Impact of Custom Interconnect Masks on Cost and Performance of Structured ASICs

by

Usman Ahmed

B.Eng., National University of Sciences and Technology, 2001
M.A.Sc., Ryerson University, 2005

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

Doctor of Philosophy

in

THE FACULTY OF GRADUATE STUDIES
(Electrical and Computer Engineering)

The University Of British Columbia
(Vancouver)

April 2011

Abstract

As process technology scales, the design effort and Non-Recurring Engineering (NRE) costs associated with the development of Integrated Circuits (ICs) is becoming extremely high. One of the main reasons is the high cost of preparing and processing IC fabrication masks. The design effort and cost can be reduced by employing Structured Application-Specific ICs (Structured ASICs). Structured ASICs are partially fabricated ICs that require only a subset of design-specific custom masks for their completion.

In this dissertation, we investigate the impact of design-specific masks on the area, delay, power, and die-cost of Structured ASICs. We divide Structured ASICs into two categories depending on the types of masks (metal and/or via masks) needed for customization: *Metal-Programmable Structured ASICs (MPSAs)* that require custom metal and via masks; and *Via-Programmable Structured ASICs (VPSAs)* that only require custom via masks. We define the metal layers used for routing that can be configured by one or more via, or metal-and-via masks as *configurable layers*. We then investigate the area, delay, power, and cost trends for MPSAs and VPSAs as a function of configurable layers.

The results show that the number of configurable layers has a significant impact on die-cost. In small MPSAs (area $< 100mm^2$), two configurable layers result in the lowest cost, whereas the lowest cost in large MPSAs (area $> 100mm^2$) is achieved with three or four configurable layers. The lowest cost in VPSAs is also obtained with four configurable layers. The best delay and power in MPSAs is achieved with three or four configurable layers. In VPSAs, four configurable layers lead to lowest power and delay, except when logic blocks have a large layout area. MPSAs have up to $5\times$, $10\times$, and $3.5\times$ better area, delay, and power than VPSAs, respectively. However, VPSAs can be up to 50% less expensive than MPSAs. The results also demonstrate that VPSAs are very sensitive to the architecture of fixed-metal segments; VPSAs with different fixed-metal architectures can have a gap of up to 60% in area, 89% in delay, 85% in power, and 36% in die-cost.

Preface

The research presented in this dissertation has also appeared in [28–30]. For each of these papers, I performed the experiments, and the analysis of the results. The manuscripts were also prepared by myself. My supervisors, Dr. Guy Lemieux and Dr. Steve Wilton, provided advice on the methodology, and made editorial changes to the manuscripts.

Parts of Chapters 3 and 4 have been included in [28–30]. Chapter 5 is based on [28, 30], and Chapter 6 is based on [29].

[28] U. Ahmed, G. Lemieux, and S. Wilton. Area, delay, power, and cost trends for Metal-Programmable Structured ASICs (MPSAs). In *IEEE International Conference on Field Programmable Technology (ICFPT)*, pages 278-284, Dec. 2009.

[29] U. Ahmed, G. Lemieux, and S. Wilton. The impact of interconnect architecture on Via-Programmed Structured ASICs (VPSAs). In *Proceedings of ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, pages 263-272, Feb. 2010.

[30] U. Ahmed, G. Lemieux, and S. Wilton. Performance and cost tradeoffs in Metal-Programmable Structured ASICs (MPSAs). *IEEE Transactions on VLSI Systems*, Oct. 2010. doi:10.1109/TVLSI.2010.2076841.

Table of Contents

List of Tables

List of Figures

Glossary

ASIC	Application-Specific IC
CAD	Computer-Aided Design
CBIC	Cell-based IC
DSM	Deep-Submicron
DSP	Digital Signal Processing
ECO	Engineering Change Order
FPGA	Field Programmable Gate Array
IC	Integrated Circuit
IP	Intellectual Property
ISPD	International Symposium on Physical Design
LUT	Look-Up Table
MCNC	Microelectronics Centre of North Carolina

MPSA	Metal-Programmable Structured ASIC
NRE	Non-Recurring Engineering
OPC	Optical Proximity Correction
PLA	Programmable Logic Array
PSM	Phase Shift Masks
RAM	Random Access Memory
RDR	Restricted Design Rule
RET	Resolution Enhancement Technique
RMST	Rectilinear Minimal Spanning Tree
RSMT	Rectilinear Steiner Minimal Tree
SRAM	Static RAM
SoC	System-On-a-Chip
TSMC	Taiwan Semiconductor Manufacturing Company
VLSI	Very Large Scale Integration
VPSA	Via-Programmable Structured ASIC

Acknowledgements

I would like to thank both of my research supervisors, Dr. Guy Lemieux and Dr. Steve Wilton, for their guidance, dedication, technical insight, patience, and encouragement. I consider myself very fortunate to have them as my mentors for the Ph.D. research.

