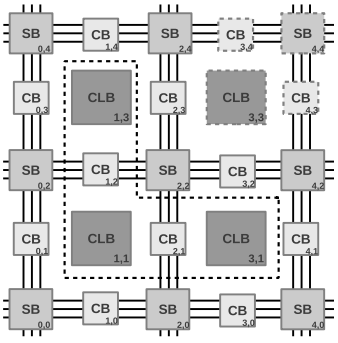
Estimating the routing resources

In the following subsections, we will derive the equations for modelling the routing and logic resource usage followed by leakage power dissipation. During our evaluations in section 2.5, we will refer back to these equations and discuss what specific behaviour or trends of experimental results we are intending to capture with the way we are constructing the model equations.

Minimizing the total wirelength is one of the primary objectives of the placement stage for minimizing area. Hence, instinctively, the logic blocks are placed close to each other resembling a near *square* shape. Based on this observation, we estimate the number of logic blocks, connection-boxes and switch-boxes using the following equations.

We use Figure 2.1 to illustrate our approach for estimating the routing resources. The dotted lines surrounding the configurable logic blocks depict the bounding box of the circuit. The unused resources are depicted with borders in dotted lines. Figure shows a circuit that requires three out of a total of four CLBs. The *square* is

Figure 2.1: The hypothetical homogeneous island-style FPGA.



denoted by the bounding box that consists of the CLBs [1,1], [1,3], [3,1] and [3,3], including the routing resources. The size of the square is defined by the number of CLBs that the edge of the *square* spans (l).

Equation 2.8 depicts l as a function of n_c.

$$l = \lceil \sqrt{n_c} \rceil \tag{2.8}$$

Equation 2.9 depicts the total number of CLBs (n_{tot}).

$$\begin{aligned} n_{tot} &= l^2 \\ &= \lceil \sqrt{n_c} \rceil^2 \end{aligned} \tag{2.9}$$

The number of switch-boxes (sb_{tot}) around the l by l *square* region is expressed by equation 2.10.

$$\begin{aligned} sb_{tot} &= (l+1)^2 \\ &= (\lceil \sqrt{n_c} \rceil + 1)^2 \end{aligned} \tag{2.10}$$

The number of connection-boxes (cb_{tot}) around the l by l *square* region is expressed by equation 2.11.

$$\begin{aligned} cb_{tot} &= 2l\,(l+1) \\ &= 2\lceil \sqrt{n_c} \rceil . (\lceil \sqrt{n_c} \rceil + 1) \end{aligned} \tag{2.11}$$

In order to calculate the number of used and unused routing resources (i.e., switch-boxes, and connection-boxes), our model assumes that logic blocks are allocated to the CLBs in a row-major order from left to right. Since the *square* is minimally sized, the number of CLBs on the bottom and left sides is always l. Equations 2.12 to 2.15 depicts the number of used CLBs with respect to the Bottom, Left, Top and Right sides.

$$B = l \tag{2.12}$$

$$L = l \tag{2.13}$$

$$T = B - (n_c \; mod \; B) \tag{2.14}$$

$$R = \lfloor \frac{n_c}{B} \rfloor \tag{2.15}$$

Equations 2.16 to 2.19 depicts the number of used/unused switch-boxes (sb_{used}, sb_{unused}) and connection-boxes (cb_{used}, cb_{unused}).

$$sb_{used} = (B+1)\,.\,(R+1) + T + 1 \tag{2.16}$$

$$sb_{unused} = sb_{tot} - sb_{used} \tag{2.17}$$

$$cb_{used} = R.\,(B+1) + B.\,(R+1) + 2T + 1 \tag{2.18}$$

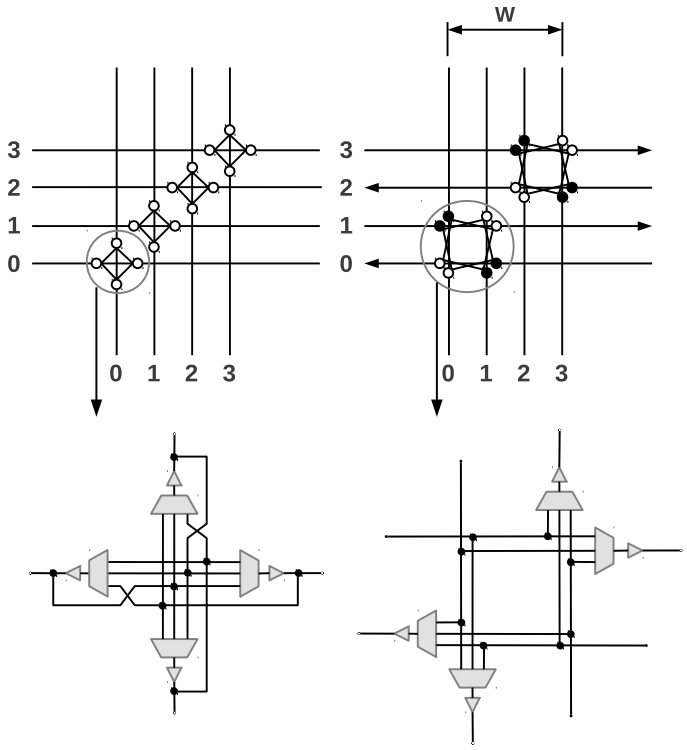
$$cb_{unused} = cb_{tot} - cb_{used} \tag{2.19}$$

Estimating the switch-box resources

Figure 2.2 depicts two switch-boxes with directional (left) and single-driver (right) wiring architectures. The figure assumes that a channel width, W, of 4 is used. The unfilled circles in the switch box denotes the presence of a buffered routing multiplexer. Both architectures have segment length, L=1, and switch-box flexibility, F_S=3. Since L is 1, all wires will terminate in the switch-box. F_S defines the number of outgoing tracks that can be connected to a terminated wire in the switch-box. Since F_S is 3, each routing switch/multiplexer will have 3 inputs. Equation 2.20 depicts the number of inputs of a routing multiplexer ($mux_{sb_{in}}$) considering only terminating wires in a switch-box.

$$mux_{sb_{in}} = F_S \tag{2.20}$$

Figure 2.2: The directional and single-driver switch box architectures.



In addition to terminating wires, our model takes into account long wires (i.e., $L > 1$) from a channel and output wires from a neighbouring CLB.

In a routing channel, longer wire segments are placed in a staggered fashion. The user-defined channel width, (W), must be a multiple of $2L$ for single-driver wiring, or a multiple of L for directional wiring. Therefore, the number of wires that terminate in the switch-box on each side of the switch-box is $\frac{W}{Dir.L}$, where Dir corresponds to 1 or 2 for directional or single-driver wirings respectively. By taking longer wires into account, we modify equation 2.20 and include the number of multiplexer inputs as shown in equation 2.21.

$$mux_{sb_{in}} = F_S + \left(\frac{W}{Dir}\right).\left(1 - \frac{1}{L}\right) \tag{2.21}$$

Each switch-box has four logic block neighbours. Hence, the total number of outputs from the four logic blocks is equal to the cluster size, N. Since each output can connect to F_{Cout} tracks, the total number of possible tracks are $F_{Cout} \times N$. Since there are a total number of $4.\frac{W}{L.Dir}$ multiplexers in a switch box, the total number of tracks are distributed equally among these multiplexers as inputs to each multiplexer. Therefore, we modify equation 2.21 to take number of multiplexer inputs into account as shown in 2.22.

$$mux_{sb_{in}} = F_S + \left(\frac{W}{Dir}\right).\left(1 - \frac{1}{L}\right) + \frac{F_{Cout}.N.Dir.L}{4W} \tag{2.22}$$

Eventually, equation 2.23 depicts the total number of multiplexers in a switchbox (mux_{sb}).

$$mux_{sb} = \frac{4W}{L.Dir} \tag{2.23}$$

Estimating the connection-box resources

The connection box provides the input connections to a CLB. Similar to the switch-box architecture, we assume that all routing switches are implemented with mul-

tiplexers. In a connection-box, each CLB input is able to select F_{Cin} tracks on the channel. Hence, equation 2.24 depicts the number of inputs for each routing multiplexer ($mux_{cb_{in}}$).

$$mux_{cb_{in}} = F_{Cin} \tag{2.24}$$

The total number of multiplexers in a connection box (mux_{cb}) is determined by the number of inputs of a CLB as depicted in equation 2.25.

$$mux_{cb} = I \tag{2.25}$$

Deriving the routing leakage power

The main sources of routing leakage power are the multiplexers and buffers. Our model assumes that all multiplexers are implemented in Pass-Transistor Logic (PTL). Static power of multiplexers is estimated by calculating the number of utilized minimum-width transistors.

We assume that each multiplexer is composed of 2-to-1 multiplexers. The leakage power for a 2-to-1 multiplexer is calculated by accumulating the leakage power of each pass transistor. If the multiplexer is in idling state, we assume that both transistors in a 2-to-1 multiplexer dissipates leakage power. An active 2-to-1 multiplexer consumes 50% of the total leakage power of an idling multiplexer, since only one NMOS transistor is conducting. Then, the total leakage power is calculated by accumulating the leakage power of 2-to-1 multiplexers.

We assume that all routing multiplexers' outputs are buffered. Buffers are implemented using CMOS technology. During idling state, both PMOS and NMOS transistors dissipate leakage power. During active state, the leakage power is reduced to half of the idle state.

Equation 2.26 depicts the overall leakage power dissipated by all switch-boxes. The ratio $\frac{W_N}{W}$ determines the expected number of multiplexers that are utilized based on the minimum channel width obtained from Fang's model.

$$
\begin{aligned}
P_{sb_{leak}} &= sb_{used}.mux_{sb}.\frac{W_N}{W}.\frac{P_{buffer_{leak}}}{2} \\
&+ sb_{used}.mux_{sb}.\frac{W - W_N}{W}.P_{mux_{leak}} \\
&+ sb_{unused}.mux_{sb}.P_{mux_{leak}} \qquad (2.26)
\end{aligned}
$$

where, $P_{mux_{leak}}$ is the leakage power dissipated by an idling routing multiplexer and $P_{buffer_{leak}}$ is the leakage power dissipated by a buffer.

Equation 2.27 depicts the overall leakage power dissipated by all connection-boxes. Similar to equation 2.26, the ratio $\frac{i}{I}$ determines the expected number of utilized multiplexers based on the expected number of used CLB inputs obtained from Lam's model.

$$
\begin{aligned}
P_{cb_{leak}} &= cb_{used}.mux_{cb}.\frac{i}{I}.\frac{P_{buffer_{leak}}}{2} \\
&+ cb_{used}.mux_{cb}.\frac{I - i}{I}.P_{mux_{leak}} \\
&+ cb_{unused}.mux_{cb}.P_{mux_{leak}} \qquad (2.27)
\end{aligned}
$$

Both equations 2.26 and 2.27 have following three components.

- Leakage power dissipated by all active multiplexer's driver buffers of used switch/connection boxes.
- Leakage power dissipated by all pairs of idling multiplexer and driver buffer of used switch/connection boxes.
- Leakage power dissipated by all pairs of idling multiplexer and driver buffer of unused switch/connection boxes.

We make use of the CMOS device model defined in VPR when calculating leakage power. However, here we note that, our model is independent of the device technology. Therefore, we can target different device architectures by regenerating the leakage power values.

We express the total routing leakage power ($P_{route_{leak}}$) using equations 2.26 and 2.27.

$$P_{route_{leak}} = P_{sb_{leak}} + P_{cb_{leak}} \tag{2.28}$$

2.3.3 FPGA Logic Static Power

Figure 2.3(a) denotes the look-up table structure used in our power model. We define the logic static power as the power dissipated by components within the CLB. The multiplexers, LUTs and buffers are the main contributors to the static power dissipation.

Each LUT can be connected to every other LUT. Therefore, we assume that there exists a full crossbar switch within the CLB. With the full crossbar switch, a multiplexer is required for each LUT input. Hence, the number of utilized multiplexers is determined by the number of used LUT input pins. Equations 2.29 and 2.30 depict the number of used and unused input routing multiplexers ($mux_{clb_{used}}$, $mux_{clb_{unused}}$).

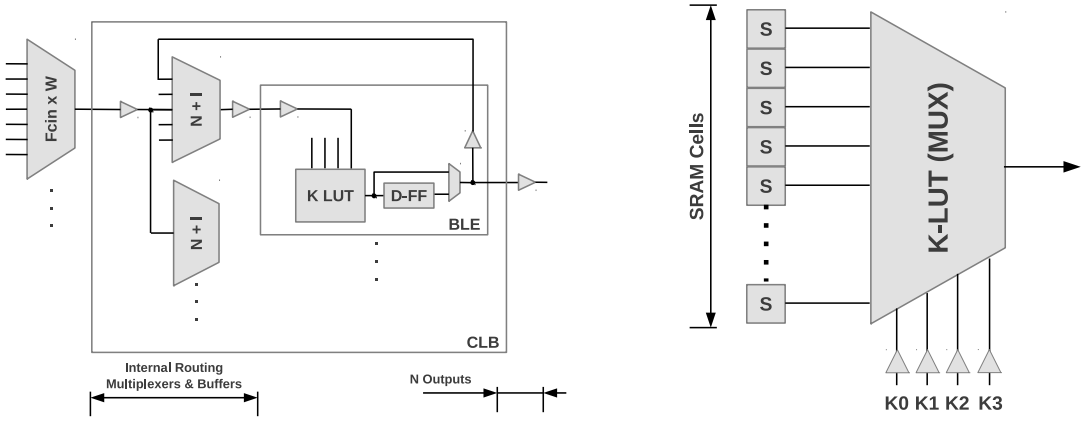
$$mux_{clb_{used}} = c.(k - \gamma) \tag{2.29}$$

$$mux_{clb_{unused}} = (N.K) - mux_{clb_{used}} \tag{2.30}$$

Similar to the connection and switch-boxes, the leakage power of a multiplexer is calculated by accumulating the leakage power in the form of 2-to-1 multiplexers. Equation 2.31 depicts the total leakage power dissipated by all the CLB routing multiplexers ($P_{mux(all)_{leak}}$) where $P_{mux_{leak}}$ denotes the leakage power of a multiplexer.

$$P_{mux(all)_{leak}} = P_{mux_{leak}}.mux_{clb_{used}} + 2.P_{mux_{leak}}.mux_{clb_{unused}} \tag{2.31}$$

Figure 2.3: The configurable logic block and look-up table architecture.



(a) Configurable Logic Block Architecture

(b) Look-Up Table Architecture

The active multiplexer will dissipate half the leakage power of a unused multiplexer. Therefore, we express the total leakage power for routing multiplexers using equation 2.31.

Figure 2.3(b) depicts the LUT architecture that is used in our model with K=4.

Given that a LUT has a structure similar to a multiplexer, the leakage power for a LUT ($P_{lut(mux)_{leak}}$) can be calculated by accumulating the leakage power of each 2-to-1 multiplexers that each LUT is composed of.

Furthermore, as illustrated in the figure, each LUT input (K0∼K3) is connected to input buffers. These buffers increase linearly with the number of K inputs of the LUT. Our model also takes into account the power dissipated by these buffers using K as a factor. Hence, equation 2.32 denotes the leakage power for a single LUT ($p_{lut_{leak}}$).

$$p_{lut_{leak}} = K.P_{buffer_{leak}} + P_{lut(mux)_{leak}} \tag{2.32}$$

Our power model takes into account the leakage power dissipated by both active and idling LUTs. Equation 2.32 depicts the leakage power dissipated by an active LUT. Assuming that 50% of the paths are conducting when the LUT is active,

equation 2.33 depicts the total leakage power dissipated by all the LUTs in a logic block.

$$P_{lut_{leak}} = c.p_{lut_{leak}} + 2.(N-c).p_{lut_{leak}} \quad (2.33)$$

Then, the total leakage power of an active CLB ($P_{clb(active)_{leak}}$) is calculated by taking the summation of the leakage power dissipated by all the LUTs and internal input routing multiplexers as depicted by equation 2.34.

$$P_{clb(active)_{leak}} = P_{lut_{leak}} + P_{mux(all)_{leak}} \quad (2.34)$$

The total leakage power of an idle CLB ($P_{clb(idle)_{leak}}$) is,

$$P_{clb(idle)_{leak}} = 2NK.P_{mux_{leak}} + 2N.p_{lut_{leak}} \quad (2.35)$$

Finally, the total logic leakage power, $P_{logic_{leak}}$ is calculated by taking the summation of the leakage power dissipated by all active and idle CLBs as depicted by equation 2.36.

$$P_{logic_{leak}} = (n_{tot} - n_c).P_{clb(idle)_{leak}} + n_c.P_{clb(active)_{leak}} \quad (2.36)$$

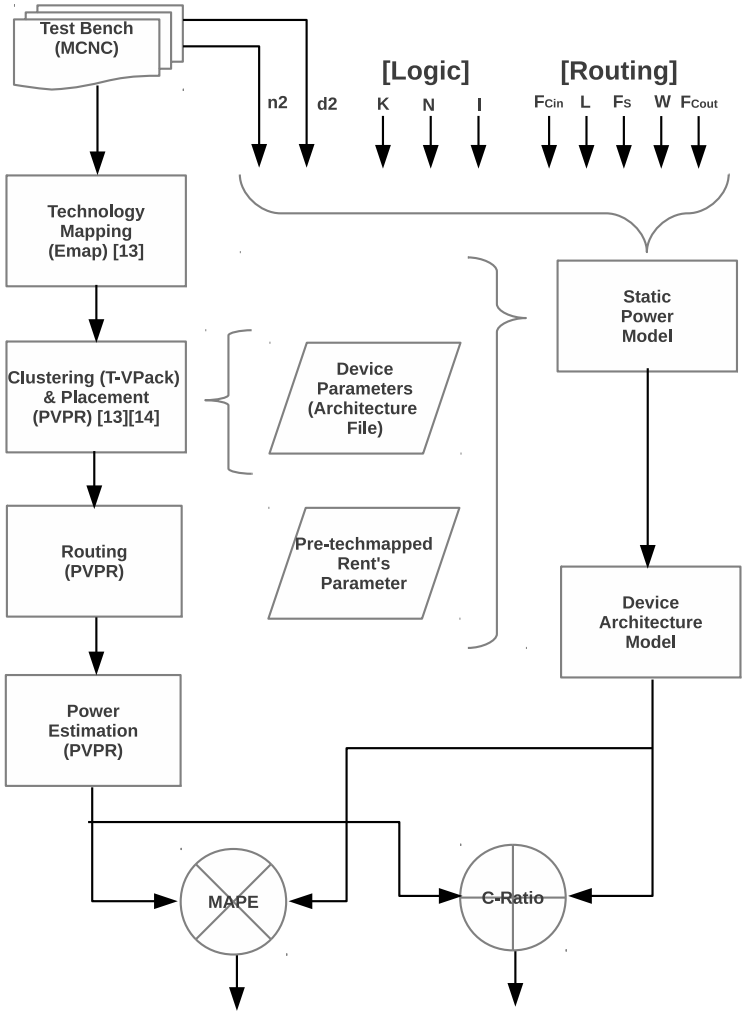
2.4 Evaluation Methodology

The quality of an analytical model is quantified by the degree of correlation between the empirical values generated by the model and the experimental values generated by CAD tools with respect to a set of benchmarks. We employ the Mean Absolute Percentage Error (MAPE) and the Correlation Ratio (c-ratio) metrics for this purpose.

Mean Absolute Percentage Error (MAPE) is defined by equation 4.8

$$MAPE = \frac{1}{C} \sum_{all\ C\ circuits} \frac{|Predicted - Measured|}{Measured} \quad (2.37)$$

Figure 2.4: Static power model validation methodology



where the *Predicted* and *Measured* refer to the model and experimental outputs respectively.

Correlation Ratio (c-ratio) is defined by equation 4.9

$$c-ratio \;=\; \frac{Number\ of\ non-violating\ pairs}{Total\ numberofpairs} \tag{2.38}$$

where a *pair* denotes data points containing experimental output and its corresponding model output.

A *pair* is *non-violating* when the model output resonates with the experimental output. For instance, an increase in the experimental value for a data point with respect to the previous data point should show an increase in the modeled value or vice-versa.

Our experimental testbed, Figure 2.4, employs EMap technology mapper [41], timing-driven clustering algorithm, T-VPack [47], followed by power-aware VPR (PVPR) [41] for the placement and routing stages. The test bench suite used during validation is based on the 20 largest MCNC benchmarks. These benchmarks are used mainly due to the reason we are targeting homogeneous FPGA architectures. Although these are relatively small circuits and realistic circuits are often much larger (i.e., close to 1 million 4-LUTs [33]), it has been shown that these larger circuits will have relatively the same Rent's parameter [53]. Since the amount of logic and routing resources are derived mainly based on the complexity of the circuit as represented by the Rent's parameter, we believe that our model will be able to cater to a larger benchmark as well. The reason for choosing the PVPR tool over the latest Verilog-to-Routing (VTR) tool [57] is because, there is currently no power estimation tool that is built on VTR. After generating the minimum routable channel width using the breadth-first routing algorithm, we perform relaxation on channel width, since a minimum channel width will put too much unrealistic stress on the router. The Rent's exponents for MCNC benchmarks are extracted before the technology mapping stage as reported in [41]. These parameters have been obtained using bi-partitioning technique on un-mapped circuits. Table 2.2 summarises the

individual Rent's parameter for each benchmark. As depicted in Figure 2.4, we estimate the static power of each benchmark for various architecture configurations by sweeping the logic (i.e. K, N, I) and routing (i.e. F_{Cin}, F_{Cout}, L, W) architecture parameters. The sweeping range for each parameter is listed in Table 2.1. Then, for each configuration, we generate the experimental values using the CAD flow. We then take the average of the static power for MCNC benchmarks. We evaluate the accuracy of our model with respect to the experimental results using MAPE and c-ratio metrics.