In addition, I would like to thank my external examiner, Dr. Herman Schmit, and my university examiners, Dr. Alan Hu and Dr. Shahriar Mirabbasi, for providing valuable comments that helped improve the quality of this dissertation. I am also thankful to Dr. Res Saleh and Dr. Andre Ivanov for being part of my supervisory committee.

I would also like to thank Dr. Roberto Rosales and all the other members of the SoC lab who made my stay at UBC very pleasant. I am thankful to Ali Bakhoda, Assem Bsoul, Brad Quinton, Christian Grecu, David Grant, Dipanjan Sengupta, Eddie Hung, Joydip Das, Reza Molavi, Samad Sheikhaei, Scott Chin, Sohaib Majzoub, and Uthman Al-Saiari.

Finally, and most importantly, I am very grateful to my family—my parents, my sister, my wife, and my newly arrived son—for their continuous support and constant encouragement. Thank you all!

Chapter 1

Introduction

1.1 Motivation

Design and implementation of modern Very Large Scale Integration (VLSI) circuits is a complex process. These circuits are traditionally fabricated using photolithographic techniques [60, 66, 82, 93]. Photolithography is a process in which geometric patterns are transferred using ultraviolet light from a photographic mask to a light-sensitive material placed on the target device. Many such masks are required to build layers of different materials. These layered geometric patterns ultimately define the transistors and their interconnect structure. Technology scaling has steadily reduced the sizes of transistors and interconnect wires, improving performance of the resulting Integrated Circuits (ICs). Small transistors and wires require smaller feature sizes on the masks. Technology scaling has now reached a stage where the feature sizes on the masks are much smaller than the wavelength of the light used to expose them. This process, known as *subwavelength lithography*, has made the fabrication process even more challenging [67].

Technology scaling has also greatly magnified many of the physical semiconductor imperfections and design challenges which had negligible impact in older technology nodes. These include variation, leakage, and signal integrity issues [37, 64, 115]. Variation occurs because the transistors and wires cannot be fabricated to exact specifications; tolerances cannot be scaled as easily as dimensions. The small transistors have higher leakage current and this increases the power dissipation even when the device is idle. The signal integrity issues occur when transition of one signal adds noise to another signal, or causes a transition on an unrelated signal. Common signal integrity issues include noise coupling [63], inductance effects [65], and IR drop [48]. All of these problems are commonly known as Deep-Submicron (DSM) effects and these make the design process very difficult.

These challenges are being mitigated using several approaches. Resolution Enhancement Techniques (RETs) are used to cope with subwavelength lithography problems [102, 120]. Optical Proximity Correction (OPC) [49, 79, 84, 103] and the use of Phase Shift Masks (PSM) [75, 78, 80] are two of the commonly used RETs. In these techniques, geometric layout shapes are transformed before fabrication of the mask in such a way that the resulting distorted shapes produce the intended layout shapes. However, these techniques are very time and memory intensive and significantly increase the cost of producing and inspecting each mask. As an example, the cost of a 90nm mask-set is 6$\times$ the cost of a 180nm mask-set [90].

DSM effects can also be mitigated by using more sophisticated manufacturing and Computer-Aided Design (CAD) techniques. The manufacturing technology can provide high- and low-leakage transistors.[1] The total leakage current can be reduced by utilizing low-leakage transistors for less critical parts of the design. The CAD tools model the physical semiconductor effects at a very detailed level to detect and reduce signal integrity issues. However, this significantly increases the runtime and cost of the CAD tools. The use of complicated manufacturing and CAD techniques have significantly increased the design time and cost; the total cost to design and fabricate an IC at 90nm in 2005 was \$25M, whereas it was only \$4M for 180nm [90]. As a result of these cost increases, access to the latest process technologies is becoming limited and many designs are still being implemented using older process technologies; advanced process technologies, including 90nm and below, accounted for only 49% of Taiwan Semiconductor Manufacturing Company (TSMC) revenue in 2009 [20].

Cell-based ICs (CBICs) (also known as standard cells) have been the dominant choice for implementing digital circuits for high performance and/or high volume applications [52, 107, 109]. Designs select cells from a standard cell library, which consists of pre-formed cells with different logic functions and various drive strengths. In this way, a CBIC design contains transistors and wires that are customized to a particular application. Applying RETs and accounting for DSM

[1]Most modern processes provide high-threshold voltage (V_t) and low-V_t transistors; high-V_t transistors have low leakage, and vice versa.

effects is, therefore, very costly and time consuming.

Field Programmable Gate Arrays (FPGAs) [35, 38] provide one way of reducing the design time and cost. In FPGAs, all the transistors and wires are prefabricated. This lowers the cost by averaging it across all the customers. Also, since the device is completely prefabricated, the DSM issues have already been dealt with by the FPGA vendor. However, there is a significant gap between the power, delay, and area performance of FPGAs compared to CBICs [74]. Consequently, FPGAs may not be suitable for applications which require low power, high volume or high performance. In particular, applications in the growing portable and hand-held device market often require lower power than what is available in today's FPGAs, but faster turn-around time than what can be achieved using CBICs.

Structured Application-Specific ICs (ASICs) offer one solution to these problems [122]. In Structured ASICs, part of the device including transistors and a few metal and/or via layers are prefabricated, and the remaining layers can be customized to implement a particular application. This makes Structured ASICs a good compromise between CBICs and FPGAs — the masks for the prefabricated portion are shared across all the customers which lowers the cost compared to CBICs, and the use of custom masks to fabricate the remaining portion improves the performance relative to FPGAs.

Although Structured ASICs were commercially introduced several years ago, they have not achieved the traction that many anticipated. There are many possible reasons for this, including unfamiliar technology, immature CAD, and claimed advantages which have not yet been concretely demonstrated. We believe that, as technology scaling continues, the advantages of Structured ASICs will become even more compelling, especially for low-power hand-held applications. When that happens, we will need optimized architectures, CAD tools, and design flows. In this research, we take a step in this direction and investigate the cost, area, delay, and power trade-offs in Structured ASICs.

The key parameter that determines the cost and area of a Structured ASIC is the number of custom masks required to implement a particular circuit. The customization also affects the turn-around time, delay, and power of Structured ASIC devices. In this dissertation, we focus on the amount of customization required in Structured ASICs.

1.2 Research Problem

In this research, we seek to answer the following question:

How are the cost and performance of Structured ASICs affected by the number of custom masks and the number of routing layers?

The custom masks refer to the design-specific metal and via masks, whereas the routing layers are the metal and via layers in the device that are used to make connections between different parts of the pre-fabricated portion. The number of custom masks and the number of routing layers appear to be related; more routing layers might imply more custom masks and vice versa. However, it is important to make this distinction because a Structured ASIC device with many routing layers may require only a few custom masks.[2]

Intuitively, the number of custom masks should be minimized, since this minimizes the cost to the designer, and may shorten the turn-around time. On the other hand, if the device is not flexible enough, the implementation of a circuit on the Structured ASIC will require more space and possibly be slower and consume more power than is required. This conflicting criteria suggest that there is an optimum number of custom masks. Understanding this trade-off is key to creating efficient and cost-effective Structured ASICs.

This work is significant for a number of reasons. To the best of our knowledge, this is the first academic effort to study the custom masks in Structured ASICs across a range of architectures. Past academic efforts have been limited to fixed architectures with a fixed number of custom masks. The research is also valuable to commercial Structured ASIC vendors as they decide the right amount of custom masks for their Structured ASIC products. In particular, earlier Structured ASICs, as described in Chapter 2, spanned a very wide range of custom masks from a single via mask to six metal and six via masks. Therefore, it is very helpful to validate whether these products offered too much or not enough flexibility for the desired cost, area, speed, and power performance.

The goal of this dissertation is not to uncover the best specific Structured ASIC architecture. As a side-effect of answering the above question across different architectures, we will gain insight into what makes a good architecture and what

[2]For example, a device that uses four metal and four via layers for routing may require only a single custom via mask. Section 6.3.3 describes this in detail.

makes a bad architecture. However, the primary focus of this research is to investigate the impact of the number of custom masks on the cost and performance of Structured ASICs architectures.

1.3 Research Approach

To answer the research question described in Section 1.2, we employ an experimental approach. Using a CAD flow and a set of benchmarks containing different types of circuits, we study how the performance and cost of a Structured ASIC is affected by a change in the number of custom masks and routing layers. Unlike previous studies, we do not focus on a single Structured ASIC architecture; rather, we consider a range of potential Structured ASIC architectures. The architecture space we explore is defined by different types of logic blocks and routing fabrics. We create abstract models to represent each architecture at a suitable level of detail. Custom CAD tools are then used to map a set of benchmark circuits to each architecture under consideration. Detailed area, delay, power, and cost models are then used to evaluate each implementation on each architecture. From these results, the efficiency of each potential architecture is assessed. Although this experimental approach relies on models rather than measured device results, it allows us to consider a wider range of architectures than would be possible unless each potential architecture is laid out and/or manufactured.