2.5 Model Validation

We start our analysis by first studying the correlation between the model and experimental values for each of the 20 MCNC benchmarks using Rent's parameter generated based on both pre-technology mapped (Figure 2.5(a)) and post-placed (Figure 2.5(b)) netlists. The logic architecture used during the experiment has LUT size, K=4, cluster size, N=10, inputs per cluster, I=22. The routing architecture has input connection flexibility, $F_{C_{in}}$=0.25, output connection flexibility, $F_{C_{out}}$=1.0, segment length, L=4, switch-box flexibility, F_s=3, and minimum circuit channel width, W. The MAPE values are represented in tuples, denoting the (average, minimum, maximum), with a standard deviation of σ. We want to show that CAD dependence of a model could be eliminated by using the Rent's parameter extracted from the pre-technology mapped netlist, and such an approach does not affect the quality of a model negatively. We set the logic and routing architecture parameters to their default values (listed in Table 2.1 in brackets) used commonly in FPGA CAD research. We show that the model that uses Rent's parameter generated based on pre-techmapped netlist preserves a high accuracy and fidelity with lower average MAPE values for both the routing (0.178) and logic leakage (0.146) power estimations. This observation also confirms the conclusion of [55], where they show that using Rent's exponent generated based on pre-techmapped netlist has negligible impact on the accuracy of their energy model.

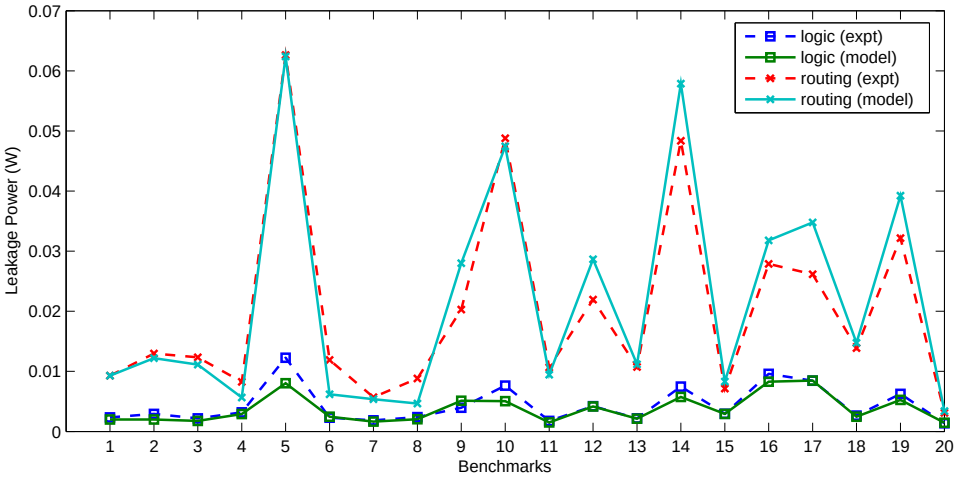
Benchmark	**Rent's Parameter**
alu4	0.77
apex2	0.61
apex4	0.78
bigkey	0.87
clma	0.51
des	0.77
diffeq	0.71
dsip	0.83
elliptic	0.84
ex1010	0.67
ex5p	0.80
frisc	0.77
misex3	0.72
pdc	0.67
s298	0.78
s38417	0.65
s38584.1	0.72
seq	0.75
spla	0.70
tseng	0.81

Table 2.2: MCNC Benchmarks' Rent's Parameters

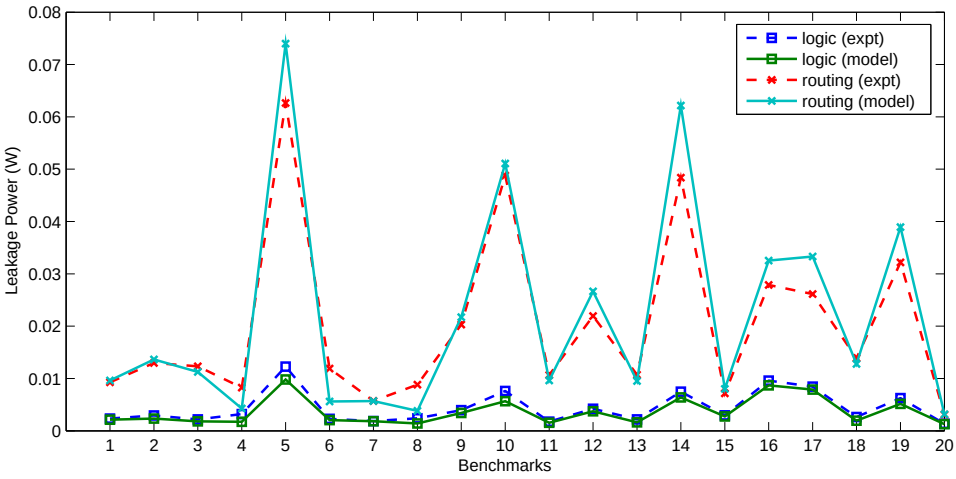
Table 2.3: Summary of pre-techmap and post-placement Rent's parameter.

Components	$M\hat{A}PE$			
	Average	Minimum	Maximum	Standard Deviation, σ
Logic	0.146	0.0	0.350	0.109
	0.175	0.053	0.469	0.115
Routing	0.178	0.0	0.479	0.152
	0.182	0.053	0.469	0.115

Figure 2.5: A study between pre-techmap and post-placement Rent's parameter.



(a) Pre-techmap individual benchmark comparison



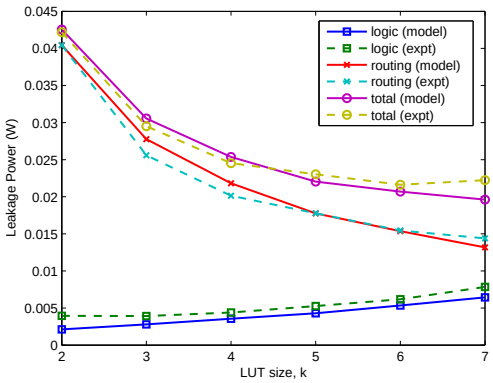
(b) Post-placement individual benchmark comparison

Even though our model is targeted for FPGA architects, who are more concerned with the changing power trend with varying architecture configurations rather than the actual values, based on Figure 2.5, we verify that the model is able to capture both the logic and routing leakage power accurately on a benchmark basis. We observe a percentage error that is higher than 20% between our model and experimental results for benchmarks 4, 6, 8, 12 and 19. Figure 2.5(a) depicts that our static power model is overestimating for benchmarks 5, 9, 12, 14, 16, 17, and 19. The static power dissipated by each benchmark is proportional to the amount of logic and routing resources. In equations 16 to 19, the number of used and unused switch boxes and connection boxes are derived based on n_c. Comparing the experimental results with the logic density model outputs of equation 3 reveals that the model is overestimating n_c for benchmarks 5, 9, 12, 14, 16, 17, and 19 as well. Hence, there is a strong correlation between n_c and our static power model. In this paper, equation 3 is used to derive the number of routing resources required. Hence, we believe that there is a need for a post-routing wirelength model to replace the logic density model for improving the accuracy of our static power model. Figure 2.5(a) depicts underestimation for benchmarks 4, 6, and 8, with 6 and 8, in particular, denoting the largest MAPE values ($> 40\%$) reported. Similar to the case with overestimation, we argue that estimating the amount of routing resources based on the number of used clusters, n_c,is an optimistic approach. Hence, we postulate that using a post-routing wirelength model will be able to ameliorate these estimation errors.

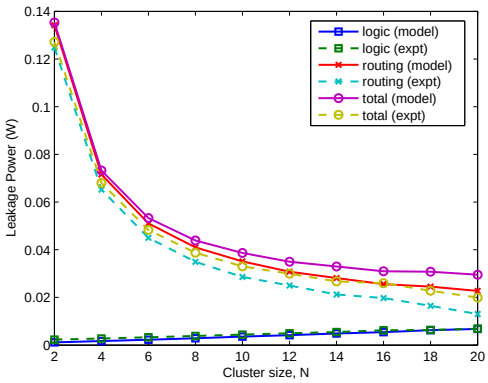
2.5.1 Effect of Logic Architecture Parameters

We evaluate the impact of change in the logic block architecture parameters individually on static power using Figures 2.6(a), 2.6(b) and 2.6(c) by fixing all other parameters to their typical values. Sweeping range and typical values for each parameter are listed in Table 2.1. Figures show logic, routing and total static power trends for the experimental and model values. Each data point in these figures represents the average leakage power over the 20 MCNC benchmarks.

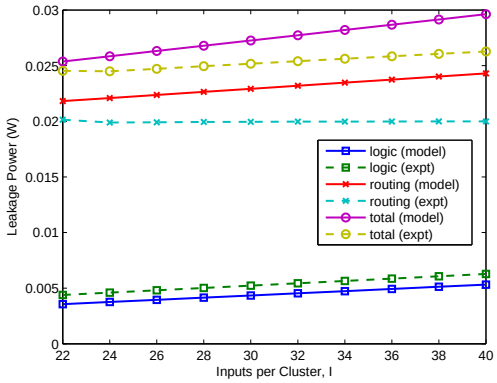
Figure 2.6: Sweep the logic architecture parameters (K, N, I).



(a) Sweep LUT Size, K ($N = 10$, $I = 22$)



(b) Sweep Cluster Size, N ($K = 4$, $I = N - Limited$)



(c) Sweep Inputs-per-cluster, I ($K = 4$, $N = 10$)

Varying LUT Size (K): While targeting a N-limited logic block architecture, the number of inputs, I, is defined by the function, $\lceil \frac{K}{2}.(N+1) \rceil$. The logic blocks are connected with a full crossbar network. As the logic block inputs are distributed across all the input routing multiplexers, the size of the internal input routing multiplexers will increase with I.

Each LUT input corresponds to a new multiplexer. The multiplexer output is connected to a buffer. Hence, as the LUT size increases, the total number of multiplexers and buffers grow which will lead to an increase in logic leakage power.

In the routing fabric side, increase in the LUT size leads to a decrease in the expected number of clusters, n_c. Reduction in number of clusters will incur a lighter inter-cluster routing overhead. This will imply that the number of connection and switch boxes will decrease. As illustrated in equations 2.10 and 2.11, the number of connection boxes (mux_{cb}) and switch boxes (mux_{sb}) are formulated as a function of n_c. These parameters are then used in equations 2.26 and 2.27 to calculate the leakage power dissipated by connection and switch boxes. Hence, as n_c decreases, the total amount of routing leakage power also decreases with increasing K. Since the routing leakage power constitutes a larger portion of the leakage power, the total power decreases for smaller LUT sizes. However, when the LUT size grows (i.e., $K = 7$), the increase in logic leakage power starts to dominate the decrease in routing leakage power. Therefore, we notice a slight increase in total leakage power when K is 7. As illustrated in Figure 2.6(a), our model captures this behaviour as we can notice the tapering decreasing trend of the total leakage power as K increases.

Overall, as K increases, our model captures the decreasing trend of routing leakage power and the increasing trend of logic leakage power with a high fidelity by achieving a low MAPE (0.048) and a high c-ratio (0.8).

Varying Cluster Size (N): The full crossbar network connection between the logic blocks implies that both the inputs and outputs from the LUTs are distributed across all input routing multiplexers. Since the number of LUT outputs is determined by N, increasing N will lead to increase in I, which will drive the multiplexer and buffer sizes and increase the logic leakage power. We observe this trend with

the experimental static power for logic resources in Figure 2.6(b), which is captured by our model accurately.

Number of connection and switch boxes decrease as N increases, for the same reasons we presented in varying LUT size section, which leads to the decrease in routing leakage power. This is reflected with the experimental routing results in Figure 2.6(b). Equations 2.26 and 2.27 depict the leakage power dissipated in a switch and connection box. These equations capture the trend we observe in experimental results by defining leakage power in terms of the total number of routing resources.

However, as illustrated in Figure 2.6(b), the routing leakage power starts to level off when N is larger ($N > 12$). This is because, there is a limit to the amount of functionality that can be packed into a single logic block, where beyond a certain cluster size no significant reduction in number of external connections is obtained. Our model captures this trend in equation 2.27 by expressing N as an inverse function of n_c after substituting I with $\lceil \frac{K}{2}.(N+1) \rceil$.

Our model shows the same behaviour we see in experimental total static power curve with respect to the change in N with a high fidelity by achieving a low $M\hat{A}PE$ (0.198), and a perfect $c - \hat{r}atio$.

Varying Inputs Per Block (I): We perform our evaluation based on N-limited architecture with $N > 22$, since before this value, the architecture becomes I-limited.

Despite the fact that the size of input routing multiplexers grows with I, the number of multiplexers is fixed as it is proportional with the number of LUT inputs. Since N and K are fixed, the total number of input routing multiplexers in each logic block is always bounded by the factor, $(N \times K)$. As the amount of logic that can be packed into a logic block saturates, n_c will not decrease any further. However, multiplexers continue to grow as I increases and the leakage power dissipated by these growing large multiplexers will cause the logic leakage power to increase eventually. We observe this behaviour with both the experimental and model based logic power curves.

As I increases, the decreasing trend for the number of clusters, n_c, saturates and the routing leakage power starts to increase, as shown with the experimental

routing power curve in Figure 2.6(c). This is because, the number of routing multiplexers in a connection box is determined by I. Hence, given a large I, these large multiplexers will dissipate leakage power causing routing leakage power to increase eventually. Our model accurately captures this behaviour with a strong correlation to the experimental curve with the second and third terms of equation 27. However, the model is increasing at a faster rate than the experimental results. This is because, the power estimations extracted from the experiments using PVPR do not take into account of routing multiplexers that connects the routing channels to the logic block input pins. Hence, the increase in experimental routing leakage power is due to the increase of routing switches in the connection boxes.

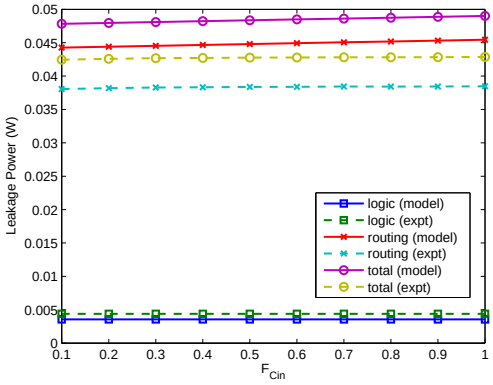
Based on the total power, our model shows high fidelity with a low $M\hat{A}PE$ (0.085) and a high $c - \hat{r}atio$ (1.0).

2.5.2 Effect of Routing Architecture Parameters

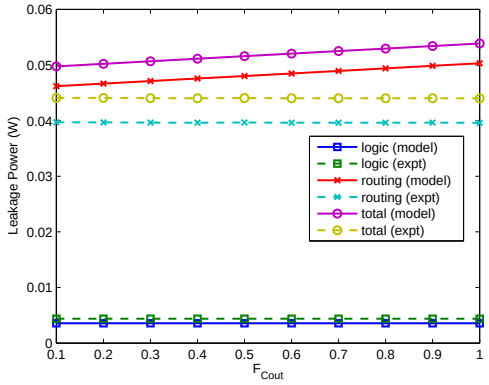
We evaluate the impact of change in the routing architecture parameters (F_{Cin}, F_{Cout}, L and W) individually on static power using Figure 2.7 by fixing all other parameters to their default values. Sweeping range and typical values for each parameter are listed in Table 2.1. Figures show logic, routing and total static power trends for the experimental and model values. Each data point in these figures represents the average leakage power over the 20 MCNC benchmarks. Here we note that changing routing architecture configuration does not impact the architecture of the logic blocks. Therefore, the logic leakage power will remain constant. We still show them in Figure 2.7 for our discussions on total power.

Varying Input Connection Box Flexibility (F_{Cin}): As F_{Cin} increases, the minimum routable channel width W_N will decrease. This is because, the PathFinder routing algorithm in VPR is optimized to generate the shortest path between each source and destination net. Moreover, having F_{Cin} large enough creates a relaxation, enabling the algorithm to explore alternative routing paths, rather than congesting to the same path all the time. This relaxation effect leads to a decrease in the minimum required channel width W_N. Hence, fewer multiplexers in the switch box

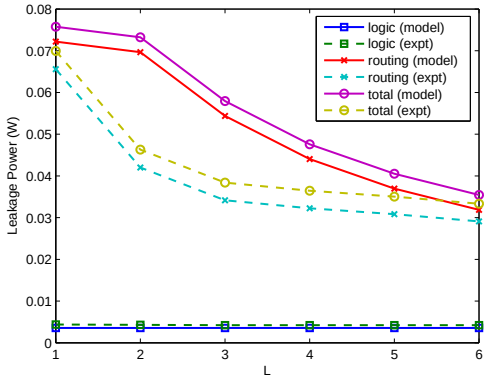
Figure 2.7: Sweep the routing architecture parameters ($F_S = 3$, F_{Cin}, F_{Cout}, L, W).



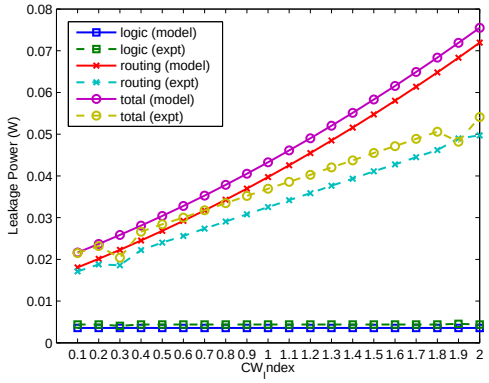
(a) Sweep input CB flexibility, F_{Cin} ($F_{Cout} = 1.0$, $L = 4$)



(b) Sweep output CB flexibility, F_{Cout} ($F_{Cin} = 0.25$, $L = 4$)



(c) Sweep segment length, L ($F_{Cin} = 0.25$, $F_{Cout} = 1.0$)



(d) Sweep channel width, W ($F_{Cin} = 0.25$, $F_{Cout} = 1.0$, $L = 4$)

Table 2.4: Summary of model's experimental performance.

Parameter	$c-\hat{r}atio$	$M\hat{A}PE$			
		Average	Minimum	Maximum	Standard Deviation, σ
K	0.8	0.048	0.005	0.090	0.216
I	1.0	0.072	0.029	0.114	0.028
N	1.0	0.175	0.040	0.472	0.125
$F_{C_{in}}$	1.0	0.133	0.125	0.143	0.006
$F_{C_{out}}$	0.889	0.176	0.127	0.255	0.032
L	1.0	0.266	0.048	0.559	0.216
W	0.947	0.207	0.005	0.492	0.130

will be utilized. Our model captures this reduction in channel width and switch box occupancy by making use of Fang's channel width model and incorporating W_N into equation 2.26. However, the size of routing multiplexers in the connection boxes is determined by F_{Cin}. Hence, as F_{Cin} increases, the leakage power dissipated by these routing multiplexers also increases. This will cause the routing leakage power to increase. Our model captures this experimental behaviour with a low $M\hat{A}PE$ (0.133) and a high $c-\hat{r}atio$ (1.0) for the total leakage power.

Varying Output Connection Box Flexibility (F_{Cout}): We define our switch box architecture to support both directional and single-driver FPGA interconnects. Therefore, the output pins from the logic block is connected into the switch box routing multiplexers directly [43]. Hence, as the F_{Cout} increases, the number of routing channels that is connected into the routing multiplexers increases. Due to the increase in multiplexer size, the number of transistors per multiplexer also increases. This inevitably leads to an increase in routing leakage power. In order to capture this relationship, we model F_{Cout} to be directly proportional to the number of switch box routing multiplexer inputs, as depicted in equation 2.22. With respect to change in F_{Cout}, our model shows high degree of accuracy with $M\hat{A}PE$ of 0.176, and $c-\hat{r}atio$ of 0.889.

Varying Segment Length (L): We assume that the length for all segments is

homogeneous. Within the routing channel, the segments are arranged in a staggered fashion if the length is greater than one. The number of multiplexers in a switch box is determined by the terminating routing segments in the switch box. Hence, given the staggered fashion that the segments are arranged, the longer the segment length, the fewer number of switch box multiplexers will be required. Equation 2.23 captures this relationship by expressing L to be inversely proportional with mux_{sb}. However, the routing multiplexers within the switch boxes will need to allow the long routing segments to change direction; since the by-passing long segments also need to connect to the routing multiplexers, the number of multiplexer inputs, $mux_{sb_{in}}$, will increase. Equation 2.22 captures this relationship in the second term. Furthermore, given a fewer number of multiplexers, the logic block outputs will constitute a larger proportion of the multiplexer inputs. Equation 2.23 captures this relationship in the third term of the expression. With the decreasing number of smaller routing multiplexers, we expect to see a reduction in the routing leakage power. However, due to the increase in inputs, the decrease in the model results is slower than the experimental results as L increases from 1 to 2. We do not observe the same experimental trend as tri-state buffers are used in place of multiplexers in PVPR. In general, Figure 2.7(c) illustrates that our hypothesis correlates with the actual experimental results. Overall, due to the decrease in routing leakage power, the total leakage power also decreases. Our model accurately captures the trend with a low $M\hat{A}PE$ (0.266) and a high $c - \hat{r}atio$ (1.0).

Varying Channel Width (W): W defines the total number of routing channels that enter or leave each side of the switch box. A routing multiplexer is required for each terminating channel in the switch box. As W increases, the number of terminating channels also increases. This increase in routing multiplexers will lead to an increase in routing leakage power. Equation 2.23 captures this trend by expressing the number of routing multiplexers, mux_{sb} to be directly proportional with W. On the other hand, the size of each routing multiplexer is dependent on the number of channels that terminate, change direction, or output from a logic block. As W increases, the number of passing channels that can potentially change direction in-

creases. Hence, the number of inputs for each multiplexer also increases. Equation 2.22 captures this behaviour by having W in the second term to be directly proportional to $mux_{sb_{in}}$. This increase in multiplexer size will also lead to an increase in routing leakage power. As illustrated in Figure 2.7(d), our model accurately captures the experimental behaviour with a low $M\hat{A}PE$ (0.207), and a high $c - \hat{r}atio$ (0.947).

2.6 Summary

In this chapter, we presented a CAD-independent static energy model by taking logic and routing architecture parameters into account. Our experiment intensive investigations involved analysis of the model behavior with respect to change in individual logic and routing architecture parameters using correlation ($c - \hat{r}atio$) and mean error ($M\hat{A}PE$) metrics. When we take the average of $c - \hat{r}atio$ and $M\hat{A}PE$ values reported for each configuration, our model achieves a correlation factor of 95% and deviates from the actual empirical values with a mean percentage error of approximately 17%. We conclude that our model, with its ability to rank hypothetical FPGA architectures accurately based on static energy consumption, is a powerful tool for application engineers and FPGA architects on evaluating the impact of their design choices on static power without having to go through the CAD intensive investigations.

CHAPTER 3

Homogeneous FPGA Post-Routing Wirelength Model

3.1 Introduction

Modeling wirelength is a critical step since channel width can be expressed as a function of total net length in a design, which is an indicator of routability for an FPGA [38]. Performance indicators, such as critical path delay and power consumption, are functions of net capacitance, which in turn is a function of net length. In this study, we introduce a post-routing wirelength model that takes into account both logic and routing architectural parameters. To the best of our knowledge, there is no prior work on analytically modeling the post-routing wirelength of a FPGA. Our wirelength model is concerned with generating a close approximation of post-routing multi-pin wirelength based on the homogeneous FPGA architecture. The model differentiates from previous modelling studies by considering the excess wiring overhead related to the routing architecture parameters (i.e., flexibility of switch boxes F_S, and connection boxes $F_{C_{in}}$ and $F_{C_{out}}$, segment lengths L, and logic block equivalence Eqv). The model is CAD independent as opposed to the existing wirelength models, since we extract Rent's coefficient of a circuit before the technology mapping stage of the FPGA CAD flow. We attempt to address the following questions:

- Does an optimized logic architecture configuration relate to an optimized post-routed FPGA architecture without taking routing architecture parameters into account?
- If not, what is the impact of incorporating routing architecture parameters? Can it supersede previous post-placement wirelength models or in what aspect can they be used for?

Our model development investigation involves revising Fang's minimum channel width model [23] using El Gamal's average connection length model [25] in order to incorporate routing architecture parameters. We revise this new expression for minimum channel width using the wirelength model of Davis [31][32] incorporate routing architecture parameters, and scale our model from expressing point-to-point to multi-point wirelength. We sweep the architecture search space exhaustively and compare the experimental wirelength values with the model based wirelength values over the MCNC and Verilog-to-Routing (VTR) benchmarks. We evaluate the quality of our model using mean absolute percentage error (MAPE) and correlation metrics. We show that our model accurately captures the post-routing wirelength with a high degree of fidelity to changes in routing objectives. In section 3.2, we give an overview of the related work, and then present the details of our model development approach in section 3.3. We describe the methodology and testbed used for evaluating the quality of our model in section 3.4. Detailed analysis of the results are presented in section 3.5, followed by conclusions and future work in section 3.7.

3.2 Related Work

Recently, a number of research thrusts have proliferated in the area of FPGA analytical modeling. Based on a limited set of architectural parameters (e.g., cluster size, LUT size, number of inputs, connection and switchbox flexibilities, etc), each model is able to provide useful architecture insights with respect to their target metric.

Fang et al. model the FPGA routability by estimating the channel width requirement based on logic, routing architectural parameters [23], and average wirelength. This model differs from Brown's work [60] in a way that, given the architectural parameters, the output of their model is the channel width targeting routability, while Brown's model gives a probabilistic estimation of routability. Their model is mainly derived based on empirical data through CAD experimentation with the 20

MCNC benchmarks and 8 benchmarks from OpenCores.

In [20], Das et al. extended on Brown's work [60] to take into account of a combined global/detailed router. Hence, all possible paths that emanate from the source to the destination are taken into account to predict the average routability of the circuit. This modelling approach is in-line with current state-of-the-art routing tools, such as the Versatile Place and Route (VPR) tool, whereby routing decisions with respect to both global and detailed routing are considered concurrently.

Smith et al. model the wirelength based on the homogeneous and heterogeneous FPGA architecture [3]. Backed by Rent's rule, their homogeneous architecture based wirelength model is an extension of Davis' work [24], which takes into account of multi-sink nets. Thereafter, building upon their model for homogeneous architecture, they explore the effects of heterogeneity with regularly spaced embedded blocks and the impact on post-placement wirelength. However, their model does not take into account of the routing architecture.

To explore the path delay for FPGAs, Hung et al. derive a closed form delay equation based on the equivalent RC-based circuit model of the logic and routing architecture [28]. Their hypothetical FPGA is based on a homogeneous FPGA. The authors also demonstrate the model's usability by integrating it with the post-technology mapping and post-clustering depth model [18] to estimate the critical path delay. The model proves to be an accurate predictor by considering the routing architectural parameters. However, due to the close binding of the model with the circuit, developers will need to adhere to strict circuit-level design rules in order to correlate with the model.

In [18], Das et al. explore the post-technology mapping, post-clustering stages' logic density of a LUT and cluster respectively. They also explore the circuit depth with respect to the same stages. The authors' advocate the derivation of the model based on fundamental relationships between architectural parameters and performance metrics, and refrain from experimentation based curve-fitting approaches. Their model uses Rent's exponent extracted from the post-placement netlist. In order to ensure a circuit's implementation does not influence their architecture based

model, the authors employ the pre-techmapped netlist and its depth during the derivation.

In our recent research work [55], Rajavel et al. integrate the wirelength model [18] and critical path delay model [3] to derive an energy model for FPGAs. By utilizing the heterogeneous wirelength model based on Davis' and Lam's work, the model is able to approximate energy of a heterogeneous architecture. This study evaluates the impact of variations in logic architecture parameters in terms of LUT size, cluster size and inputs per CLB on the energy performance using VPR.

3.3 Wirelength Modeling for Homogeneous FPGAs

3.3.1 Model Development Overview

Our wirelength model is concerned with generating a close approximation of post-routing multi-pin wirelength based on the homogeneous FPGA architecture. The multi-pin wirelength refers to the sum of all possible paths that span the Minimum Steiner Tree, with regards to all pin destinations.

The FPGA post-placement wirelength model introduced by Smith et al. [3] is based on the wirelength models derived for the ASIC domain by Feuer [24] and Davis [31][32]. Smith uses the logic architecture parameters in their model (i.e., assumes an unlimited number of channel tracks with fully-flexible routing architecture). Therefore, models of Feuer and Davis can be applied towards FPGA analytical modeling due to the close architectural resemblance of logic blocks interconnected among each other by routing wires.

Deriving the post-routing wirelength model requires the demarcation between the ASIC and FPGA routing interconnect architecture. The first difference is the finite number of channel tracks in a FPGA routing interconnect architecture as defined by the term, W, in our model. Hence, the first step in our model construction burgeons by establishing the relationship between the circuit demand channel width W_N and W, with respect to the post-routing wirelength. *With only 1 routing parameter, W, taken into consideration, this implies a FPGA device with fully-flexible*

routing architecture where circuit routability is only constrained by the number of channel tracks in a device. We differentiate the circuit demand channel width of this device with a separate term, $W_{N_{min}}$.

The second difference is the limited routing flexibility in a FPGA routing interconnect architecture. Hence, the second step is to introduce additional routing parameters into the equation to model the routing flexibilities in a typical FPGA. Intuitively, this will imply that the effective W will become smaller. Therefore, a circuit will require a larger channel width in order to get routed (i.e., $W_{N_{min}}$ will increase). In order to analyse the impact of an increasing $W_{N_{min}}$ on average wirelength, the model development progress by defining the operating region for scaling $W_{N_{min}}$ by conducting investigations based on two configurations. In the first configuration, we define the boundary when a FPGA device behaves like an ASIC with no limitations on W (i.e., W is determined based on $W_{N_{min}}$). In the second configuration, a typical FPGA is modeled by fixing, W, while sweeping $W_{N_{min}}$. Intuitively, a device with limited routing flexibilities will have a higher average wirelength than one that has fully flexible routing flexibilities. Therefore, identifying the region where the wirelength for the second configuration is higher than the first configuration will imply a valid operating region of the model.

Then, the third step is to derive an expression which can describe the trend for a varying $W_{N_{min}}$. We postulate that this target expression will be very similar to Fang's channel width model that captures the essence of routing architecture with circuit demand channel width. Therefore, instead of duplicating the work to remodel the equation again, we use the Fang's channel width model as a starting point for further empirical characterizations through independent training and validation CAD experimentations.

3.3.2 Model Training

The complete model development of the post-routing wirelength model needs to employ both analytical derivations and guided empirical approaches. Section 3.3.3 provides the justification details on the two approaches that are used.

Table 3.1: Training Benchmark Circuits

Circuit Name	n_2	d_2	Inputs	Outputs	Rent's Parameter
alu4	2732	14	14	8	0.77
apex4	2196	12	9	19	0.78
bigkey	2979	10	263	197	0.87
clma	14253	40	383	82	0.51
des	2901	14	252	243	0.77
dsip	2531	10	229	197	0.83
misex3	2557	13	14	14	0.72
pdc	8408	19	16	40	0.67
s38584.1	12491	25	39	304	0.72
seq	2939	14	41	35	0.75
diffeq1	12062	114	162	96	0.736
diffeq2	11708	108	66	96	0.737
stereovision1	102235	41	133	145	0.780

Table 3.2: Validation Benchmark Circuits

Circuit Name	n_2	d_2	Inputs	Outputs	Rent's Parameter
apex2	3165	17	39	3	0.61
diffeq	2556	39	64	39	0.71
elliptic	5474	52	131	114	0.84
ex1010	8020	17	10	10	0.67
ex5p	1779	15	8	63	0.80
frisc	6023	67	20	116	0.77
s298	4272	32	4	6	0.77
s38417	13656	25	29	105	0.65
spla	7438	19	16	46	0.70
tseng	1861	43	52	122	0.81
sha	8841	115	38	36	0.637
blob_merge	33404	107	36	100	0.727
stereovision0	36535	41	169	197	0.713

To facilitate the guided empirical flow, we split the set of benchmark circuits into independent training and validation sets. Then, data is collected by launching CAD experiments based on different architectures using the training set. The model is trained by fine-tuning equation constants to fit into the data trends. Similar to Fang et al., the least square method is used during the curve fitting [23].

In order to ensure that the model will not suffer from overfitting, cross validation [6] is performed. Hence, a second set of data based on the same CAD experimentations is collected using the validation set to evaluate the model's performance. The model is deemed to suffer from overfitting if the model's accuracy with respect to the validation data is significantly worse as compared to the training set [66].

The benchmark circuits that are used in the training and validation sets comprise of the 20 largest MCNC benchmarks and 6 of the default benchmarks from the Verilog-to-Routing (VTR) tool package. The entire benchmark suite is randomly partitioned under the constraint that the statistical circuit characteristics of each independent set has to be similar to the entire combined benchmark suite. Tables 3.1 and 3.2 depict the characteristics of each circuit of each set.

3.3.3 Model Construction

Post-routing wirelength model for ideal routing fabric

Step 1: Fang's minimum channel width model (W_N) as depicted in equation 3.1, is a function in terms of the average two-point wirelength, $\bar{R}$, based on routing architectural parameters, (i.e., F_S, F_{Cin}, F_{Cout}, L, Eqv) [23]. In order to facilitate our model development, we perform two simplifications on this equation.

Firstly, we assume that the circuit is routed on a FPGA with fully flexible routing architecture. Table 1 denotes the routing architectural parameters with respect to a fully flexible routing fabric given a target hypothetical FPGA with a channel width of W. We substitute these values into the equation to simplify the model.

Secondly, we set the values of α_{in} and α_{out} to 0.5. α_{in} and α_{out} denote the impacts for the connection box flexibilities, F_{Cin}, and F_{Cout} respectively. In Fang's

channel width model, more weight is allocated to F_{Cin} than F_{Cout} by assigning a larger α_{in} (0.5) than α_{out} (0.25). VPR tool is configured to favor a short wirelength connection. In other words, the most desirable case is, relative to the source block, when the destination block is the adjacent block. Giving input connection a higher priority encourages a connection to be established as soon as possible. However, since we are assuming a fully flexible routing architecture to begin with, we argue that the average wirelength will be relatively small, and the impact on wirelength for the input or output connection boxes should be almost equivalent. This is because, with an unlimited number of channel tracks and a fully-flexible routing interconnect architecture, the original constraint with domain partitioning of the channel tracks becomes negligible. Hence, to minimize the wirelength, it is as important to obtain a connection either at the source or destination pins.

Equation 3.2 depicts the simplified Fang's channel width model, where $W_{N_{min}}$ is the minimum channel width to route a circuit assuming a fully flexible architecture with respect to a specific user-defined channel width, W.

$$\begin{aligned} W_N &= W_{am} + \frac{1}{\beta}\left(\frac{W_{am}}{F_S}\right)\left(\frac{W_{am}}{F_{C_{in}}}\right)^{\alpha_{in}}\left(\frac{W_{am}}{F_{C_{out}}}\right)^{\alpha_{out}} \\ &\quad + \frac{\lambda\,(L-1)}{4}\left(1+\frac{1}{F_{C_{in}}^{\alpha_{in}}}\right) \end{aligned} \tag{3.1}$$

$$W_{N_{min}} = W_{am} + \frac{1}{3\beta}\left(\frac{W_{am}}{W}\right)^2 \tag{3.2}$$

Step 2: El Gamal's master slice interconnect model defines a relationship between the average number of wires used in a channel, W_{avg}, and the average two-point wirelength, $\bar{R}$. In [23], by introducing a *peak factor*, ρ, this relationship is

Table 3.3: Routing parameters based on a fully flexible FPGA architecture [23].

F_s	F_{cin}	F_{cout}	L	Eqv
3W	W	W	1	1

re-expressed in terms of W_{am}, the absolute minimum channel width required to route a circuit with a fully flexible routing architecture as shown in equation 3.3. We obtain equation 3.4 by substituting equation 3.3 into equation 3.2.

$$W_{am} = \rho\frac{\lambda.\bar{R}}{2} \tag{3.3}$$

$$W_{N_{min}} = \rho\frac{\lambda.\bar{R}}{2} + \frac{1}{3\beta}\left(\frac{\rho\frac{\lambda.\bar{R}}{2}}{W}\right)^2 \tag{3.4}$$

Step 3: After solving equation 3.4 for $\bar{R}$, we obtain an expression as shown in equation 3.5.

$$\bar{R} = \frac{3\beta W^2}{\rho\lambda}\left(-1 + \sqrt{1 + \frac{4W_{N_{min}}}{3\beta W^2}}\right) \tag{3.5}$$

Step 4: Davis defines a relationship between point-to-point ($\bar{R}$) and multi-point ($\bar{R_S}$) wirelength using equation 3.6, where f_{avg} is the expected fanout of a net. f_{avg} calculation is detailed in [51] using equations 3.7-3.9.

$$\bar{R_S} = \bar{R}\frac{4f_{avg}}{3 + f_{avg}} \tag{3.6}$$

$$f_{avg} = \frac{1 - (f_{max} + 1)^{(p-1)}}{1 - (f_{max} + 1)^{(p-2)} - \phi\,(p, f_{max})} - 1 \tag{3.7}$$

$$\phi\,(p, f_{max}) = \sum_{n=1}^{f_{max}} \frac{n^p}{n^2\,(n+1)} \tag{3.8}$$

$$f_{max} = \left[(i+N)\,\frac{n_k}{N}\,(1-p)\right]^{\left(\frac{1}{3-p}\right)} \tag{3.9}$$

We substitute $\bar{R}$ in equation 3.6 with equation 3.5, and express multi-point wirelength in terms of routing architecture parameters and $W_{N_{min}}$ with equation 3.10.

$$\bar{R_S} = \frac{3\beta W^2}{\rho\lambda}\left(-1+\sqrt{1+\frac{4W_{N_{min}}}{3\beta W^2}}\right)\frac{4f_{avg}}{3+f_{avg}} \tag{3.10}$$

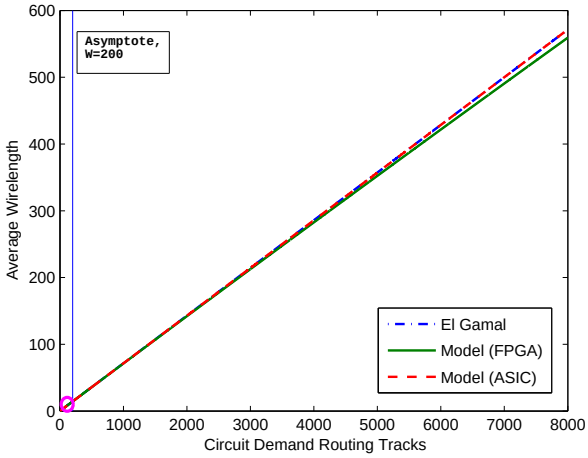
Analyse ideal wirelength model's asymptotic behaviour

In order to extend the model to accommodate a less flexible routing architecture, we perform two studies to define the operating regions based on the wirelength for a two-pin net. To compare our model with El Gamal's model, we make use of equation 3.5 to perform our studies. The following denotes the approach to our studies.

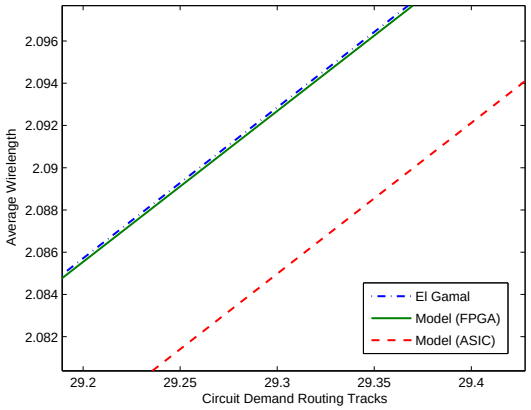
- *Study 1*: The hypothetical architecture is able to accommodate any channel width demands from the circuit. We denote this architecture as *Model (ASIC)*. This case depicts a typical ASIC scenario in which the amount of routing tracks is not bounded by the device channel width. In this case, since the number of channel tracks is not a limiting factor, W is defined to be the same as the circuit required channel width, $W_{N_{min}}$. Equation 3.11 is the result of equation 3.5 after substituting W with $W_{N_{min}}$.

$$\bar{R} = \frac{3\beta W_{N_{min}}^2}{\rho\lambda}\left(-1+\sqrt{1+\frac{4}{3\beta W_{N_{min}}}}\right) \tag{3.11}$$

- *Study 2*: The hypothetical architecture is assigned a fixed channel width. We denote this architecture as *Model (FPGA)*. In this case, the variable, W in equation 3.5 is substituted with a typical device channel width of 200. This case depicts a typical FPGA scenario in which a limited number of channel tracks are available for routing. Therefore, any circuit which requires a channel width that exceeds 200 cannot be implemented on the device. Equation 3.12 is the result of equation 3.5 after substituting W with 200.

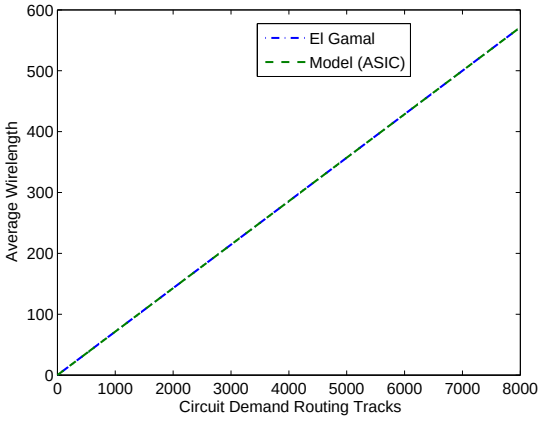


(e) Comparing model's variations with El-Gamal's model

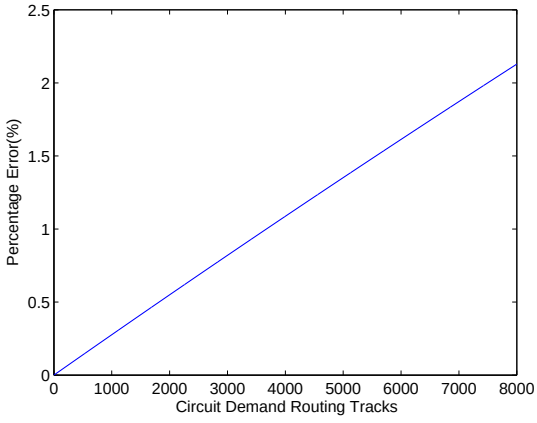


(f) Pre-asymptotic model's behaviour

Figure 3.1: Model analysis with varying circuit demand channel width



(a) Comparing model's variations with El-Gamal's model



(b) Error deviations

Figure 3.2: Model analysis with varying circuit demand channel width

$$\bar{R} = \frac{3\beta\,(200^2)}{\rho\lambda}\left(-1+\sqrt{1+\frac{4W_{N_{min}}}{3\beta\,(200^2)}}\right) \tag{3.12}$$

Figure 3.1(e) denotes the average two-point wirelength as $W_{N_{min}}$ is varied from 1 to 8000. This large maximum sweeping limit of 8000 is defined in order to illustrate the model's behaviour as compared to El Gamal's model. *Model (ASIC)* denotes equation 3.11 in *Study 1*. *Model (FPGA)* and the blue vertical asymptote denotes equation 3.12 and the device channel width in *Study 2* respectively.

It can be observed that *Model (ASIC)* coincides with El Gamal's model over the entire study range. This high correlation and minimal error deviation shows that equation 3.5 is able to preserve the channel width to wirelength relationship of an ASIC which resembles a fully flexible FPGA for the study range (1-8000). However, as circuit demand channel width increases, *Model (FPGA)* shows a dipping trend and becomes smaller than *Model (ASIC)*. In theory, *Model (ASIC)* should have a smaller average wirelength than *Model (FPGA)* due to the unlimited number of channel track provisioning in *Model (ASIC)*. Examining the figure in close details also reveals that this under-estimation of *Model (FPGA)* can be observed originating from the vertical asymptote. Therefore, based on this argument, the region of $W > 200$ is not a valid operation region for *Model (FPGA)*.