An important part of the experimental framework is a detailed cost model which relates the area and the number of custom masks and routing layers of a Structured ASIC device to its dollar-cost. The cost model considers the manufacturing cost of each device, the mask-set cost for a design, and device volume requirements. The cost model is described in Chapter 3.

Another important part of this research is a CAD flow that allows the comparison of architectures with different custom mask requirements. The metrics used for the comparison include die-cost, area, delay and power. For each potential architecture, we will use the CAD flow to estimate the cost, area, delay and power metrics for different possibilities of customization. Chapter 4 describes the experimental framework and CAD flow in detail.

We then divide Structured ASICs into two categories: one that requires both custom metal and custom via masks; and the other in which all the metal masks are fixed and only custom via masks are needed. We define the former class as Metal-Programmable Structured ASICs (MPSAs), and the latter as Via-Programmable Structured ASICs (VPSAs). The results for MPSAs and VPSAs are shown separately in Chapters 5 and 6, respectively.

1.4 Contributions

The key contributions of the research are as follows:

- A cost model that relates die-area and number of routing layers to Structured ASIC die-cost. The cost model takes into account mask-set cost, wafer processing costs, device volume requirements, die-yield, die-area, and number of custom masks. It estimates the dollar-cost of a single Structured ASIC die. Traditionally, for CBICs and FPGAs the area has been used as a proxy to estimate the cost of the device. However, because of the large mask-set costs, the number of custom masks in a Structured ASIC have a significant impact on its cost. The cost model allows us to demonstrate that in most cases, contrary to the popular belief, the area savings due to the availability of more routing layers does not translate into a cost savings.

- A sensitivity analysis of the Structured ASIC die-cost to various parameters such as device volume requirements and mask-set costs. In terms of volume requirement, the MPSA die-cost is more sensitive to per-customer volume (V_c) than total device volume (V_{tot}); a smaller V_c makes MPSAs more cost-effective than CBICs. Similarly, increasing mask-set costs also make MPSAs more economical than CBICs.

- A CAD flow that places and routes a benchmark circuit for a variety of Structured ASIC architectures. The CAD flow makes use of open-source CBIC global placement and global routing tools. It includes a custom detailed placer to perform legalization of different types of logic blocks. It also includes a custom detailed router to perform detail routing on fixed-metal routing fabrics found in VPSAs. The CAD flow provides area, delay, and power estimates for a benchmark circuit when implemented on a particular Structured ASIC architecture.

- Dollar-cost, area, delay and power trends for MPSAs, and VPSAs as a function of the number of routing layers. The area, delay, and power improve as the number of routing layers increases for both MPSAs and VPSAs. However, as the layout area of the logic block increases, the improvement diminishes. In terms of die-cost, small MPSA dies (die-area less than $100mm^2$) have the lowest cost with 2 routing layers; the lowest cost in large MPSA dies (die-area greater than $100mm^2$) is obtained with 3 to 4 routing layers. The lowest cost in VPSAs is achieved with 4 routing layers using a single custom via mask, except when sparse logic blocks (large layout area) are used. MPSAs are up to $10\times$, $3.5\times$, and $5\times$ better than VPSAs in terms of delay, power, and area respectively. However, in terms of die-cost, VPSAs are up to 50% less expensive.

These contributions are an important step towards the development of improved Structured ASIC architectures and CAD algorithms. To the best of our knowledge, the Structured ASIC die-cost has not been directly used as a performance metric in the past. The proposed cost model will help to compare the different architectural choices in future Structured ASICs by considering their dollar-cost. The CAD flow that has been set up as part of this research can be instrumental in the future research related to Structured ASICs. The flow provides a mechanism to analyze different architectures and different CAD algorithms. In this way, it will facilitate the development of new Structured ASIC architectures and more efficient CAD algorithms. Finally, the performance and cost trends that have been presented in this dissertation provide an insight into various trade-offs involved in Structured ASIC design. These results can also serve as a baseline result for future Structured ASIC studies.

Chapter 2

Background and Related Work

2.1 Overview

In this chapter, we describe the Structured ASIC concept in detail. We identify two different methods for configuring these devices and explain some of their advantages in relation to FPGAs and CBICs. We also review prior academic research and commercial efforts on Structured ASICs.

The Structured ASIC die-cost model described in Chapter 3 makes use of a die-yield model. We review the basics of this die-yield model in this chapter.