Figure 3.1(f) denotes an enlarged portion of the range when $W < 200$ in figure 3.1(e) as denoted by the circle drawn with the color, magenta. The higher estimated average wirelength of *Model (FPGA)* than *Model (ASIC)* clearly indicates the validity of this operating range when the circuit demand channel width is smaller than the device available channel width (i.e., $W_{N_{min}} < W$). This argument also coincides with the actual CAD tool behaviour as it is impossible to route a circuit that requires a channel width above the architecture limiting number of channel tracks. Therefore, during wirelength estimation, the user-defined device channel width (i.e., specific to a FPGA architecture) needs to be larger than the minimum circuit required channel width in order for the estimated wirelength value to be regarded valid.

Post-routing wirelength model for typical routing fabric

In the following steps, equation 3.10 is extended to take into account of a resource constraint routing interconnect architecture. This is typical in the case of FPGAs as routing resources often occupy the majority of resources (e.g., area, power, delay). Therefore, it is too costly (i.e., near impossible) to have a fully-flexible routing interconnect architecture.

Step 5: Equation 3.10 depicts the expression that describes the average Steiner tree wirelength for a given circuit using a fully flexible routing architecture. From section 3.3.3, we have proved that the wirelength model is valid as long as the channel width needed by the circuit, W_N is less than the user-defined device channel width, W. Since, by definition, W_N will need to fall within the range, $W_{N_{min}} < W_N < W$, we will proceed to approximate the term $W_{N_{min}}$ with Fang's channel width model, as depicted by equation 3.1. Theoretically, it might have been possible to derive a new channel width model. However, we postulate that Fang's model will capture the majority of the relationship between the circuit required channel width and routing architecture parameters. Hence, it will make sense not to duplicate the work. Equation 3.13 depicts the post-routing wirelength equation after substitution and equation 3.14 depicts the channel width equation before training.

$$\bar{R_S} = \frac{3\beta W^2}{\rho\lambda}\left(-1+\sqrt{1+\frac{4W_N}{3\beta W^2}}\right)\frac{4f_{avg}}{3+f_{avg}} \tag{3.13}$$

where,

$$\begin{aligned} W_N &= W_{am} + \frac{1}{\beta}\left(\frac{W_{am}}{F_S}\right)\left(\frac{W_{am}}{F_{C_{in}}}\right)^{\alpha_{in}}\left(\frac{W_{am}}{F_{C_{out}}}\right)^{\alpha_{out}} \\ &+ \frac{\lambda\,(L-1)}{4}\left(1+\frac{1}{F_{C_{in}}^{\alpha_{in}}}\right) \end{aligned} \tag{3.14}$$

Fang et al. employs curve fitting methods to derive the constants (β, α_{in}, α_{out}) for equation 3.14. Directly applying this channel width model to the post-routing wirelength model will affect the model's fidelity due to the following factors below:

- There is a difference between the CAD tools which are used in both works. Our work emphasized on using the Verilog-to-Routing (VTR) flow. This differentiates from Fang's work which makes use of FlowMap and FlowPack for technology mapping stage and VPR for clustering, placement and routing stages.

- We postulate that using routing algorithm of different objectives will impact the actual wirelength and its trend. Fang's work is primarily focused on routability driven algorithms. Since we plan to study the impact of different routing objectives on our model, it will make sense to re-train our model to evaluate the difference.

Step 6: We derived the constants α_{in} and α_{out} for equation 3.14 by conducting independent training and validation CAD experimentations with the routability-driven router. Our detailed methodology is described in section 3.3.2. As illustrated in figure 3.9, F_{Cout}, and wirelength $\bar{R_S}$ has a direct proportional relationship using both the routability-driven and timing-driven routing algorithm. Based on this observation, we shifted F_{Cout} into the numerator. Thereafter, we derive the parameters $\alpha_{in} = 0.3$, and $\alpha_{out} = 0.06$ by performing curve-fitting using least squares method. Section 3.6 presents a detailed case study that analyses the architecture impact of F_{Cout}. Equation 3.15 depicts the final form of the post-routing wirelength model and equation 3.16 depicts the channel width equation after training.

$$\bar{R_S} = \frac{3\beta W^2}{\rho\lambda}\left(-1+\sqrt{1+\frac{4W_N}{3\beta W^2}}\right)\frac{4f_{avg}}{3+f_{avg}} \tag{3.15}$$

where,

$$\begin{aligned} W_N &= W_{am} + \frac{1}{\beta}\left(\frac{W_{am}}{F_S}\right)\left(\frac{W_{am}}{F_{C_{in}}}\right)^{0.3}(W_{am}.F_{C_{out}})^{0.06} \\ &+ \frac{\lambda\,(L-1)}{4}\left(1+\frac{1}{F_{C_{in}}^{0.3}}\right) \end{aligned} \tag{3.16}$$

Equation 3.16 is expanded by substituting W_{am} with equation 3.3. Equation 3.17 denotes the equation after expansion.

$$\begin{aligned} W_N &= \rho\frac{\lambda\bar{R}}{2} + \frac{1}{\beta}\left(\frac{\rho\frac{\lambda\bar{R}}{2}}{F_S}\right)\left(\frac{\rho\frac{\lambda\bar{R}}{2}}{F_{C_{in}}}\right)^{0.3}\left(\rho\frac{\lambda\bar{R}}{2}.F_{C_{out}}\right)^{0.06} \\ &+ \frac{\lambda\left(L-1\right)}{4}\left(1+\frac{1}{F_{C_{in}}^{0.3}}\right) \end{aligned} \tag{3.17}$$

To deploy the post-routing wirelength model, $\bar{R}$ needs to be substituted with a suitable two-point wirelength model (i.e., Davis[31][32] or Feuer[24]).

Step 7: Davis expresses $\bar{R}$ in terms of logic architecture parameters as shown in equation 3.18, where n_c and p represent the number of clusters and Rent's exponent respectively. We incorporate logic architecture parameters into equation 3.15 by approximating W_N using Davis' wirelength model.

$$\bar{R} = \frac{\frac{p-0.5}{p} - \sqrt{n_c} - \frac{p-0.5}{6\sqrt{n_c}(p+0.5)} + \frac{-p-1+4^{(p-0.5)}}{2p(p+0.5)(p-1)}n_c^p}{1 + \frac{-2p-1+2^{(2p-1)}}{2p(p-1)(2p-3)}n_c^{(p-0.5)} - \frac{p-0.5}{6p\sqrt{n_C}} - \frac{(p-0.5)\sqrt{n_c}}{p-1}} \tag{3.18}$$

We revise equation 3.17 by substituting $\bar{R}$ with equation 3.18, and then replacing W_N in equation 3.15 with its new expression. We will refer to this new model as Model^D.

Step 8: Feuer has also introduced a wirelength model based on logic architecture parameters as shown in equation 3.19. We derive a second wirelength model by replacing Davis' model in step 7 with Feuer's model. We will refer to this model as Model^F.

$$\bar{R} = \frac{2\sqrt{2}\left(3+3p\right)}{\left(1+2p\right)\left(2+2p\right)}n_c^{(p-0.5)} \tag{3.19}$$

3.4 Validation Methodology

The quality of an analytical model is quantified by the degree of correlation between the model, and the experimental values generated by CAD tools with respect to a

set of benchmarks. We use MCNC and VTR benchmarks, and employ the Mean Absolute Percentage Error ($MAPE$) and the Correlation Ratio ($c-ratio$) metrics to evaluate the quality of our wirelength model. Despite that the actual model's performance is quantified by the benchmarks from the validation set, the $MAPE$ and $c-ratio$ values for the training set are presented for completeness.

Mean Absolute Percentage Error ($MAPE$) is defined by equation 3.20

$$MAPE = \frac{1}{C} \sum_{\forall Circuits} \frac{|Predicted - Measured|}{Measured} \tag{3.20}$$

where the *Predicted*, and *Measured* refers to the model, and experimental output respectively.

Correlation Ratio ($c-ratio$) is defined by equation 3.21

$$c-ratio = \frac{Number\ of\ non-violating\ pairs}{Total\ number of pairs} \tag{3.21}$$

where a *pair* denotes a pair of data points containing experimental output and its corresponding model output. A *pair* is defined as *non-violating* when the model output resonates with the experimental output. For instance, an increase in the experimental value for a data point with respect to the previous data point should show an increase in the modeled value or vice-versa.

We use the Rent's exponents extracted before the technology mapping stage as reported in [12] for the 20 MCNC benchmarks. In addition, 6 out of the 8 benchmarks from the Verilog-to-Routing (VTR) tool [57] that do not have hard-IPs are selected. 2 of the benchmarks (i.e., *stereovision2*, *stereovision3*) are excluded due to the exceeding long time[1] that is required to sweep through the entire architecture exploration search space (i.e., total of 88 architectures explored). To extract the pre-techmapped Rent's exponent, the 6 VTR benchmarks are synthesized using the

[1]Each benchmark, *stereovision2* and *stereovision3*, require a longer execution time than the largest benchmark, *stereovision1*, in our 6 selected VTR benchmarks that require more than 10 days to sweep through the entire architecture design space.

Quartus II tool before feeding the *blif* netlist to the *bcgen* [46]. This is required in order for *bcgen* to process the netlist. Tables 3.1 and 3.2 summarises the circuit details.

Our CAD experimental testbed uses the technology mappers EMap [41] for MCNC [2] and ABC [59] for VTR benchmarks. Then, we sweep the logic and routing architecture parameters over a predefined range as summarized in table 2, using timing-driven clustering algorithm (T-VPack [47]) followed by VTR for placement and routing (i.e., VPR7.0). Then, analysing each configuration parameter involves calculating the average $M\hat{A}PE$ and $c-\hat{r}atio$ metrics across the sweeping range as depicted by equations 3.22 and 3.23, where n refers to the specific parameter's sweeping range. Figure 3.3 denotes our methodology flowchart.

$$M\hat{A}PE = \frac{1}{n}.\sum_{i=n_{min}}^{n_{max}} MAPE_i \tag{3.22}$$

$$c-\hat{r}atio = \frac{1}{n}.\sum_{i=n_{min}}^{n_{max}} c-ratio_i \tag{3.23}$$

In section 3.3.3, we proposed two wirelength models denoted as Model^F and Model^D. These two models are trained based on section 3.3.2 and reviewed for their quality in capturing the experimental trends with the logic architecture parameters by comparing the $c-\hat{r}atio$ and $M\hat{A}PE$ performance. We also compare our models against the current state-of-the-art FPGA-based wirelength model developed by Smith et al. Then, the better model (i.e., either Model^F or Model^D) is further evaluated for its robustness by varying the routing objectives (i.e., F_S, F_{Cin}, F_{Cout}, L, W).

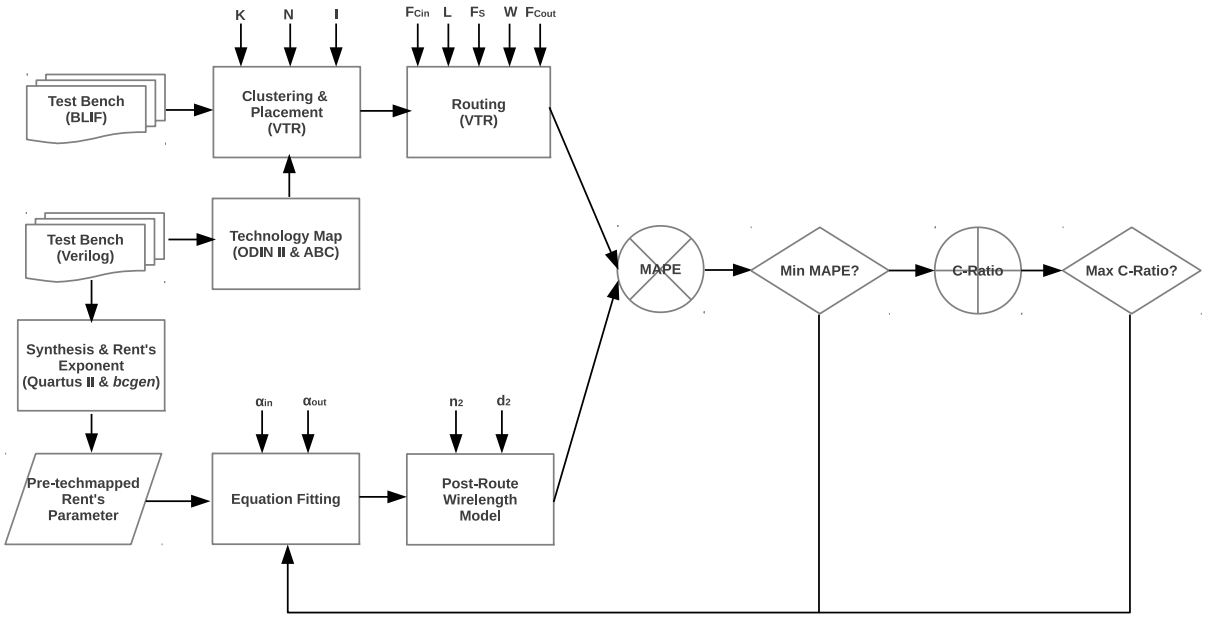
A model is robust only if its predictions are in agreement with changing objectives of the the CAD flow. Therefore, we investigate the robustness of our model by varying the routing optimization objectives (i.e., timing-driven, breadth-first) using

[2]The default MCNC benchmarks in *blif* format inherited from the VTR flow is tech-mapped using the technology mapper, EMap [41].

Table 3.4: FPGA logic and routing architecture parameters.

Parameter	Sweeping Range
LUT size, K	2-7
Logic block size, N	2-20
No. of logic block inputs, I	4-40
Switch box flexibility, F_S	3-21
Input connection box flexibility, $F_{C_{in}}$	0.1-1.0
Output connection box flexibility, $F_{C_{out}}$	0.1-1.0
Channel Width, $W = W_N \times (1 + CW_I)$	$CW_I \in \{10 - 200\%\}$
Segment length, L	1-6
Logical Equivalence, Eqv	1

Figure 3.3: Wirelength model validation methodology



both the MCNC and VTR benchmarks from the training and validation set. The entire framework is implemented in Matlab, and the source code is available at [56].

3.5 Model Validation

3.5.1 Feuer's and Davis' Model Impact on Post-Routing Wirelength

We present the change in average wirelength with respect to logic architecture parameters K, N and I in figures 3.4, 3.5, and 3.6 respectively. The comparison study reveals that Model^D consistently correlates with experimental results better than Model^F. As shown in figures 3.4 and 3.6, Model^D exhibits a much closer and consistent curve gradient with the empirical curve compared to Model^F throughout the sweeping range. For all the three figures, Model^F has a tendency to over-estimate while Model^D has a tendency to under-estimate. However, Model^D's under-estimation is reasonable.

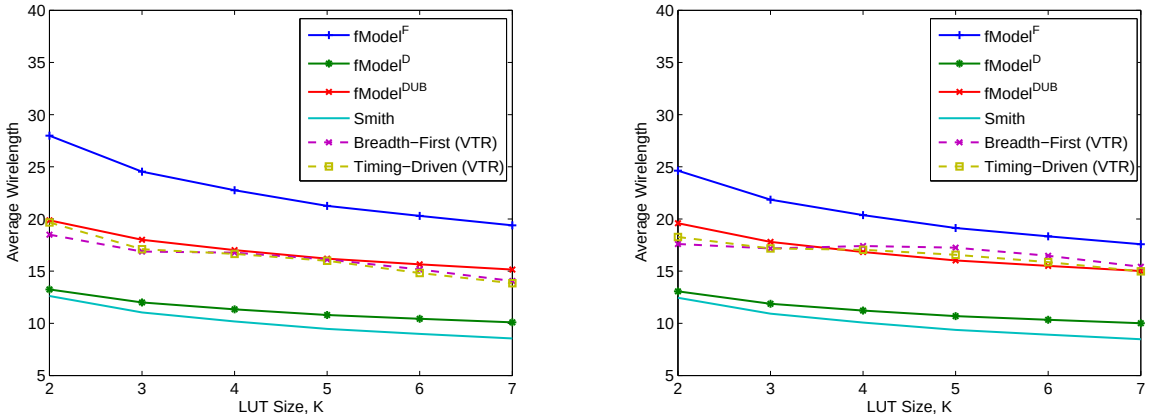
This is because, Hwang et al. [30] has proven theoretically that the minimum Steiner tree wirelength is approximately $\frac{3}{2}$ times smaller than the spanning tree wirelength. We apply this coefficient on Model^D to derive an upper-bound model (i.e., after multiplying the compensation factor of $\frac{3}{2}$), which we refer to as Model$^{D(UB)}$. In the following sections, we will evaluate both Model^D and Model$^{D(UB)}$ with respect to Smith's model.

3.5.2 Analysis of Model^D on Logic Block Parameters

We perform evaluations based on both an N-limited and I-limited FPGA architecture[3]. The number of logic inputs for an N-limited architecture is calculated using the expression, $I = \lceil \frac{K}{2}.(N+1) \rceil$). The I-limited architecture is exposed as we sweep I while fixing K and N. This allows us to verify our wirelength model based on both architectural extremes. To generate the channel width (W), a minimum

[3] N-limited - an architecture where the number of LUTs that can be packed into a logic cluster (CLB) is limited by the cluster size, N. I-limited - an architecture where the number of LUTs that can be packed into a logic cluster (CLB) is limited by the number of inputs of the cluster, I.

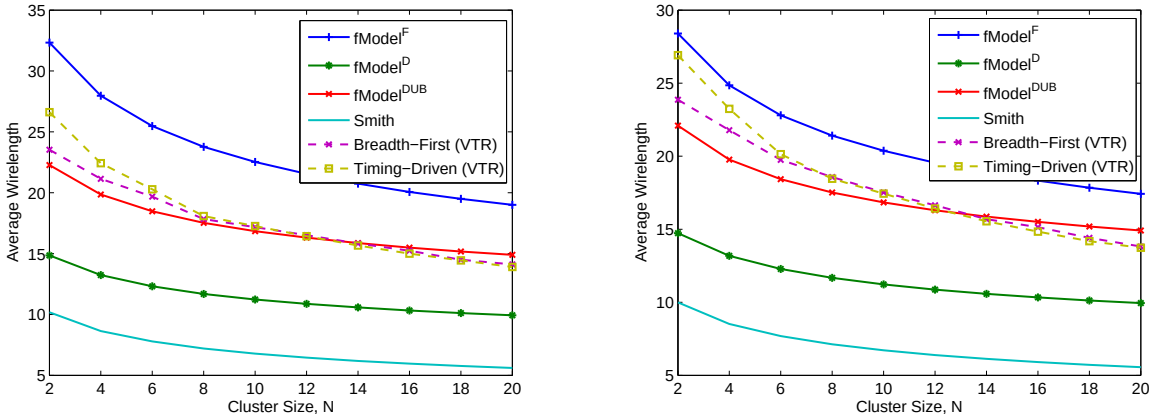
Figure 3.4: Sweep LUT Size, K ($N = 10$)



(a) Training Set

(b) Validation Set

Figure 3.5: Sweep Cluster Size, N ($K = 4$)



(a) Training Set

(b) Validation Set

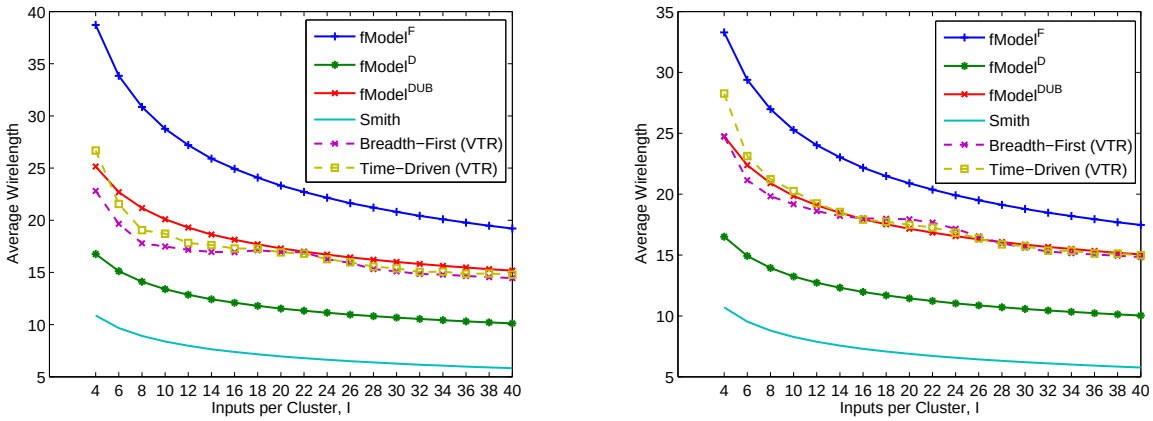
Table 3.5: Performance comparison of Model^F, Model^D and Smith

Parameter	Objective	Model^F		Model^D		Smith	
		$M\hat{A}PE$	$c-\hat{r}atio$	$M\hat{A}PE$	$c-\hat{r}atio$	$M\hat{A}PE$	$c-\hat{r}atio$
K	timing	0.216	1.0	0.329	1.0	0.400	1.0
	routing	0.201	0.833	0.338	0.833	0.407	0.833
N	timing	0.175	1.0	0.352	1.0	0.612	1.0
	routing	0.189	1.0	0.346	1.0	0.607	1.0
I	timing	0.209	1.0	0.341	1.0	0.603	1.0
	routing	0.233	1.0	0.328	1.0	0.600	1.0
F_S	timing	0.077	1.0	0.380	1.0	0.592	0.0
	routing	0.093	1.0	0.370	1.0	0.585	0.0
$F_{C_{in}}$	timing	0.231	1.0	0.319	1.0	0.585	0.0
	routing	0.222	1.0	0.324	1.0	0.589	0.0
$F_{C_{out}}$	timing	0.178	0.2	0.350	0.2	0.609	0.0
	routing	0.162	0.3	0.358	0.3	0.615	0.0
L	timing	0.252	1.0	0.334	1.0	0.569	0.0
	routing	0.238	1.0	0.341	1.0	0.573	0.0
W	timing	0.126	0.5	0.379	0.5	0.620	0.0
	routing	0.177	0.6	0.351	0.6	0.603	0.0

Table 3.6: Performance comparison of Model$^{D(UB)}$ and Smith$^{(UB)}$

Parameter	Objective	Model$^{D(UB)}$		Smith$^{(UB)}$	
		$M\hat{A}PE$	$c-\hat{r}atio$	$M\hat{A}PE$	$c-\hat{r}atio$
K	timing	0.03	1.0	0.107	1.0
	routing	0.056	0.833	0.131	0.833
N	timing	0.073	1.0	0.418	1.0
	routing	0.052	1.0	0.411	1.0
I	timing	0.0187	1.0	0.404	1.0
	routing	0.025	1.0	0.393	1.0
F_S	timing	0.07	1.0	0.387	0.0
	routing	0.0554	1.0	0.377	0.0
$F_{C_{in}}$	timing	0.036	1.0	0.378	0.0
	routing	0.0193	1.0	0.383	0.0
$F_{C_{out}}$	timing	0.024	0.2	0.414	0.0
	routing	0.038	0.3	0.422	0.0
L	timing	0.030	1.0	0.353	0.0
	routing	0.038	1.0	0.360	0.0
W	timing	0.068	0.5	0.430	0.0
	routing	0.027	0.6	0.404	0.0

Figure 3.6: Sweep Input-per-Cluster, I ($K = 4$, $N = 10$)



(a) Training Set

(b) Validation Set

channel width is first generated using the breadth-first routing algorithm. Then, a 30% channel width relaxation is performed to prevent excess routing stress [5]. We set the rest of the parameters, switchbox flexibility (F_S) to 3, segment length (L) to 4, connection box flexibilities ($F_{C_{in}}$ and $F_{C_{out}}$) to 0.25 and 1.0 respectively. The switch box, and connection box flexibilities are set to typical values used in previous modeling works [44].

Varying LUT size (K): As the LUT size increases, more functionality can be packed into the logic block and less number of LUTs are needed to implement a design. Inter-cluster wirelength is the main contributor to the overall pin-to-pin wirelength. Therefore the decrease in the overall number of LUTs will inadvertently lead to a decrease in expected wirelength. Here, we set N to 10 and observe this decreasing trend for the experimental results in figure 3.4. Compared to Smith's wirelength model with a $M\hat{A}PE$ value of 0.407, Model^D achieves a $M\hat{A}PE$ value of 0.338. We observe a low $M\hat{A}PE$ of 0.056 with a high $c - \hat{r}atio$ of 1.0 for Model$^{D(UB)}$ with the validation set of benchmarks. This shows that our model not only correlates well with empirical trend, but also deviates lesser from the empirical values.

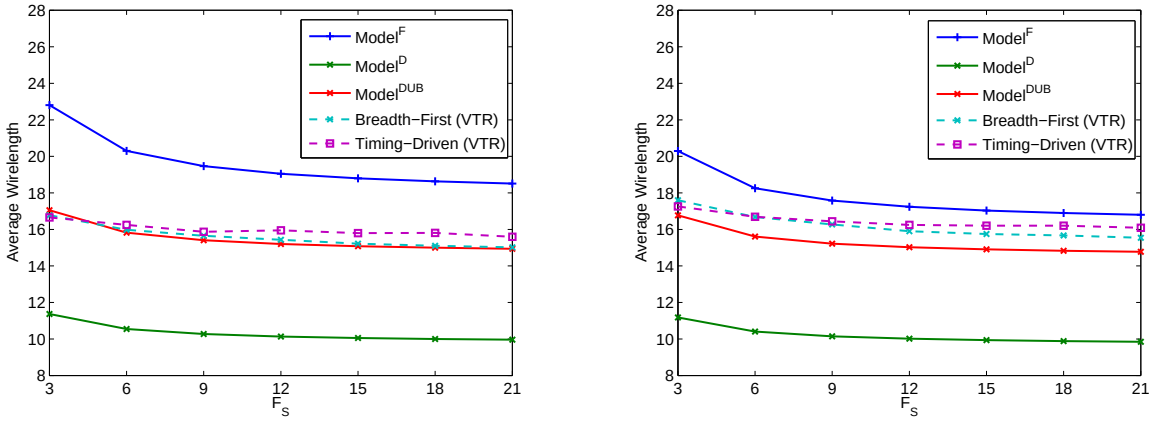
Varying logic block size (N): In this experiment, to lift the inputs per block constraint, we evaluate the block size in a N-limited architecture with K fixed to 4.

As N increases, the reduction in number of CLBs will lead to a decrease in average wirelength. Our model captures the experimental trend better than Smith's model ($M\hat{A}PE$ of 0.607) as shown in figure 3.5 with a $c - \hat{r}atio$ of 1.0 and a $M\hat{A}PE$ of 0.346. Model$^{D(UB)}$ results with values very close to experimental values with a $c - \hat{r}atio$ of 1.0 and $M\hat{A}PE$ of 0.052 with the validation set of benchmarks. However, Model$^{D(UB)}$ has a tendency to under-estimate with smaller N and over-estimate with bigger N.

Varying inputs-per-block (I): In this experiment we fix K and N to 4 and 10 respectively. As I increases, we see a transition from an I-limited architecture to an N-limited architecture. When I is small (i.e., $I < 22$), I-limited architecture, the amount of logic that a block can accommodate is limited by the available number of inputs. Hence, the expected number of excess/unused LUTs per block will increase. This increase will cause an excessive inter-cluster routing overhead due to a larger number of required CLBs. When I increases, this relaxation allows utilization of LUTs within the block causing a rapid reduction in the wirelength. When I is large (i.e., $I > 22$), N-limited architecture, the amount of logic that can fit into a block is limited by the available number of LUTs. Given an adequate number of inputs per block, the number of CLBs needed to implement a circuit will reach its minimum when majority of the CLBs are full. This corresponds to the minimum number of inter-cluster connections. Therefore, beyond a certain I value (i.e., $I > 35$), reduction amount in wirelength will saturate. Our model follows these experimental trends shown in figure 3.6 with a $c - \hat{r}atio$ of 1.0 and a $M\hat{A}PE$ of 0.328, while we observe a $M\hat{A}PE$ of 0.6 with Smith's model. Furthermore, with the upper-bound model (Model$^{D(UB)}$) we achieve a $M\hat{A}PE$ of 0.025 and a $c - \hat{r}atio$ of 1.0 with the validation set.

Figures 3.4, 3.5, and 3.6 also illustrate that the model correlates well with the experimental values despite using three different routing algorithms. In all cases, our model achieves a wirelength closer to empirical values than Smith's model.

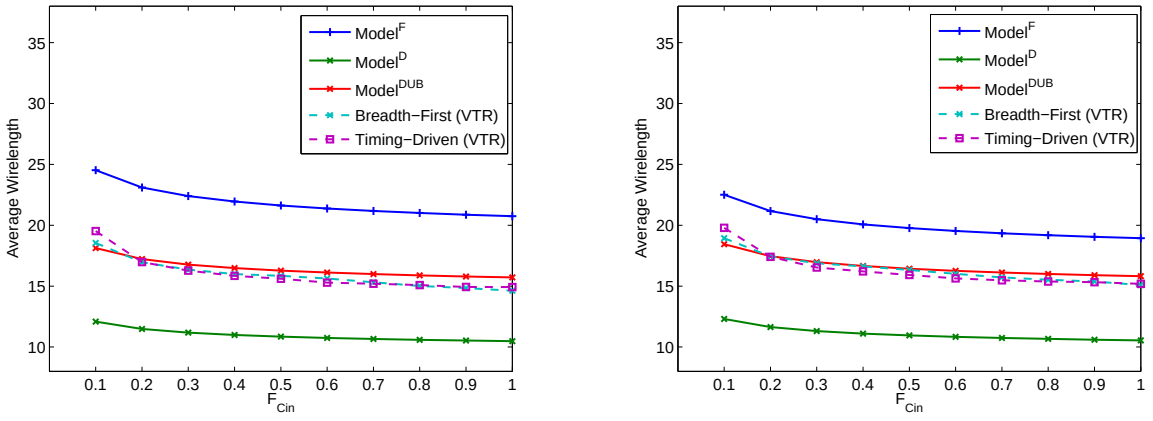
Figure 3.7: Sweep SB Flexibility, F_S ($F_{Cin} = 0.25$, $F_{Cout} = 1.0$, $L = 4$)



(a) Training Set

(b) Validation Set

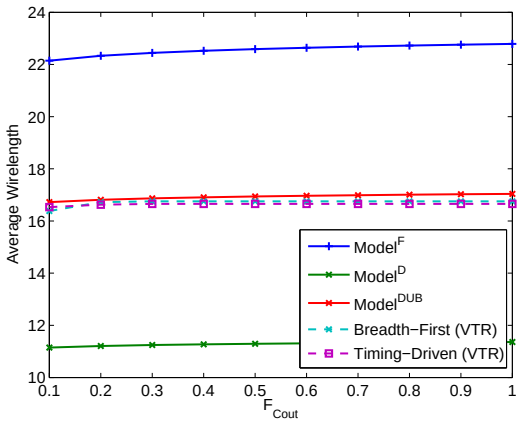
Figure 3.8: Sweep Input CB Flexibility, F_{Cin} ($F_S = 3$, $F_{Cout} = 1.0$, $L = 4$)



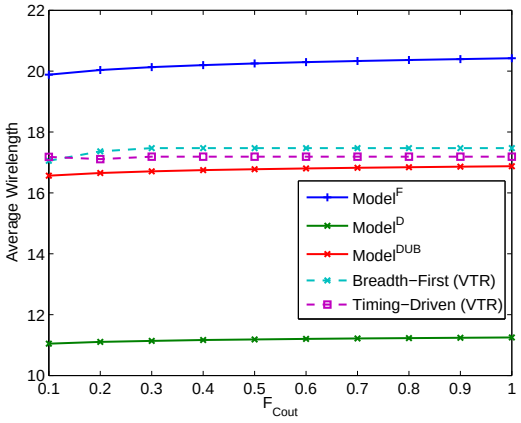
(a) Training Set

(b) Validation Set

Figure 3.9: Sweep Output CB Flexibility, F_{Cout} ($F_S = 3$, $F_{Cin} = 0.25$, $L = 4$)

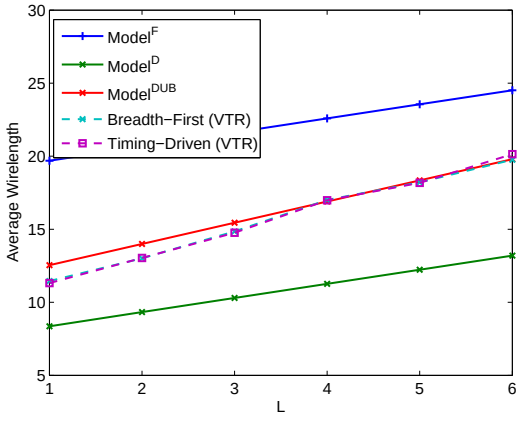


(a) Training Set

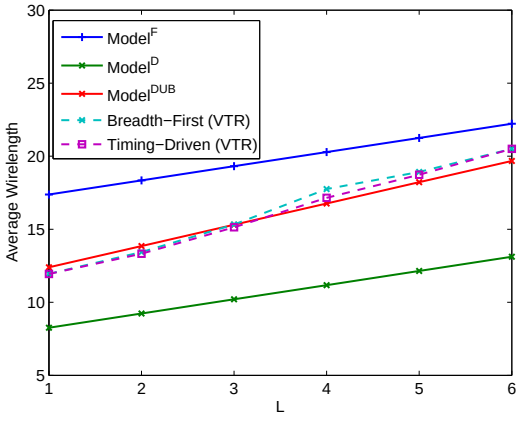


(b) Validation Set

Figure 3.10: Sweep Segment Length, L ($F_S = 3$, $F_{Cin} = 0.25$, $F_{Cout} = 1.0$)

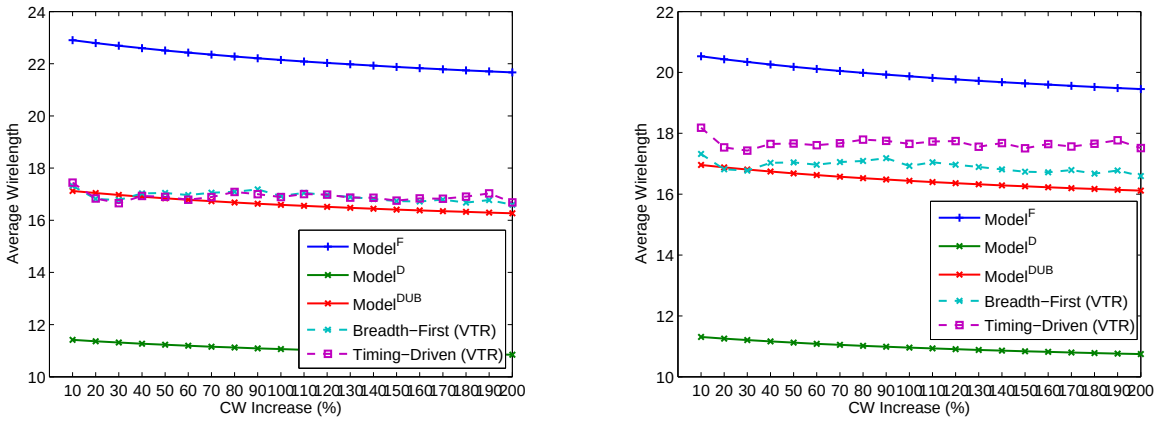


(a) Training Set



(b) Validation Set

Figure 3.11: Sweep Channel Width Relaxation, CW (Default routing architecture)



(a) Training Set

(b) Validation Set

3.5.3 Analysis of Model^D on Routing Parameters

Figures 3.10 through 3.11 illustrate the quality of our wirelength model by sweeping routing architecture parameters individually. In these experiments, we set the logic architecture parameters, K to 4, N to 10, and I to 22. We fix each routing architecture parameter, F_S, F_{Cin}, F_{Cout} and L to 3, 0.1, 0.1 and 1 respectively when it is not under evaluation. Similar to logic block architecture experiments, input channel width is pre-generated using breadth-first routing algorithm followed by relaxation.

Varying switch box flexibility (F_S): When F_S is small, the average wirelength is long. This is because, given a limited number of programmable switches in a switch-box, a router is constrained while resolving congestion, causing an increase in the expected number of routing turns and segments. As F_S increases, router has more flexibility, hence a higher probability of finding a shorter path. However, as F_S gets larger, the switching flexibility is no longer a constraint for finding the shorter path. Therefore, the average wirelength is expected to remain relatively constant beyond a certain F_S. We observe this trend in figure 3.7 with a saturation when $F_S > 12$. Compared with Smith's model, which reports a $M\hat{A}PE$ of 0.585, Model^D accurately captures the impact of F_S with a $c - \hat{r}atio$ of 1.0 and a $M\hat{A}PE$ value of 0.37. We further improve the accuracy using Model$^{D(UB)}$ with a $M\hat{A}PE$ of 0.093

and a $c - \hat{r}atio$ of 1.0 with the validation set.

Varying input connection box flexibility (F_{Cin}): When F_{Cin} is small, the average wirelength is long. This is because, given a small number of routing switches in connection-boxes, the router has fewer number of options for ensuring that an allocated track is connected to an input pin. This will lead to an increase in routing turns and number of segments. As F_{Cin} increases, the relaxation allows more number of direct connections for the input pins of a block, resulting with a reduction in the average wirelength. Compared with Smith's wirelength model, which reports a $M\hat{A}PE$ of 0.589, figure 3.8 shows that Model^{D} accurately captures this trend with a $M\hat{A}PE$ of 0.324 and $c - \hat{r}atio$ of 1.0. Model$^{D(UB)}$ has further improvement with a $M\hat{A}PE$ of 0.019 and a $c - \hat{r}atio$ of 1.0 with the validation set.

Varying output connection box flexibility (F_{Cout}): Intuitively, one might feel that increasing F_{Cout} will lead to a decrease in wirelength. However, based on our experimental results, we observed an opposing behaviour despite using an up-to-date routing switch-box (i.e., Wilton switch-box) with uni-directional routing fabric. This is because an increase in the number of output track choices (i.e., F_{Cout}) will simply imply that an output pin is faced with a higher possibility of connecting to a track which will be in a different domain that destination pin will have access to. As a result, this will lead to additional routing overhead. Furthermore, the breadth-first routing algorithm in VPR does not have domain negotiation for the breadth-first search algorithm [67] which is demonstrated by a steeper slope in Figure 3.9. In order to capture a typical routing flexibility, we impose a 30% channel width relaxation in our experiments. Compared with Smith's model, which reports a $M\hat{A}PE$ of 0.615, Model^{D} accurately captures this trend with a $M\hat{A}PE$ of 0.358, and $c - \hat{r}atio$ of 0.3. Compensation factor further improves the accuracy (Model$^{D(UB)}$) with a $M\hat{A}PE$ of 0.038, and a $c - \hat{r}atio$ of 0.3 with the validation set.

Varying segment length (L): Wire segments in horizontal and vertical channels may span the width of a single block ($L = 1$) or multiple blocks ($L > 1$). When segment length is short, long connections are established using multiple wire segments and programmable switches. As the segment length increases, number of connec-

tions needing wires shorter than the segment length itself will increase. Connections will then require extra turns through the switchboxes, which will increase the stress on routing. These factors will result with an increase in the average wirelength. Our model captures this experimental trend shown in figure 3.10 almost linearly as we sweep L with a $M\hat{A}PE$ of 0.341, and $c - \hat{r}atio$ of 1.0. With Model$^{D(UB)}$, we observe further improvement with a $M\hat{A}PE$ of 0.038 with the validation set. On the other hand, Smith's model is not sensitive to changes in L with a $M\hat{A}PE$ of 0.573.

Varying channel width (W): As the channel width increases, the router has more options for finding a path between the source and destination pins. This relaxation increases the possibility finding a shorter route. As our wirelength model adheres to the routability-driven routing algorithm, a breadth-first search will require the router to sweep all possible tracks that leads to the shortest path. Hence, this will result in a shorter average wirelength. We observe these trends in figure 3.11 with a negative trend gradient as we sweep CW_I. Compared with Smith's model, which reports a $M\hat{A}PE$ of 0.603, Model^{D} accurately captures this trend with a $M\hat{A}PE$ of 0.351 and $c - \hat{r}atio$ of 0.6. Model$^{D(UB)}$ on the other hand results with a close estimation of the experimental data with a $M\hat{A}PE$ of 0.027 and a $c - \hat{r}atio$ of 0.6.

In overall, since Smith's wirelength model does not take routing architecture parameters into account, it is not able to capture the effect of changes on wirelength as accurate as our model. Furthermore, the model's estimations are often lower than the empirical results. This behavior is expected as the model assumes a fully flexible routing architecture.

In this section, we show that our model is able to correlate well with single-variable (i.e., logic and routing architecture parameters) architecture trends. Analysing the variation and correlation with multi-variate objectives is a natural next step which we feel is an interesting future work to extend on our post-routing wirelength model. This can be achieved by analysing empirical and model results by analysing various parameter pairs. We believe that comparing the waveform charts of experimental and analytical results would give insights to the fidelity of our model.

3.6 Application of Our Model

In this section, we present an architecture optimization case study. The objective of this case study is to derive the best output connection box flexibility, F_{Cout}, to optimize the critical path delay with an area constraint using a model-based approach. We imposed a 30% channel width relaxation [5]. Then, the architecture is optimized assuming a timing-driven routing algorithm (i.e., similar to PathFinder).

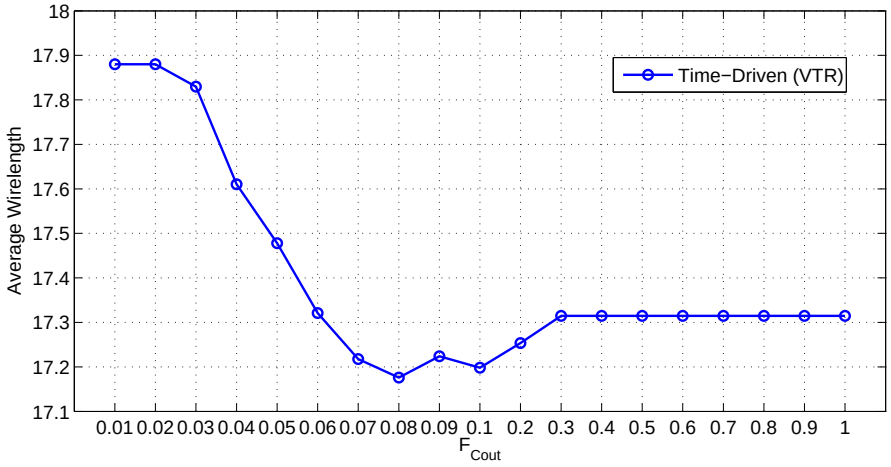
The investigation comprises of analysing the impact of F_{Cout} over two regions, (1) 0.01 to 0.09 (i.e., assuming that the device's channel width will be smaller than 1000) and (2) 0.1 to 1.0.

To anticipate the solution to our study, we first sweep F_{Cout} value over the two ranges using VTR-7.0. The rest of the architecture parameters (K, N, I, F_S, F_{Cin}) are set to their default values. Based on the empirical output, our model-based approach should also derive an equivalent connection-box flexibility of $F_{Cout} = 0.07$.