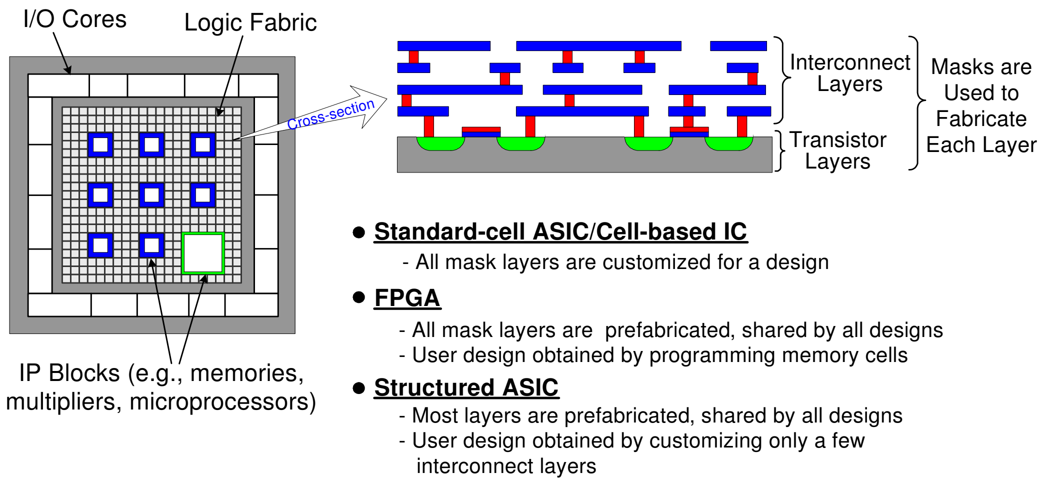
Finally, we review existing open source placement and routing tools. Several of these tools are a part of the experimental CAD flow used in this work.

2.2 Structured ASIC

ICs are multilevel structures with transistors at the bottom and the interconnect layers on top of the transistors. The interconnect layers are made up of metal wire segments and vias. Each layer in the IC structure is fabricated using one or more photo-lithographic masks. Two of the most common IC implementation technologies are CBICs and FPGAs. In CBICs, all the transistors and interconnect layers are specific for a single circuit implementation. In FPGAs, all the layers are prefabricated, and the chip is customized in the field by either setting the state of internal Static RAM (SRAM), flash memory bits, or programmable fuses/antifuses.

Structured ASICs aim to combine the good features of CBICs and FPGAs. A Structured ASIC is a generic IC which can be customized to implement any digital circuit by adding one or more metal layers and/or via layers. A Structured ASIC vendor produces many such generic ICs and may establish an inventory of chips in a partially fabricated state, where the top metal and via layers have not yet been fabricated. A designer who wants to implement a circuit on a Structured ASIC then provides design information to the Structured ASIC vendor, who customizes the pre-fabricated chips by adding the extra layers. Some of the extra layers may be designed by the vendor, while the others may be user-defined. The Structured ASIC concept is illustrated in Figure 2.1.

Structured ASICs are often seen as a compromise between CBICs and FPGAs, combining advantages and disadvantages of each design style. CBICs generally



I/O Cores
Logic Fabric
Cross-section
Interconnect Layers
Transistor Layers
Masks are Used to Fabricate Each Layer
IP Blocks (e.g., memories, multipliers, microprocessors)
● Standard-cell ASIC/Cell-based IC
- All mask layers are customized for a design
● FPGA
- All mask layers are prefabricated, shared by all designs
- User design obtained by programming memory cells
● Structured ASIC
- Most layers are prefabricated, shared by all designs
- User design obtained by customizing only a few interconnect layers

Figure 2.1: The Structured ASIC concept

have the best area, delay and power performance. However, there is a large Non-Recurring Engineering (NRE) cost of \$1–3M associated with CBIC design to pay for a complete set of masks [121]. Thus, despite the small area, they are cost-effective only when the devices have to be produced in very large numbers (e.g., 500*k* units or more [108]). SRAM-programmable FPGAs, on the other hand, have negligible NRE cost, but they are 3–4× slower, consume 14× more dynamic power, and are 35× larger than CBICs [73, 74]. The large die-area increases per-device cost, making them suitable only for applications which have small volume requirements (e.g., less than 5*k* units [108]). Many applications have moderate volume requirements, but they demand better performance and lower per-device cost than FPGAs. An example is the emerging portable and handheld devices. Such applications can be addressed by Structured ASICs. Structured ASICs are 1.5–2× slower, and their area overhead is 1.5–3× that of CBICs [87, 121, 122].

Just as in the case of modern FPGAs [16–18, 23–25], Intellectual Property (IP) blocks such as embedded memory, Digital Signal Processing (DSP) blocks and microprocessors can also be included in a Structured ASIC to improve the performance, density and power dissipation.