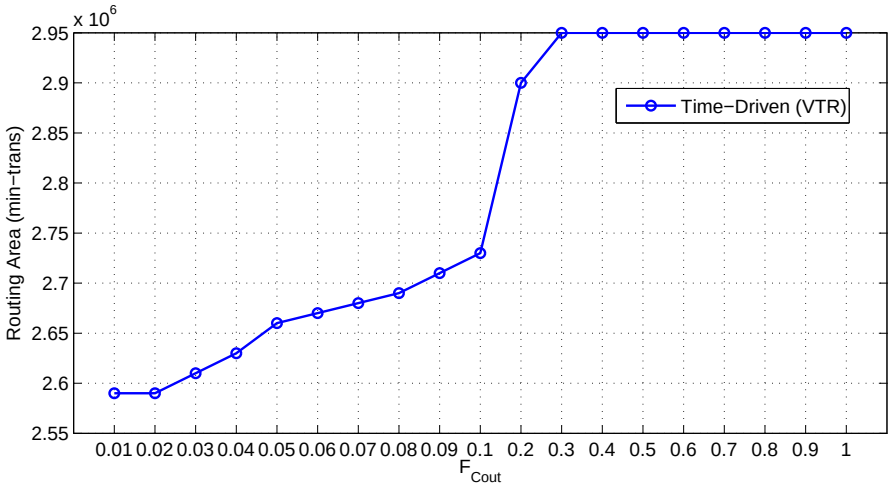
Figures 3.12(c), 3.12(d) and 3.12(e) denote the average wirelength, routing area and critical path delay obtained from CAD experimentations. One important observation is that, the average wirelength decreases with F_{Cout} as it varies from 0.01 to 0.09. The decreasing average wirelength trend is due to the increase in F_{Cout} that relieves the routing congestion by providing more output track selections resulting in shorter paths from the source to the destination pins. One might feel that this decreasing trend might be observed in the region when F_{Cout} is greater than 0.1 as well. However, since majority of the FPGA community will at least introduce a greater than 30% channel width relaxation margin in order to maintain a fast routing time [34] and optimal wirelength performance [5], we postulate that this decreasing trend will mainly be encapsulated within the range of 0.01 to 0.09. Therefore, our study represents a scenario that is applicable to most typical applications.

3.6.1 Region 1: Analysing the flexibility, $F_{Cout} > 0.1$.

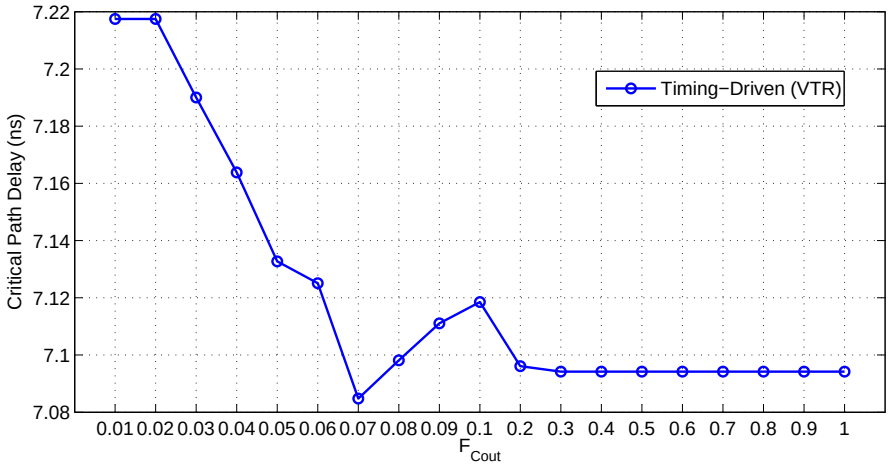
Our model-based approach starts by analysing the wirelength trends when F_{Cout} is greater than 0.1 (i.e., 0.1 1.0).



(c) Varying F_{Cout} on average wirelength



(d) Varying F_{Cout} on routing area



(e) Varying F_{Cout} on critical path delay

Figure 3.12: Study the effect of F_{Cout} over the ranges $[0.01 - 0.09]$ and $[0.1 - 0.9]$.

Step 1: Deriving the value of F_{Cout} which corresponds to the minimum area for the range of 0.1 to 1.0. Due to the strong correlation between routing area and average wirelength (i.e., a smaller average wirelength will imply a smaller routing area), it is important to observe the average wirelength as F_{Cout} increases. Intuitively, one may think that with an increasing F_{Cout}, a more flexible routing interconnect fabric will lead to shorter wirelength, implying a smaller routing area. However, our wirelength model demonstrates the opposite. This is because, with the negotiated congestion routing algorithm, the non-critical paths are not optimized for a shorter wirelength in order to maintain a low routing congestion. Therefore, the routing multiplexers/switches needed to support these non-critical paths with an increase in F_{Cout} leads to the increase in routing area. Hence, in order to optimize area, a smaller F_{Cout} is desirable. Our post-routing wirelength model correlates with the empirical observation in Figure 3.12(d). Using the model, one can observe that a minimum area can be achieved with an F_{Cout} of 0.1.

Step 2: Starting from $F_{Cout} = 0.1$ of step 1, we derive the F_{Cout} range that will encompass the best critical path delay performance. Then, CAD tools are launched to explore this smaller search space to locate the F_{Cout} with the optimum delay performance. Intuitively, the larger the F_{Cout}, the smaller the routing congestion and critical path delay. Hence, a F_{Cout} of 1.0 will contribute to a small critical path delay. However, since an increasing F_{Cout} will lead to an increase in routing area overhead, it is important to identify the saturation point in critical path delay where the area is minimum. Using the post-routing wirelength model, it can be observed that the average wirelength trend will start to plateau with a F_{Cout} value of larger than 0.4. Therefore, as the critical path delay has a strong correlation with the average wirelength, conducting CAD experiments for the first 0.1 to 0.4 flexibilities is adequate for an architect to locate the optimal point of $F_{Cout} = 0.25$ (i.e., corresponding to optimal delay with small area overhead).

3.6.2 Region 2: Analysing the flexibility, $F_{Cout} < 0.1$.

Step 3: Deriving the F_{Cout} which corresponds to the smallest critical path delay with minimum area overhead for the range of 0.01 to 0.09. Since the post-routing wirelength model does not support this range, a simple algorithm is proposed to analyse this region. The algorithm is describe as following:-

- Launch CAD flow for $F_{Cout} = 0.02, 0.04, 0.06, 0.08$.
- Start analysing the critical path delay trend beginning from $F_{Cout} = 0.1$ to $F_{Cout} = 0.01$.
- Compare the critical path delay trend of the regions $F_{Cout} = 0.1 \rightarrow 0.08$ and $F_{Cout} = 0.08 \rightarrow 0.06$.
- If the critical path delay is decreasing in region $F_{Cout} = 0.1 \rightarrow 0.08$ and increasing in region $F_{Cout} = 0.08 \rightarrow 0.06$ (i.e., there exist a minimum point), launch CAD flow for $F_{Cout} = 0.09$ and $F_{Cout} = 0.07$ to obtain the minimum F_{Cout}.
- The same process repeats for the regions $F_{Cout} = 0.06 \rightarrow 0.04$ and $F_{Cout} = 0.04 \rightarrow F_{Cout} = 0.02$.

In our model-based approach, an extra CAD experiment will be conducted with $F_{Cout} = 0.07$. $F_{Cout} = 0.07$ also corresponds to the output connection box flexibility which minimizes critical path delay with the minimal area overhead in the region $F_{Cout} < 0.1$.

Step 4: Comparing the F_{Cout} derived for the regions $F_{Cout} > 0.1$ and $F_{Cout} < 0.1$, an overall optimal output connection-box flexibility of $F_{Cout} = 0.07$ is obtained.

3.6.3 Architecture improvements over the default configurations.

Comparisons between the model-based approach and the empirical approach is performed. We examine the difference in exploration time between the two approaches.

Table 3.7 illustrates the execution time for both the timing-driven place and route time using VPR-7.0 and the MATLAB script for the post-routing wirelength model. It can be observed that using a model-based approach, a 53.4% reduction in design space exploration time can be achieved.

The architecture improvements are quantified by comparing between the newly derived, $F_{Cout} = 0.07$ versus the VTR architecture default $F_{Cout} = 0.1$. Table 3.8 depicts the improvements in terms of critical path delay, maximum circuit frequency and routing area in terms of minimum width transistor area [69].

We quantify the improvements from the manufacturing perspective in terms of the gross Die Per Wafer (DPW) for a die area reduction of 1.9%. Assuming a die size of 12x12mm with a 300mm wafer, an area reduction of 1.9% will amount to an increase in yield by *9* dies per wafer. Equation 3.24 depicts the formula to calculate DPW [73].

$$DPW = d\pi \left(\frac{d}{4S} - \frac{1}{\sqrt{2S}} \right) \tag{3.24}$$

where d refers to the wafer diameter (in mm) and S refers to the die area (in mm^2).

3.7 Summary

In this chapter, we derived the first analytical post-routing wirelength model that takes into account both logic and routing architectural parameters for a homogeneous FPGA architecture. We evaluated the integrity of our model based on its

Table 3.7: Empirical versus Model-Based design exploration time.

Approach	Place & Route (*Hours*)	MATLAB Script Execution (*Hours*)
Empirical (Baseline)	1.25	0.0
Model-Based	0.39	<0.01 ($\approx 4secs$)
Improvement (%)		53.4

Table 3.8: VTR default versus Model-Based performance summary.

Architecture (F_{Cout})	Critical Path Delay (ns)	Frequency (MHz)	Area ($min\text{-}trans$)
Model (0.07)	7.085	14.12	2,680,000
Default (0.1)	7.119	14.15	2,730,000
Improvement (%)	0.5	0.5	1.9

accuracy ($M\hat{A}PE$) and trend correlation ($c - \hat{r}atio$). The model is CAD independent and is able to capture the variations in both logic and routing architecture parameters consistently with respect to the empirical results from CAD experimentations. In this work, we also presented the differences between the post-placement and post-routing wirelength. The post-placement wirelength model is able to provide accurate experimental trend analysis with varying logic architecture parameters but deviates further from actual experimental values as we decrease the routing fabric flexibilities. Hence, architects are unable to use the model to predict correlation trends with routing architecture parameters. In this work, we generate a set of constants (i.e. β, α_{in}, α_{out}) via curve-fitting techniques and prove that the values remain constant over our selected suite of benchmark circuits. However, it is worth exploring where these constants hold their value for different benchmarks and whether these constants could be generalized or not. If they are shown to be generic, then the impact of such a contribution will be as dramatic as Rent's coefficient especially for future analytical models, FPGA architectures and CAD tools.

CHAPTER 4

Multiplexer Logic Utilization Hybrid Model

4.1 Introduction

Due to the consumers' insatiable need to support a large genre of applications, there is an increase in demand for higher computational power in today's devices. This requirement is driving an emerging trend of ASIC SoC platforms (e.g., QualComm's SnapDragon [71], AMD's Geode [70], nVidia's Tegra [72], etc) in the embedded market.

However, designing a robust FPGA-based SoC platform is non-trivial and time consuming. This is mainly due to the fact that application developers are still limited to low-level RTL system specification tools to design digital hardware [27]. Despite that RTL allows fine granularity descriptions of the design, implementing a large and complex system like the SoC with these low-level tools is highly error-prone and prevent designers/architects from focusing on the overall architecture [27]. Furthermore, the multi processor SoC (MPSoC) platforms are often desired in the mobile devices that have strict performance constraints (i.e., power, delay). As the communication backbone (i.e. crossbar switches) is a critical component for ensuring performance efficiency, it is imperative to design a lightweight yet power-efficient crossbar switch [62]. Currently, to meet the performance constraints (e.g., area, delay, power), time consuming Computer Aided Design (CAD) experiments are invoked during each design iteration.

Recently, there is a proliferation of FPGA analytical models targeting various performance metrics (e.g., logic utilization [18], wirelength [3], routability [20], delay [28], energy [55], etc) for designers to evaluate an architecture based on a set of configuration parameters. These tools are able to provide fast and fairly accurate trend analysis at design time, eliminating the requirement to conduct time consuming

CAD-based experiments that sweep the configuration search space. Unfortunately, these models only predict the average or expected performance metric values based on a generic class of applications. Hence, they suffer from accuracy issues, namely *discrete effects*, when used to estimate the performance of specialized circuits like the crossbar switches [19]. In this paper, we propose a model that combines both the analytical and algorithmic approaches [39] to estimate the logic utilization (i.e. number of Look-Up Tables, LUTs) of crossbar switches.

We evaluate the quality of our model with mean absolute percentage error (MAPE) and correlation metrics (c-ratio) against CAD experiments and the state-of-the-art analytical tools. In section 4.2, we give an overview of the related work, and establish the research problem in section 4.3. In section 4.4, we describe the model development approach followed by the methodology and testbed used for evaluating the quality of our model in section 4.5. Detailed analysis of the results are presented in section 4.6, followed by conclusions and future work in section 4.7.

4.2 Related Work

FPGA analytical models enable fast architecture exploration and eliminate the need for conventional CAD based evaluations that require sweeping the configuration parameter values exhaustively. Majority of these models are derived from the ASIC domain, which are based on Rent's rule. In order to create an unbiased model, ASIC analytical models assume a *generic* circuit that is infinitely large with a combination of hardware blocks having unique circuit characteristics (i.e., number of fan-ins/outs, logic block size, etc) [22]. These hardware blocks communicate with each other through a fully flexible routing interconnect [22]. Hence, given the reconfigurable nature of logic blocks and routing interconnects, there is a strong resemblance between the ASIC model and the conventional island-style FPGA.

Since the FPGA analytical models are designed mainly for architecture design explorations, current models' estimations will deviate from actual empirical values for atypical application genres. Das et al. indicate that circuits that are less generic

(e.g., circuits with hardware blocks having the same number of fan-ins/outs) will exhibit *discrete effects* [19]. As opposed to analytical models, which always exhibit a smooth trend, *discrete effects* are disruptions in the corresponding empirical curve, which emerge as discontinuities while sweeping the architecture parameters. The authors illustrate large deviations between model and empirical values due to these effects by analysing the logic utilization of crossbar switches when sweeping the LUT size, K. Our work complements this study with a new accurate logic utilization model that addresses the problem of *discrete effects*.

Despite majority of the analytical models (i.e., logic utilization, routability, delay, energy) still focus on an island-style homogeneous FPGA architecture, Smith et al. derive a wirelength model that estimates the post-placement average wirelength of a heterogeneous FPGA architecture [3]. In essence, the homogeneous wirelength model is extended to accommodate the impact of heterogeneous embedded blocks such as the DSP and internal memory blocks. In the model, they take into account of placement constraints with respect to the fixed position embedded blocks, the dead blocks (i.e., the unused embedded blocks), and the additional embedded block pins. Even after these modifications, the heterogeneous FPGA wirelength model still suffers from *discrete effects*, since the model does not take specialized circuits into account.

4.3 Definitions and Problem Formulation

The communication backbone of a SoC platform is typically built based on the shared bus or crossbar switch. In this paper, we present our results with respect to the full crossbar switch based topology. However, as our model uses a bottom-up approach, it can be easily extended to the other topologies when necessary. Section 4.4 gives an elaborate description of our model development strategy.

We define the overall SoC architecture to consist of n η-bit processing nodes, fully connected by a n-by-n crossbar switch. The crossbar switch is composed of η 1-bit crossbar switches, with each crossbar handling a single lane on the bus.

Each 1-bit crossbar is composed of $(n.\eta)$ multiplexers. Since each node needs to communicate with the rest of the $n-1$ nodes, each multiplexer has a fan-in size of $n-1$.

Since the crossbar circuits are synthesized as multiplexers, we develop our model based on a generic multiplexer tree architecture which is composed of 2-to-1 multiplexers. Figure 4.1 depicts the multiplexer tree given that the total number of nodes is 8.

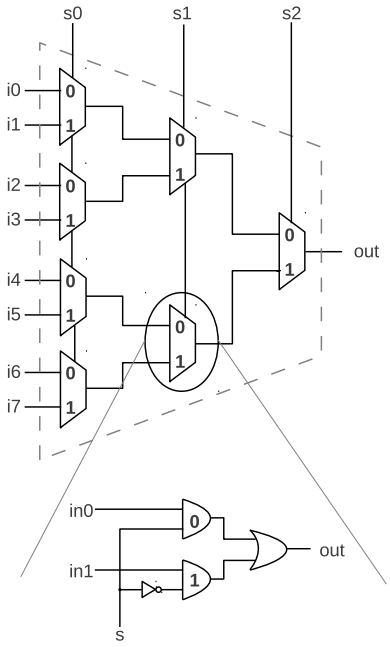
Then, given a K-input LUT, in order to estimate the logic utilization of the multiplexer tree, we formulate our approach as a pattern matching problem that approximates the minimum number of cuts based on a binary tree structure (i.e., multiplexer tree). Note that the cut enumeration step in the technology mapping stage addresses the problem in a grand scale which can be applied to multiple circuit genres. However, our main objective is to develop a model capable of identifying *discrete effects* using a combination of both analytical, and algorithmic approaches towards fast architecture exploration for crossbar switches.

Given the gate level multiplexer tree which is depicted as a boolean network, considering an arbitrary root node, v, Equation 4.1 summarises the cut enumeration step [16].

$$f\left(k,v\right) \quad = \quad \otimes_{u\in input(v)}^{k}\left[u+f\left(k,u\right)\right] \tag{4.1}$$

where u is the leaf (input) node and v is the root node, provided that there is an edge leading from u to v. The operators $\otimes^K$ and $+$ represent the boolean AND and OR respectively. Equation 4.1 enumerates all the K-feasible cuts that are rooted at v. As opposed to the original cut enumeration algorithm, given a LUT with K inputs, our model will produce one optimal K feasible cut by maximizing both node and depth coverages. We define node coverage as the number of nodes/gates that can be covered by the cut, and depth coverage as the distance/depth of the furthest leaf node, u, from the root node, v. Since the model takes both node and depth coverages into account, the model's optimization objectives are area and path delay.

Figure 4.1: An 8-to-1 multiplexer tree composed of 2-to-1 multiplexers.



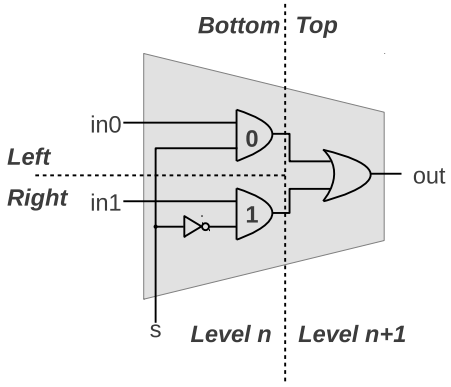
As opposed to cut enumeration, which approaches the problem at gate-level, our binary tree definition, as depicted in Figure 4.1, considers each node as a 2-to-1 multiplexer. Furthermore, by definition, each node in a binary tree can only have 2 inputs. However, each 2-to-1 multiplexer has 3 inputs. Hence, we modify the binary tree to accommodate the following two aberrations.

- Each node is partitioned into two, which we define as top and bottom. Then, the bottom partition is further divided into left and right corners. The bottom left partition consists of the left *AND* gate, while the bottom right partition consists of the *INVerter* and *AND* gates. The top level consists of the *OR* gate.

- Each bottom left or right partition will have an additional edge, which depicts the multiplexer selection, s.

Figure 4.2 depicts the multiplexer node after the partitioning.

Figure 4.2: A three partitioned 2-to-1 multiplexer node.



With the two additional modifications, the multiplexer tree depth is doubled. Hence, given a n-to-1 multiplexer tree, the new depth is denoted by $2.log_2 n$. Furthermore, when all partitions reside within the same cut, the number of inputs to each node is increased to 3. This is due to the additional selection input that is connected either to the bottom left or bottom right partition. The maximum number of inputs is 4, when each partition is considered separately in different cuts, because the selection input edge will need to be duplicated on both of the bottom partitions.

4.4 Model Development Approach

Our model operates based on the following two assumptions. Firstly, we assume that logic duplication will not have significant impact on our logic utilization estimation. Logic duplication is applied after placement stage by performing Basic Logic Element (BLE) moves in order to improve the delay performance of the netlist with a trade-off over area [63]. As this technique is applied during post-placement, the number of LUTs will be the same, as the moves do not consider merging multiple BLEs. Furthermore, given that the objective of logic duplication is to improve delay, this technique will mostly impact the logic blocks on the critical path leaving the majority of the blocks relatively unaffected. Secondly, our model does not take into account of resources required by the communication control logic, as we hy-

pothesize that the overhead will be insignificant when the size of crossbar switches scales larger.

4.4.1 Estimating Logic Utilization of Multiplexers

In Figure 4.3, analysing the binary tree, each cut can expand either horizontally, across multiple inputs, or vertically, with increasing depth coverage. Hence, in order to take into account of both node and depth coverages, firstly, we calculate an initial K-feasible cut using a *balanced* sub-tree. Here we define a *balanced* sub-tree to be a tree where all leaf nodes are equidistant from the root node. Then, since the number of inputs incurred during a depth increment is lesser, the remaining inputs is used to extend the cut by depth.

Figure 4.3: The partitioned 8-to-1 multiplexer tree

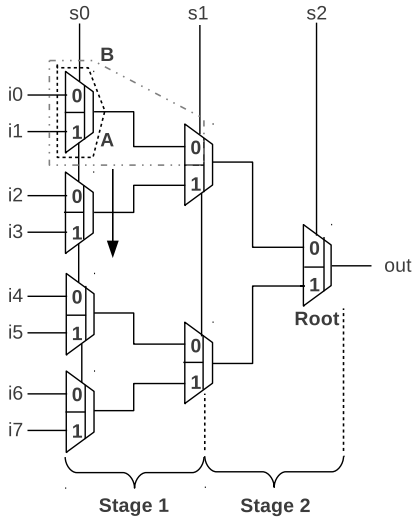


Figure 4.3 depicts two cuts assuming that the LUT size, K, is 4. Cut A is the initial cut that consist of the balance tree, and cut B includes the additional depth coverages. Eventually, cut B is the final selected cut. Since the 8-to-1 multiplexer tree is a *balanced* tree itself, multiple cuts similar to cut B can be replicated in stage 1 of the multiplexer tree. After all nodes in stage 1 are covered, the model progresses to stage 2 and calculates the number of cuts that cover all the nodes.

However, if the multiplexer tree is un-*balanced*, the left and right sub-trees with respect to the root node, will have different number of inputs. For example, in Figure 4.3, the multiplexer inputs allocated to the left and right sub-trees are denoted by $\{i0 \sim i3\}$ and $\{i4 \sim i7\}$ respectively. Hence, both sub-trees need to be handled separately in order to cater to a wider range of multiplexer sizes. Although the model is able to handle various multiplexer input sizes, since we allow the user to determine the fan-in distribution of both sub-trees, it is important to advocate an even distribution of fan-ins as far as possible. This is because, an even distribution will ensure an optimized multiplexer depth, and a fewer number of cuts.

Algorithm 1 estimates the number of logic slices of a multiplexer tree with respect to K, and the number of fan-ins allocated to the left sub-tree, $left\_in$, and right sub-tree, $right\_in$ respectively. The algorithm is separated into two portions. The first portion (i.e., Steps 1, 2, and 3), calculates the maximum coverage of a cut. Then, in the second portion (i.e., Steps 4 and 5), populates the cut across the entire multiplexer with respect to the left and right sub-trees.

In step 1, the model assumes that the multiplexer is un-*balanced*. The multiplexer depths of the left and right sub-trees are denoted by the variables mux_left_depth and mux_right_depth respectively. Hence, the depth of the multiplexer, as denoted by *mux_depth*, is determined by the larger sub-tree, and *small_depth* denotes the depth of the smaller sub-tree.

In step 2, the initial cut, as depicted by cut A of Figure 4.3, is calculated. Since the initial cut is always a *balanced* tree, the size of the cut can be calculated with respect to the tree depth. The depth is calculated by deducting the number of fan-ins incurred by the multiplexer inputs and selection bits required on the next level, $depth + 1$, where the number of fan-ins is denoted as,

$$\begin{aligned} fanin &= f(select\_bits) \\ &= select\_bits + 2^{select\_bits} \end{aligned} \tag{4.2}$$

In step 3, the depth of the final cut after adding the additional depth, as depicted

Algorithm 1 Multiplexer logic utilization estimation tool.

Algorithm 4.4.1: MUX_SLICE_COUNT($INPUTS$)

$INPUTS : K, left\_in, right\_in$

comment: Step 1

$big\_depth = max\,(log_2 left\_in, log_2 right\_in)$

$mux\_depth = 2.\,(big\_depth + 1)$

$small\_depth = min\,(log_2 left\_in, log_2 right\_in)$

$small\_depth = 2.small\_depth$

comment: Step 2

while $K - (depth + 1) - 2^{depth+1} >= 0$

$\Big\{ depth = depth + 1$

$depth = 2.depth$

comment: Step 3

if $depth > mux\_depth$

$\Big\{ depth = mux\_depth$

else

$\Big\{ add\_depth = K - \frac{depth}{2} - 2^{\frac{depth}{2}}$

if $add\_depth > mux\_depth + depth$

$\Big\{ add\_depth = mux\_depth - depth$

$extend\_depth = depth + add\_depth$

comment: Step 4

$get\_slice\_count\,(big, mux\_depth, extend\_depth, depth, slice\_count)$

comment: Step 5

$get\_slice\_count\,(small, small\_depth, extend\_depth, depth, slice\_count)$

return $(s)lice\_count$

Algorithm 2 Calculate number of slices in each sub-tree.

Algorithm 4.4.2: GET_SLICE_COUNT($INPUTS$)

$INPUTS :< big|small >, mux\_depth, extend\_depth, balance\_depth, slice\_count$
while $mux\_depth - level.extend\_depth - balance\_depth >= 0$
$\begin{cases} matching\_depth\_subtree = \\ ceiling\left(\frac{1}{2}.(mux\_depth - level.extend\_depth - balance\_depth)\right) \\ add\_slice\_count = 2^{matching\_depth\_subtree} \\ \textbf{if } (subtree\ is\ big)\ and\ (add\_slice\_count > 1) \\ \left\{ add\_slice\_count = \frac{add\_slice\_count}{2} \right. \\ slice\_count = slice\_count + add\_slice\_count \\ level = level + 1 \end{cases}$

by cut B in Figure 4.3, is calculated. Since the increase in fan-in with each depth increment is always one [1] [2], the cost of extending the cut vertically (i.e., depth coverage) is smaller than the cost of extending the cut horizontally (i.e., breadth coverage). Therefore, the vertical extensions in step 3 act as a fine-tuning step to increase the overall cut coverage.

In steps 4 and 5, the number of cuts incurred by the multiplexer is calculated by populating cut B across the large (i.e. with larger number of fan-ins) and small (i.e. with smaller number of fan-ins) sub-trees separately. The population process is carried out by calculating the logic utilization of individual *stages* as depicted in Figure 4.3 progressing from the multiplexer fan-ins to the fan-out as denoted by Equations 4.3 to 4.5.

Equation 4.3 denotes the number of LUTs utilized by a single cut with *depth* denoting the depth of the cut.

[1] If the next partition is a bottom partition, the number of fan-ins increase due to the inputs of the *AND* gate. However, one fan-in will be absorbed due to the edge inclusion between the preceding and successive partition.

[2] If the next partition is a top partition, the number of fan-ins increase is always one.

$$slice\_count \;=\; 2^{depth} \tag{4.3}$$

Equations 4.4 and 4.5 denote the depth of cut B in the big and small sub-trees of a single *stage*. The variables, *mux_depth*, *level*, *extend_depth* and *balance_depth* refer to the overall multiplexer depth, the current *stage*, the additional depth of cut B, and the initial depth of cut A respectively.

$$\begin{aligned} cut\_depth_{big\_subtree} &= \frac{1}{2}.mux\_depth \\ &- \frac{1}{2}.level.extend\_depth \\ &- \frac{1}{2}.balance\_depth \end{aligned} \tag{4.4}$$

$$\begin{aligned} cut\_depth_{small\_subtree} &= \frac{1}{2}.small\_depth \\ &- \frac{1}{2}.level.extend\_depth \\ &- \frac{1}{2}.balance\_depth \end{aligned} \tag{4.5}$$

The number of LUTs utilized by cut B in the big and small sub-trees is calculated by substituting the term, *depth*, in Equation 4.3 with Equations 4.4 and 4.5 respectively.

After calculating the depth in a single *stage*, the number of LUTs for subsequent *stages* are calculated by iterating the number of *stages* that is required to converge to the multiplexer tree fan-out. After calculating the total number of LUT slices needed for a single multiplexer, we then proceed to estimate the total logic utilization of a full crossbar switch.

4.4.2 Estimating Logic Utilization of Crossbar Switches

Given a SoC with n nodes, the number of LUT slices required for a 1-Bit full-crossbar switch is denoted by Equation 4.6, where k indicates the number of inputs of a LUT.

$$slice\_count_{[1]} = GET\_SLICE\_COUNT\,(k, n)\,.n \tag{4.6}$$

Then, extending it to a wide bus with an arbitrary data width of η, Equation 4.7 depicts the final logic utilization function for a full crossbar switch.

$$slice\_count_{[1..\eta]} = \frac{1}{2}.GET\_SLICE\_COUNT\,(k, n)\,. \\ (1 + \eta) \tag{4.7}$$

4.5 Evaluation Methodology

The quality of the model is verified by quantifying both the correlation and the percentage deviation between the model results and the empirical data. We make use of the Mean Absolute Percentage Error (MAPE) and Correlation Ratio (c-ratio) metrics to evaluate our model.

Mean Absolute Percentage Error (MAPE) is defined by Equation 4.8

$$MAPE = \frac{1}{C} \sum_{\forall Circuits} \frac{|Predicted - Measured|}{Measured} \tag{4.8}$$

where the *Predicted* and *Measured* refers to the model and experimental output respectively.

Correlation Ratio (c-ratio) is defined by Equation 4.9

$$c - ratio = \frac{Number\ of\ non - violating\ pairs}{Total\ number of pairs} \tag{4.9}$$

where a *pair* denotes a pair of data points containing experimental output and its corresponding model output. A *pair* is defined as *non-violating* when the model output resonates with the experimental output. Since we emphasize the detection of *discrete effects*, there are possibly three different trends (i.e., increasing, decreasing,

level constant) we can observe. For example, an increase in experimental value for a data point with respect to the previous data point should show an increase in the modelled value or vice-versa.

4.5.1 Crossbar Switch Test Benches

We make use of the crossbar generation scripts created by Das et al. with different number of ports and data width [17]. Table 4.1 depicts the ranges for the port and data width that consolidate the permutation of various crossbar switches generated during our evaluation.

We make use of the Quartus II Web Edition to synthesize the crossbar circuits. The output '*blif*' netlist is fed into the FlowMap [11] technology mapper to calculate the total required number of LUT slices for the LUT size, K which ranges from 4 till 12.

4.5.2 Lam's Area Equations and Rent's Parameter Generation

We compare our model with Lam's area model, which is a Rent's rule based analytical model [18]. Equation 4.10 depicts the logic utilization from Lam's model.

$$slice\_count, n_k \quad = \quad n_2 . \sqrt[p]{\frac{3}{K+1-\gamma}} \tag{4.10}$$

where n_2, γ, and p denote the number of 2-input gates that are required to implement the circuit, number of unused inputs in the LUT, and Rent's parameter

Table 4.1: Crossbar switches benchmark circuits

Ports	Data Width
4-16	1, 2, 4, 6, 8, 10, 12
20	1, 2, 4, 6, 8, 10, 12
24	1, 2, 4, 6, 8, 10, 12
32	1, 2, 4, 6, 8, 10, 12
40	1, 2, 4, 6, 8, 10, 12
48	1, 2, 4, 6, 8, 10, 12

respectively. Note that, as opposed to our model which does not have any CAD tool dependencies, Lam's model will require an extra synthesis step to generate the n_2 parameter which might become a process overhead when rapid evaluation is required. Then, we make use of the relationship, $\gamma = \frac{K}{4} - \frac{1}{2}$, that is denoted in [18] to approximate the value of γ for various K values.

We make use of Rent's rule to solve for the value of Rent's parameter, p, for different crossbar configurations. Each gate in the multiplexer tree has 2 inputs and 1 output. Hence, the number of external inputs/outputs can be approximated by the Rent's rule, as denoted by Equation 4.11.

$$y = 3.x^p \tag{4.11}$$

Given a full crossbar switch with data width, η, supporting n communicating nodes, x denotes the number of 2-input gates that are required. Given the multiplexer tree architecture in Figure 4.1, the number of gates required is depicted in Equation 4.12.

$$x = 4.(\eta + 1).(n + 1) \tag{4.12}$$

Similarly, we can calculate the required number of external inputs/outputs based on the proposed multiplexer tree architecture. Equation 4.13 depicts the total number of external inputs/outputs and selection lines required by the same crossbar switch.

$$y = (\eta + 1).2n + \log_2 n \tag{4.13}$$

Hence, solving Equations 4.11 and 4.13 for the Rent's parameter, p, we derive Equation 4.14.

$$p = \log_x \left(\frac{(\eta + 1).2n + \log_2 n}{3} \right) \tag{4.14}$$

Upon substituting x with Equation 4.12, Equation 4.14 is the final expression used to generate the Rent's parameter, p, which is used in Lam's area model for the various permutations of crossbar switch circuits.

4.6 Model Validation

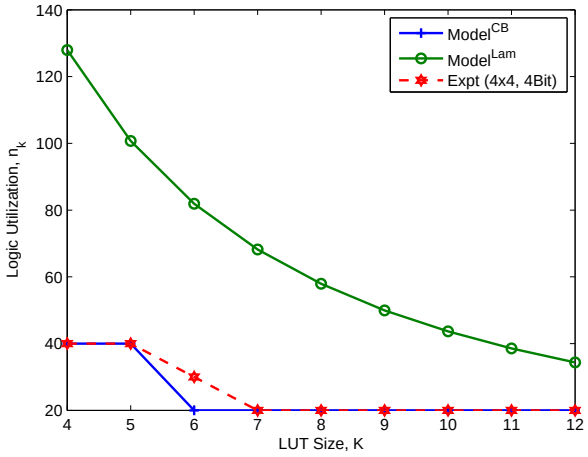
To compare the empirical results for each crossbar circuit, we sweep the LUT size, K from 4 to 12. Figure 4 depicts the results of crossbar circuits with different combinations of number of ports (4, 8, 12, 16, 24) and data width (4, 12). In the figure, we show the trends with respect to the changes in LUT size for our model ($Model^{CB}$), Lam's model ($Model^{Lam}$), and the empirical data.

In general, we can observe that $Model^{Lam}$ has a tendency to overestimate the logic utilization for all crossbar circuits. Lam's model assumes a *generic* circuit. The Rent's parameter, p, denotes the complexities of the circuits. However, despite the model's dependency on p, it is unable to capture the detailed circuit characteristics. Hence, the model will miss optimization opportunities for a more efficient LUT packing strategy with regular circuits like the crossbar switch.

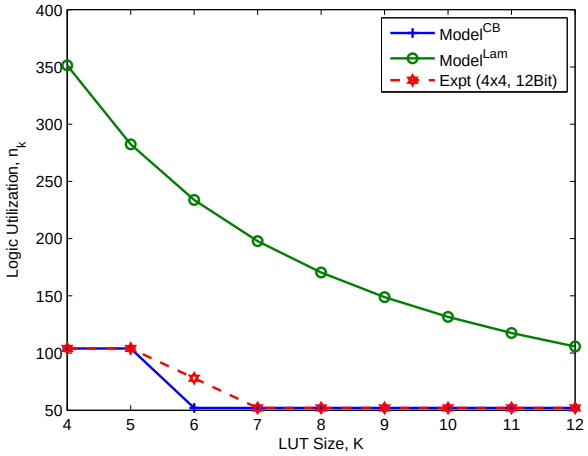
On the contrary, $Model^{CB}$ achieves a much closer estimation of the actual empirical data. In contrast to the smooth curve, which is typical of all other analytical models, $Model^{CB}$ has rapid changes in the line. These are the attempts of our model for capturing the *discrete effects* of the circuits.

4.6.1 Capturing The Discrete Effects

In Figure 4.4(a), we can observe that the empirical trend is level constant as K increases from 4 to 5, and only starts decreasing when K increases from 5 to 7. Thereafter, it is again level constant from 8 till 12. The first level constant region (i.e., 4 to 5) is due to the fact that the technology mapper is unable to increase the LUT packing density by increasing the LUT size from 4 to 5. This is because, according to the multiplexer tree architecture, the depth of the multiplexers in the 4x4 crossbar switch is only 2. Hence, increasing the LUT size from 4 to 5 will only

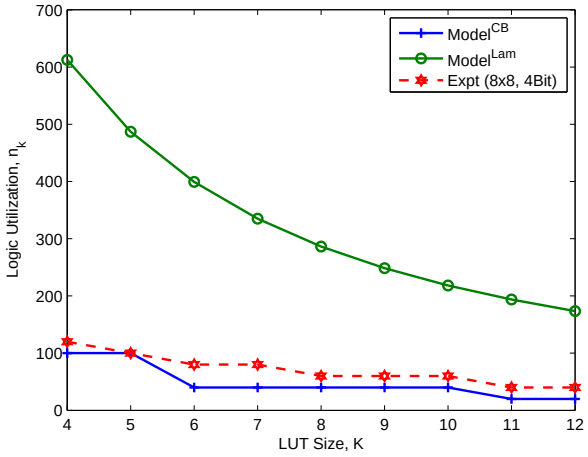


(a) 4-Bit data bus width

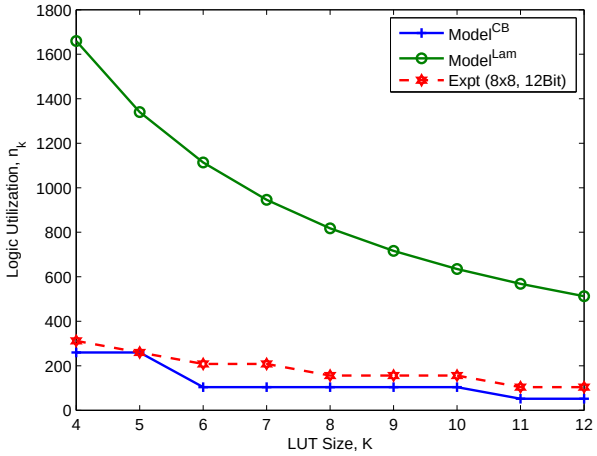


(b) 12-Bit data bus width

Figure 4.4: Model validation of 4x4 crossbar configurations

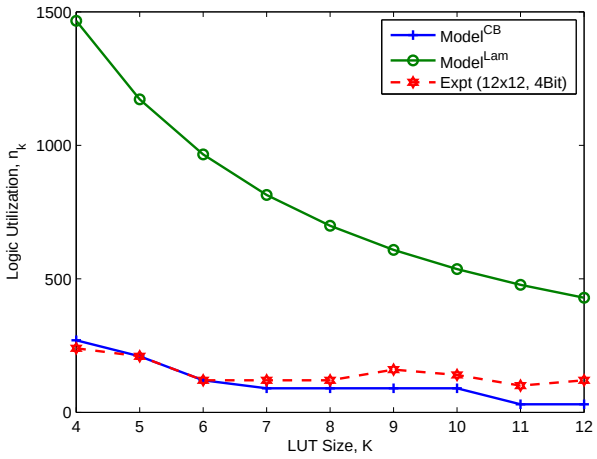


(a) 4-Bit data bus width

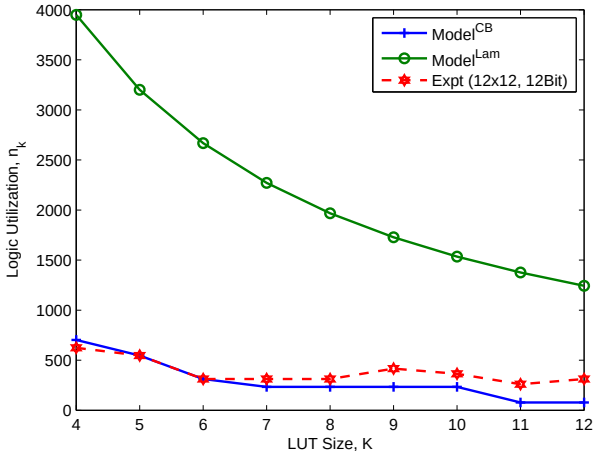


(b) 12-Bit data bus width

Figure 4.5: Model validation of 8x8 crossbar configurations

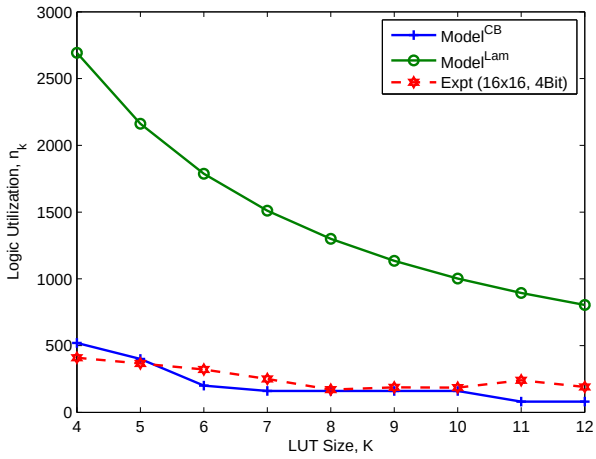


(a) 4-Bit data bus width

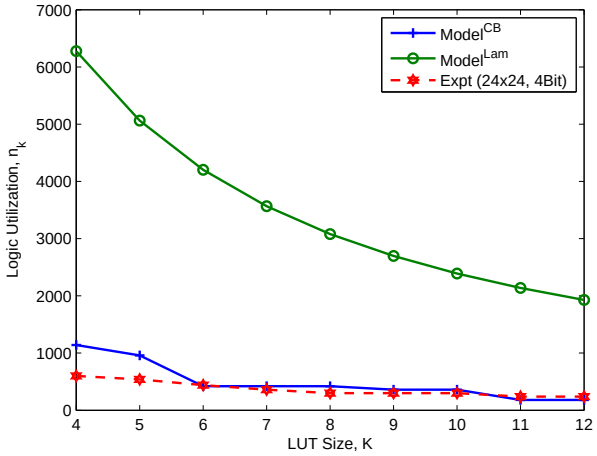


(b) 12-Bit data bus width

Figure 4.6: Model validation of 12x12 crossbar configurations



(a) 4-Bit data bus width



(b) 4-Bit data bus width

Figure 4.7: Model validation of 16x16 and 24x24 crossbar configurations

Table 4.2: Summary of model's experimental performance.

Config	$c - \hat{r}atio$	$M\hat{A}PE$
4x4 (4-Bit)	0.89	0.04
4x4 (12-Bit)	0.89	0.04
8x8 (4-Bit)	0.78	0.35
8x8 (12-Bit)	0.78	0.35
12x12 (4-Bit)	0.56	0.32
12x12 (12-Bit)	0.56	0.32
16x16 (4-Bit)	0.44	0.30
24x24 (12-Bit)	0.67	0.35

help when there are more than 2 levels of 2-to-1 multiplexers. $Model^{CB}$ is able to capture this point and reflect a flat line. The second level constant region (i.e., 8 till 12), implies that the entire multiplexer is able to fit into each LUT, and increasing the LUT size further does not cause logic utilization to improve. However, in the region, when K is from 6 to 7, the empirical data indicates a decreasing trend while $Model^{CB}$ indicates a level constant. This is due to the fact that the crossbar generation script will produce an extra selection line. This extra input signal causes $Model^{CB}$ to deviate from the empirical results.

In general, $Model^{CB}$ is able to capture the *discrete effects*. However, in Figures 4.6(b) and 4.7(a), the empirical trend increases when K increases from 8 to 9 and 10 to 11 respectively. Hence, $Model^{CB}$ is not able to capture indeterministic behaviours that inherently exists in CAD tools [45].

4.6.2 Scaling The Data Width

We analyse the effects of data width by comparing $Model^{CB}$'s results with the empirical trend by increasing the crossbar switches' data width. By comparing the Figures 4.4(a) and 4.4(b), 4.5(a) and 4.5(b), 4.6(a) and 4.6(b), we conclude that the $Model^{CB}$ scales very well between the two differing data widths. Furthermore, the

same $M\hat{A}PE$, and $c - \hat{r}atio$ values are reported for the two different data widths. This implies that the model correlates linearly with data width scaling. This linear relationship is captured by Equation 4.7.