Structured ASICs are closely related to a previous technology called Gate Array technology [36, 57]. Like Structured ASICs, Gate Arrays can be customized by implementing one or more metal layers. The primary difference is that Gate Arrays prefabricate only the transistors, whereas Structured ASICs prefabricate some of the interconnect layers as well. This partial fabrication of the device improves the cost and turnaround time and makes Structured ASICs very attractive for many applications, especially for the low-power, hand-held devices. In the following, we examine some of the advantages and disadvantages of Structured ASICs relative to FPGAs and CBICs in more detail.

2.2.1 Advantages

Low Cost

The most important advantage of Structured ASICs is the low development and fabrication cost. A cost comparison of FPGAs, Structured ASICs, and CBICs for a one million gate design in 130nm process is shown in Table 2.1. It can be seen that FPGAs have a very low NRE cost. However, this only benefits applications that require small volumes of less than 5000 units (e.g., medical instruments). For applications that require relatively large volume, FPGAs can be too expensive because of high per-device cost. On the other hand, the NRE cost for CBICs is

Table 2.1: Cost comparison of 1M gate design in 130*nm* [108]

	FPGA	**Structured ASIC**	**CBIC**
Total Design Cost	$\sim \$165k$	$\sim \$500k$	$\sim \$5.5M$
Vendor NRE	None	$\sim \$100k - \$200k$	$\sim \$1M - \$3M$
#Tools Required	2 to 3	2 to 3	6 to 10
Cost of Tools	$\sim \$30k$	$\sim \$120k - \$250k$	$> \$300k$
#Engineers	1 to 2	2 to 3	5 to 7
Price per chip	$\$220 - \$1k$	~\$30 to \$150	$\sim$ \$ 30
Total Unit Cost (Qty: 1k)	$\sim$ \$1000('03)	$\sim \$500 - \$650$	$\$55k$
Total Unit Cost (Qty: 5k)	$\sim$ \$220(4Q'04)	$\sim \$100 - \$150$	$\$1.1k$
Total Unit Cost (Qty: 500k)	$\sim$ \$40(4Q'04)	$> \$21$	$\$11 - \$20$

in the range of \$1–3M. This is mainly due to the cost of generating a full mask set; the average cost of a mask set for a 90nm process when it was introduced was \$2 million [90]. The mask set costs are increasing at a rapid rate, and this makes the development of a CBIC infeasible for many companies. A Structured ASIC solution requires the masks for only the top metal layers to be generated. This significantly reduces the mask cost, which lowers the NRE costs. For the example design of Table 2.1, the NRE cost for a Structured ASIC implementation is only 10% of a CBIC implementation. Thus, for moderate volume requirements (e.g., less than 500*k* units), Structured ASICs are more cost-effective than both FPGAs and CBICs.

Regularity

Another major advantage of Structured ASICs is their regular structure. The irregular structure of CBICs makes them highly prone to process variation. Process variation causes chip failures which reduce the yield and increase the per-chip cost. The main cause of these variations is the use of deep sub-wavelength lithography to print the transistor and interconnect layout features which causes distortion in the printed features. This distortion depends not only on the layout of the feature being printed but also on other features in its neighborhood [120]. OPC is a commonly-used technique to reduce these distortions. In OPC, geometric layout shapes are transformed before fabrication so that the resulting distorted shapes are closer to the intended layout shapes [77, 119]. Initially, these transformations were rule-based, but now OPC is performed using simulation models of lithography. This is a very time-consuming and memory-intensive process. The large number of layout patterns in a typical CBIC makes this very difficult. A Structured ASIC fabric is built by repeating the same basic logic block design. Thus, the number of unique features that need to be printed are very limited and OPC and other RETs can then be efficiently applied.

The regularity also helps with technology scaling. Each cell in the standard cell library is tuned for a specific speed and power requirement. The cells designed for an old process cannot be used with a new process because of a new set of design rules. As a result, whenever technology scales, the entire cell library needs to be rebuilt. This is known as *library migration*. Building a new library requires the custom layout of each cell and a careful characterization of each cell's behavior. This can be a very time-consuming process. Regular circuit structures simplify this process, since there are only a few basic cells to be designed and characterized [85].

The regularity of Structured ASICs can thus improve the yield, and reduce the design effort and design cost compared to CBICs.

Low Power

Leakage in the SRAM cells and interconnect buffers makes it very hard for SRAM-based FPGAs to meet the power requirements of low-power applications such as portable and hand-held devices. Also, these devices tend to spend most of their time in “idle” state, thus they demand extremely low static power dissipation in addition to low dynamic power [68]. Flash-based FPGAs have been introduced by Actel to reduce the static power dissipation [8, 15]. However, these FPGAs still have Flash cells, multiplexers and buffers that are required for re-configurability. This extra circuitry is not required by Structured ASICs. Although FPGAs consume $14\times$ more power than CBICs [73, 74], Structured ASICs consume much less, only 2–$3\times$ of a CBIC [122]. Thus, low-power applications can potentially be better implemented on a Structured ASIC.