4.6.3 Scaling The Number of Ports

Our $Model^{CB}$ allows deriving a hypothetical cut that has the best depth and area performance. This is in contrast to the technology mapper, that exhaustively enumerates all possible cuts before selecting an optimal cut based on a pre-defined cost function. As the number of ports increase, the number of enumerations to pack a LUT will increase. Hence, the model will inevitably exclude higher order optimization opportunities such as global and local duplication cost adjustments [16]. In Figure 4.7(b) we show the results for the largest crossbar switch circuit (24x24) that is evaluated in this work. $Model^{CB}$ shows a correlation of 0.67, and an absolute error of 0.35. However, given that the model is mainly deployed for fast architecture exploration to conduct first order approximations, we argue that $Model^{CB}$ is still relatively accurate and robust. $Model^{CB}$ consistently results with a much closer correlation with respect to the empirical data when compared to $Model^{Lam}$.

4.7 Summary

The age of FPGAs is moving into the realm with multiple functional embedded hard blocks, which supports computational intensive SoC to be implemented on these platforms. The complexity of these platforms requires greater innovations on simple but efficient design tools in order to shorten the development cycles. Our paper addresses the inherent problem of *discrete effects* in FPGA analytical models, which hinders the use of FPGA analytical models for crossbar switches.

This chapter presents a hybrid model which makes use of a combination of algorithmic approach and analytical modelling techniques to estimate the number of LUT slices utilized in full crossbar switches. This model can be extended to evaluate the logic utilization in SoC platforms, which require these crossbar switches

as means of communications. Our results show that the proposed model is able to produce more accurate estimations when compared with existing analytical models. Our future work will include looking into other FPGA analytical models to assist in the development of FPGA based SoC platforms.

CHAPTER 5

Conclusion

5.1 Summary and Contributions

FPGAs are an interesting genre of Very Large Integrated Circuits (VLSI) that require a mixture of programmable metal dominated routing interconnects and silicon based logic blocks/hard-IPs in order to support post-fabrication reconfiguration. This unique combination of optimizing both the soft-logic/accelerators and routing interconnect architectures has made FPGA architecture development a challenging and exquisite task to tackle. Furthermore, due to the inherent reconfigurable nature, a new architecture cannot be evaluated unless benchmark circuits are synthesized and implemented based on the new architecture. In today's context, industry and academia approaches to design a new FPGA device always require a strong reliance on CAD tools. This entanglement with CAD tools to complement architecture development is a burdensome process. Furthermore, the huge architecture design space has make searching for architectural optimal pareto configurations (i.e., more than 1,000,000 configurations) near impossible!

This dissertation is in the area of FPGA analytical/empirical based architecture models. Example applications of these models are in areas of (1) expediting early architecture design space exploration, (2) evaluating application/algorithm performance, specific to a FPGA architecture and (3) complementing FPGA Computer-Aided Design (CAD) tools. The major contributions of these models are to facilitate *fast* first-hand evaluation of their design without launching extensive time-consuming CAD flows.

- Chapter 2 proposes a static power model targeting homogeneous FPGA architectures that is composed of a device and an architecture layer to estimate

both the active and idle leakage power in FPGA devices. Taking into account of both logic and routing architectures, the model first estimates the total number of minimum width transistors and then performs a first-order sub-threshold power estimation. Despite being applied towards the 180nm process technology, the model is capable of being extended to smaller technology feature sizes by merely modifying the device abstraction layer to support additional leakage power components.

- Chapter 3 proposes a post-routing wirelength model targeting homogeneous FPGA architectures. Unlike other previous wirelength model work [3], this work takes into account of both logic and routing architecture parameters. Two variations of the model, Model^F and Model^D, are derived with Model^D achieving the best trend correlation. A case study is performed to demonstrate the model's usage of deriving an optimal output connection-box flexibility, F_{Cout}, based on the trade-off between area and delay. The case study leads to a lower $F_{Cout} = 0.07$, than the default $F_{Cout} = 0.1$ in the latest VTR-7.0 CAD tool. This observation leads to an increase in manufacturing yield of 9 dies/wafer with a slight improvement in system frequency. Moreover, utilizing a model-based approach results in a reduction of 53.4% in architecture design space exploration time.

- Chapter 4 proposes an *algorithmic* approach to estimate the logic density (i.e., number of LUTs) of multiplexer-based circuits. This approach addresses the problem of *discrete effects*[19] in FPGA analytical models. In a corresponding CAD tool flow, the technology mapper is responsible for accounting for partitioning the logic among LUTs, thereby accounting for the number of LUTs. Hence, using the model to generate logic density of a fundamental circuit element, such as a multiplexer, can be used in later stages of modeling to estimate later CAD stages performance metrics, such as critical path delay [28] or even power [76].

5.2 Future Work

There are some potential research directions as a result of the work described in preceding chapters. We enumerate them based on the flow of the chapters from the dissertation. Then, possible extensions of our models into heterogeneous FPGA architectures are discussed.

5.2.1 Potential Model Extensions

The static power model over-estimates with respect to the routing architecture parameters. The model's approach of estimating the amount of routing resources is based on the number of used logic cluster in the FPGA which is rationalized by a normalization factor which is based on the circuit required channel width from Fang's channel width model. This approach has a tendency to over-estimate as the channel width model is estimating the minimum required channel width that is needed to route the circuit. Hence, a potential future work will be to use a post-routing wirelength which takes into account of the FPGA device's channel width (i.e., typically, it will have 30% more routing tracks than the minimum circuit required channel width). The post-routing wirelength in this dissertation is the first post-routing wirelength model that can be deployed for this purpose.

The post-routing wirelength model is able to capture the average amount of routing resources that are used in a single net. Moreover, due to the programmable nature of FPGA, the routing interconnect fabric inherently becomes a major contributor of important performance metrics, such as the energy [55][76], critical path delay [28] and area [18] of a device. Despite so, all these models do not take into account of post-routing wirelength information. The post-routing wirelength can be integrated to enhance the accuracy of these models.

5.2.2 Extension to a Heterogeneous FPGA Architecture

Today's FPGA architecture is heterogeneous and consist of both the soft logic Configurable Logic Blocks (CLBs) and ASIC hardIPs. Extending the post-routing wire-

length models to a heterogeneous wirelength model require capturing the properties of hardIP blocks. As compared to a CLB, an ASIC hardIP usually has a higher pin-ins/outs with a lower block distribution. Smith et al. model the post-placement heterogeneous wirelength model by taking into account of these additional features which contributes to the overall wirelength as additional terms in the form of overhead. Therefore, for the first step, the post-routing wirelength can be modified by including the additional terms which models the overhead.

However, Smith's heterogeneous wirelength model does not capture the area variations between inter hardIPs and CLBs. This is important as this irregularity in block distribution will affect the spatial locality between individual blocks [21]. Specifically, the physical routing distance with respect to inter-CLB, inter-hardIP and CLB-to-hardIP is subjected to the area, distribution (i.e., assuming a ASMBL architecture) and fan-out of a hardIP. This variation increases with the varieties of hardIPs that are supported. Hence, the trade-off between hardIP functionality/optimization (i.e., direct implication on hardIP size) and logic block granularity becomes a critical design decision that is application driven. Therefore, for the second step, the post-routing wirelength can be modeled to take into account of area variations which can also be modeled as additional terms which denotes extra overhead on the overall wirelength.

The static power model requires to capture the leakage power contributed by both the logic and routing resources. Hence, extending the static power model for a heterogeneous FPGA architecture will require the heterogeneous post-routing wirelength model to capture the number of routing resources that are utilized. On the other hand, the logic resources are now divided into the soft logic, CLBs and hardIPs now. The number of used logic block clusters can be derived using the Rent's rule which allows the static power derivation for soft logic blocks to be relatively straight-forward. However, the hardIPs which are in the form of ASIC require ASIC synthesis tools (e.g., Synopsys Design Compiler) which can extract the static power. This is based on the assumption that the FPGA architects will have access to the utilized IP-cores that implements these ASIC hardIPs.

5.3 Concluding Remarks

In summary, we believe that the complexities of today's FPGA architectures have grown to an extent where using CAD tools to perform architecture design space sweeping has become exceedingly time consuming and tedious. To aggravate the situation, the inter-dependencies between architecture and CAD tool development are an impetus to designing domain-specific reconfigurable hardware. At this point, introducing models as enablers is critical to not only expedite the development time but also facilitate the development process based on model-guided architecture estimations.

REFERENCES

[1] A. Gayasan, M. K., V. Narayanan and A. Rahman (2008). Designing a 3-D FPGA: Switch Box Architecture and Thermal Issues. *IEEE Transactions on VLSI*, **16**(7), pp. 882–893.

[2] A. Kesharvarzi, C. F. H., K. Roy (1997). Intrinsic Leakage in Low Power Deep Submicron CMOS ICs. *Proceedings of the IEEE International Test Conference*, pp. 146–155.

[3] A. M. Smith, J. D. and S. J. E. Wilton (2009). Wirelength modeling for homogeneous and heterogeneous FPGA architectural development. *In Proceedings FPGA*, pp. 181–191.

[4] A. Ye, J. R. (2006). Using Bus-Based Connections to Improve Field-Programmable Gate-Array Density for Implementing Datapath Circuits. *IEEE Transactions on VLSI*, **14**(5), pp. 462–473.

[5] Ahmed, E. and J. Rose (2004). The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Delay. *IEEE Transactions on VLSI*, **12**(3), pp. 288–298.

[6] Allenby, G. M. (2007). Cross-validation, the bayes theorem, and small-sample bias. *Journal of Business and Economic Statistics*, **8**(2), pp. 171–178.

[7] Betz, V. and J. Rose (1997). VPR: A new packing, placement and routing tool for FPGA research. *In Proceedings of the 7th International Workshop on Field-Programmable Logic and Applications*, pp. 213–222.

[8] Burch, R., F. Najm, P. Yang, and T. Trick (1992). McPOWER: A Monte Carlo approach to power estimation. *IEEE/ACM International Conference on Computer-Aided Design*, pp. 90–97.

[9] C. Ababei, H. M. and K. Bazargan (2006). Three-Dimensional Place and Route for FPGAs. *IEEE Transactions on CAD of Integrated Circuits and Systems*, **25**(6), pp. 1132–1140.

[10] C. Dong, S. C. and D. Chen (2009). Variation Aware Routing for Three-Dimensional FPGAs. *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 298–303.

[11] Cong, J. and Y. Ding (1994). FlowMap: An Optimal Technology Mapping Algorithm for Delay Optimization in Lookup-Table based FPGA Designs. *IEEE Transactions on CAD*, **13**(1), pp. 1–12.

[12] Cong, J. and K. Minkovich (2007). Optimality study of logic synthesis for LUT-based FPGAs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **26**, pp. 230–239.

[13] Cong, J., J. Peck, and Y. Ding (1996). RASP: A General Logic Synthesis System for SRAM-based FPGAs. *In Proceedings of the 1996 ACM fourth international symposium on FPGAs*, pp. 137–143.

[14] Corporation, A. (2007). FPGA Performance Benchmarking Methodology. Altera White Paper.

[15] Crit, M. A. (1987). Estimating dynamic power consumption of CMOS circuits. *IEEE International Conference on Computer-Aided Design*, pp. 534–537.

[16] D. Chen, C. D. L. H. F. L., J. Cong and C. C. Peng (2010). Technology Mapping and Clustering for FPGA Architectures with Dual Supply Voltages. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **29**(11), pp. 272–277.

[17] Das, J. (2010). Code for our analytical models. University of British Columbia.

[18] Das, J., A. Lam, S. J. E. Wilton, P. Leong, and W. Luk (2011). An Analytical Model Relating FPGA Architecture to Logic Density and Depth. *IEEE Transactions on Very Large Integrated (VLSI) Systems*, **19**(12), pp. 2229–2242.

[19] Das, J. and S. J. E. Wilton (2011). Accelerated FPGA Architecture Design: Capabilities and Limitations of Analytical Models. *International Conference on Field-Programmable Technology (FPT)*, pp. 1–8.

[20] Das, J. and S. J. E. Wilton (2011). An analytical model relating FPGA architecture parameters to routability. *In Proceedings of the 19th ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 181–184.

[21] Dehon, A. (2013). Location, Location, Location - The Role of Spatial Locality in Asymptotic Energy Minimization. *Proceedings of the FPGAs*.

[22] Donath, W. (1979). Placement and Average Interconnection Lengths of Computer Logic. *IEEE Transactions on Circuits and Systems*, **26**(4), pp. 272–277.

[23] Fang, W. M. and J. Rose (2008). Modeling routing demand for early-stage FPGA architecture development. *In Proc, FPGA08*, p. 139.

[24] Feuer, M. (1982). Connectivity of random logic. *IEEE Transactions on Computers*, pp. 29–33.

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

[26] Ghosh, A., S. Devadas, K. Keutzer, and J. White (1992). Estimation of average switching activity in combinational and sequential circuits. *29th ACM/IEEE Design Automation Conference*, pp. 253–259.

[27] H. Nikolov, T. S. and E. Deprettere (2008). Systematic and Automated Multiprocessor System Design Programming and Implementation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **27**(3), pp. 542–555.

[28] Hung, E., H. Yu, T. Chau, P. H. W. Leong, and S. J. E. Wilton (2009). A Detailed Delay Path Model for FPGAs. *In Proceedings Field-Programmable Technology*, pp. 96–103.

[29] Hussein, J., M. Klein, and M. Hart (2011). Lowering Power at 28 nm with Xilinx 7 Series FPGAs. White Paper WP389, Xilinx Inc.

[30] Hwang, F. K. (1976). On steiner minimal trees with rectilinear distance. *SIAM Journal on Applied Mathematics*, **30**(1), pp. 105–114.

[31] J. A. Davis, V. K. D. and J. D. Meindl (1998). A stochastic wire-length distribution for gigascale integration (GSI). Part I. Derivation and validation. *IEEE Transactions on Electron Devices*, **45**(3), pp. 580–589.

[32] J. A. Davis, V. K. D. and J. D. Meindl (1998). A stochastic wire-length distribution for gigascale integration (GSI). Part II. Applications to clock frequency, power dissipation, and chip size estimation. *IEEE Transactions on Electron Devices*, **45**(3), pp. 590–597.

[33] J. Luu, J. H. A. and J. Rose (2011). Architecture description and packing for logic blocks with hierarchy, modes and complex interconnect. *In Proc, FPGA11*, pp. 227–236.

[34] J. S. Swartz, V. B. and J. Rose (1998). A Fast Routability-Driven Router for FPGAs. *In Proc. FPGA'98*, pp. 140–149.

[35] K. C. Nunna, F. M. and K. Murakami (2012). Thermal-aware Partitioning for 3D FPGAs. *International Conference on Field Programmable Logic and Applications (FPL)*, pp. 475–476.

[36] K. E. Murray, S. L. J. L., S. Whitty and V. Betz (2013). TITAN: Enabling Large and Complex Benchmarks In Academia CAD. *International Conference on Field Programmable Logic and Applications, FPL'13*.

[37] K. Siozios, V. F. P., K. Sotiriadis and D. Soudris (2007). A Software-Supported Methodology for Designing High-Performance 3D FPGA Architectures. *IFIP International Conference on VLSI-SoC*, pp. 54–59.

[38] Kannan, P. and D. Bhatia (2004). Estimating Pre-Placement FPGA Interconnection Requirements. *In Proceedings International Conference on VLSI Design*, pp. 869–874.

[39] Kannan, P. and D. Bhatia (2006). Interconnect Estimation for FPGAs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **25**(8), pp. 1523–1534.

[40] Kuon, I. and J. Rose (2006). Measuring the Gap Between FPGAs And ASICs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits*, **26**(2), pp. 203–215.

[41] Lamoureux, J. and S. J. Wilton (2003). On the Interaction Between Power-Aware FPGA CAD Algorithms. *In Proceedings of the 2003 IEEE/ACM international Con- ference on Computer-Aided Design*.

[42] Landman, B. and R. Russo (1971). On a pin vs block relationship for partitions of logic graphs. *IEEE Transactions on Computers*, **C-20**, pp. 1469–1479.

[43] Lemieux, G., E. Lee, M. Tom, and A. Yu (2004). Directional and Single-Driver Wires in FPGA Interconnect. *IEEE International Conference on Field-Programmable Technology*, pp. 41–48.

[44] Li, F., D. Chen, L. He, and J. Cong (2003). Architecture evaluation for power-efficient FPGAs. *In Proceedings of the 2003 ACM/SIGDA 11th International Symposium on FPGAs*, pp. 175–184.

[45] Luu, J., I. Kuon, P. Jamieson, T. Campbell, W. M. F. A. Ye, K. Kent, and J. Rose (2009). VPR 5.0: FPGA CAD and architecture exploration tools with single-driver routing, heterogeneity and process scaling. *In Proceedings FPGAs*, pp. 133–142.

[46] Mark, C. (2008). A System-Level Synthetic Circuit Generator for FPGA Architectural Analysis. University of British Columbia. Masters Thesis.

[47] Marquardt, A., V. Betz, and J. Rose (1999). Using cluster-based logic blocks and timing-driven packing to improve FPGA speed and density. *In Proceedings of the 1999 ACM/SIGDA 7th International Symposium on FPGAs*, pp. 37–46.

[48] Najm, F., R. Burch, P. Yang, and I. Hajj (1988). CREST a current estimator for CMOS circuit. *IEEE International Conference on Computer-Aided Design*, pp. 204–207.

[49] Najm, F. N. (1993). Transition density: a new measure of activity in digital circuits. *IEEE Transactions on Computer-Aided Design*, **12**(2), pp. 310–323.

[50] Najm, F. N. (1994). A Survey of Power Estimation Techniques in VLSI Circuits. *IEEE Transactions on VLSI Systems*, **2**(4), pp. 446–455.

[51] P. Zarkesh-Ha, W. L., J. A. Davis and J. D. Meindl (2000). Prediction of interconnect fan-out distribution using rent's rule. *In System-Level Interconnect Prediction, SLIP*, pp. 107–112.

[52] Pi, T. and P. J. Crotty (2004). FPGA Lookup Table with Transmission Gate Structure for Reliable Low-Voltage Operation. *U.S. Patent 6 809 552*.

[53] Pistorius, J. and M. Hutton (2000). Placement rent exponent calculation methods, temporal behaviour and FPGA architecture evaluation. *In Proc. SLIP'03*, pp. 31–38.

[54] Poon, K., A. Yan, and S. Wilton (2002). A flexible Power Model for FPGAs. *In Proceedings of 12th International Conference on Field-Programmable Logic and Applications*, pp. 312–321.

[55] Rajavel, S. T. and A. Akoglu (2011). An Analytical Energy Model to accelerate FPGA Logic Architecture Investigation. *2011 International Conference on Field-Programmable Technology*.

[56] RCL (2010). Reconfigurable Computing Lab website, wirelength modeling tool.

[57] Rose, J., J. Luu, C. W. Yu, O. Densmore, J. Goeders, A. Somerville, K. B. Kent, P. Jamieson, and J. Anderson (2012). The VTR Project: architecture and CAD for FPGAs from verilog to routing. *In Proceedings of the ACM/SIGDA international symposium on FPGAs*, pp. 77–86.

[58] Roy, K., S. Mukhopadhyay, and H. Mahmoodi-Meimand (2003). Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits. *Proceedings of the IEEE*, pp. 305–327.

[59] S. Cho, A. M., S. Chatterjee and R. Brayton (2007). Efficient FPGA mapping using priority cuts. *In Proceedings FPGA*.

[60] S. D. Brown, J. R. and Z. G. Vranesic (1993). A Stochastic Model to Predict the Routability of Field-Programmable Gate Arrays. *IEEE Transactions On CAD of Integrated Circuits and Systems*, **12**(12), pp. 1827–1838.

[61] S. M. Kang, Y. L. (1999). CMOS Digital Integrated Circuits: Analysis and Design. *McGraw-Hill*.

[62] S. Murali, L. B. and G. D. Micheli (2007). An Application-Specific Design Methodology for On-Chip Crossbar Generation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, **26**(7), pp. 1283–1296.

[63] Schabas, K. and S. D. Brown (2003). Using Logic Duplication to Improve Performance in FPGAs. *In Proceedings FPGAs*, pp. 136–142.

[64] Sentovich, E., K. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. Stephan, R. Brayton, and A. Sangiovanni-Vincentelli (1992). SIS: A system for sequential circuit synthesis. *In Technical Report*.

[65] Singh, D. P. and S. D. Brown (2002). Incremental Placement for Layout-Driven Optimizations on FPGAs. *IEEE/ACM International Conference on Computer-Aided Design*, pp. 752–759.

[66] Snee, R. (1977). Validation of regression models: Methods and examples. *Technometrics*, **19**, pp. 415–428.

[67] Tessier, R. (1998). Negotiated A* Routing for FPGAs. *In Proceedings of the 5th Canadian Workshop on Field-Programmable Devices*.

[68] Tsui, C. Y., M. Pedram, and A. M. Despain (1993). Efficient estimation of dynamic power consumption under a real delay model. *IEEE International Conference on Computer-Aided Design*, pp. 224–228.

[69] V. Betz, J. R. and A. Marquardt (1999). Architecture and CAD for Deep-Submicron FPGAs. *Kluwer Academic Publishers*.

[70] Wikipedia, T. F. E. (2013). Geode (processor). Wikimedia Foundation, Inc.

[71] Wikipedia, T. F. E. (2013). Snapdragon (system on chip). Wikimedia Foundation Inc.

[72] Wikipedia, T. F. E. (2013). Tegra. Wikimedia Foundation, Inc.

[73] Wikipedia, T. F. E. (2013). Wafer (Electronics). Wikimedia Foundation, Inc.

[74] Wilson, R. (2010). ARM Extends Toward a Future of Low-Power System Design. Altera Inside Edge eNewsletter.

[75] Xakellis, M. and F. Najm (1994). Statistical Estimation of the Switching Activity in Digital Circuits. *31st Conference on Design Automation*, pp. 728–733.

[76] Y. K. Leow, A. A. and S. Lysecky (Accepted for publication). An Analytical Model for Evaluating Static Power of Homogeneous FPGA Architectures. *ACM Transactions Reconfigurable Technology and System (TRETS).*