Fast Turn-around Time

The typical turn-around time (time to fabricate the device once the design has been completed) for a CBIC is 20 to 24 weeks [87]. Often, CBIC designers require one or more re-spins before volume production which increases the cost and also affects the time to market. This long period poses a risk of a lost market opportunity.

According to International Business Strategies, 3 to 12 months of delay can affect the sales by 27% to 91% [47]. Structured ASICs, on the other hand, have a much smaller turnaround time because most of the chip is already fabricated and only a few masks need to be generated. Also, many of the DSM and signal integrity issues, which take a significant amount of time to reach closure in standard cell ASICs, have already been taken care of. Because of these characteristics, Structured ASICs can improve the time to market by 4 to 6 months [121].

2.2.2 Disadvantages

Structured ASICs have a few disadvantages. Structured ASICs cannot be customized in the field. Structured ASICs cannot be reconfigured at zero cost because of an associated NRE cost. Also, many ICs are developed using pre-designed IP blocks (e.g., embedded memory, DSP/communication blocks, microprocessors etc.). In this domain, a Structured ASIC device cannot be used if it does not have the right mix of IP blocks for a particular application [13]. Finally, the infrastructure and CAD tools for Structured ASICs are less mature than those for FPGAs and CBICs.

2.2.3 Types

Structured ASICs can be broadly categorized based on which mask layers need to be customized. We divide Structured ASICs into two classes:

1. Metal-Programmable Structured ASICs (MPSAs)

2. Via-Programmable Structured ASICs (VPSAs)

MPSAs can be customized using metal and via layers, whereas in VPSAs, all the metal layers are fixed, and the device can only be configured through one or more via layers. There are different architectural decisions that a Structured ASIC vendor can make regarding the interconnect and logic fabric of MPSAs and VPSAs. In the following, we describe some of the possibilities for these architectural decisions.

Interconnect Fabric

An interconnect fabric is used to make connections between different logic blocks. In an MPSA interconnect fabric, selected metal and via layers can be customized to implement a user circuit. This scheme offers most flexibility for routing but increases the customization time and cost because, in addition to the via masks, it requires custom masks for each configurable metal layer to be processed (PSM, OPC etc.), created, and verified. Metal layers are more complex than via layers.

In VPSAs, all metal layers are fixed and only via layers are customizable to implement a given user circuit. One way to implement such an architecture is to position the fixed metal segments in different layers orthogonal to each other to form a crossbar structure. The crossbar is configured by inserting vias at sites where metal segments in adjacent layers intersect.

There are different ways of designing a fixed-metal fabric for VPSAs. All the metal segments can either have the same length, or the fabric can contain a mix of long and short segments. If fully utilized, long segments can improve performance; however, if the entire length of the segment is not utilized, the capacitance of the unused portion can result in power and delay degradation. Selecting the length for the long segments is an important architectural decision for VPSAs.

Similarly, the mechanism for connecting wire segments to adjacent wire segments in the same layer has different possibilities. The fabric can either use a segment from an adjacent layer, or use small, dedicated segments known as *jumpers* to connect adjacent wires in the same layer. Jumpers can be implemented either above or below the wires they connect. Figure 2.2 illustrates a jumper-based interconnect fabric. In Chapter 6, the interconnect fabrics used in our experiments are described in detail.

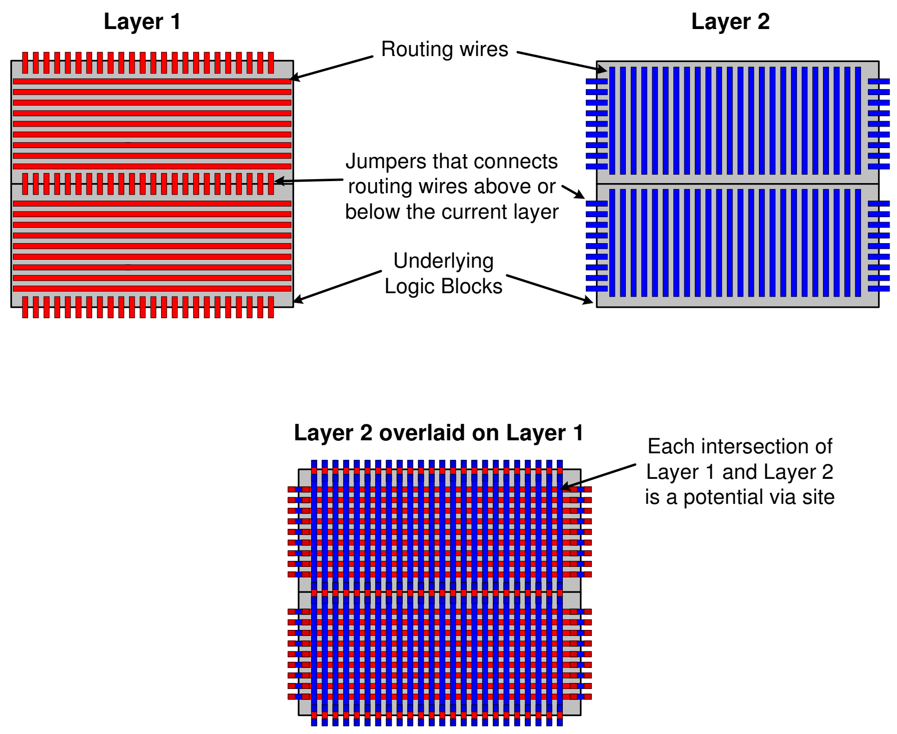

Layer 1
Layer 2
Routing wires
Jumpers that connects routing wires above or below the current layer
Underlying Logic Blocks
Layer 2 overlaid on Layer 1
Each intersection of Layer 1 and Layer 2 is a potential via site

Figure 2.2: Jumper-based interconnect fabric

It is also possible to fix some of the via layers, in addition to the fixed metal layers. This makes the fabric more restrictive which could degrade the performance of the device. However, such a fabric reduces the device cost since fewer custom masks are required. We describe one such fabric in Chapter 6.

Logic Fabric

The logic functionality of a particular design is implemented by the logic blocks. Depending on the granularity of the basic logic cell, there are different possibilities for the logic blocks. One possibility is to design logic blocks using fine-grained basic gates such as NAND, XOR, flip-flop, buffers, etc. Such logic blocks have the advantage that they do not need to be configured in any way; all the configuration is done in the interconnect fabric. An example of such a logic block is shown in Figure 2.3. The highlighted points in the figure represent pins that connect to the routing fabric and basic gates are drawn instead of actual layout shapes for clarity.

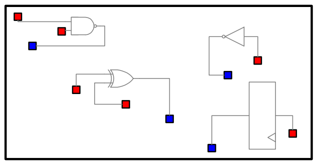


Figure 2.3: Fine-grained logic block

The use of medium-grained blocks such as Look-Up Tables (LUTs) is another possibility for building the logic fabric. Unlike fine-grained gates, LUTs need to be configured to implement a particular logic function. This configurability can be implemented by connecting the configuration inputs of the LUT to ground or

V_{dd}. There are three different possible ways of configuring these logic blocks: (1) to use a lower layer via (e.g., between metal-1 and metal-2) to connect the LUT configuration inputs to ground or V_{dd}; (2) to connect the configuration inputs of the LUT to SRAM cells; or (3) to use higher-level vias (connections between upper metal layers) to connect the LUT configuration inputs to ground or V_{dd}. Each possibility has its own advantages and disadvantages. In the first case, there are fewer I/Os at the higher levels, however, manufacturing the device only up to the first few layers may increase both the manufacturing cost and manufacturing time. The use of SRAM cells in the second case increases the area of logic block. In the third case, the presence of tall via-stacks can create routability problems since the metal segments cannot pass through these via stacks. The three different possibilities for the medium-grained logic blocks are shown in Figure 2.4. The routability problem posed by using higher-level vias is illustrated in Figure 2.5.

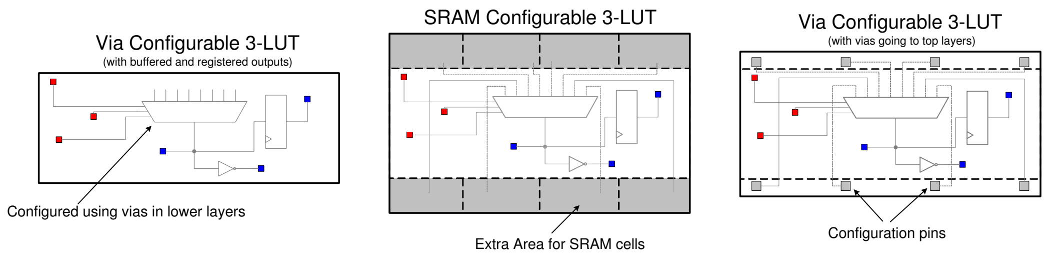

Via Configurable 3-LUT
(with buffered and registered outputs)
Configured using vias in lower layers
SRAM Configurable 3-LUT
Extra Area for SRAM cells
Via Configurable 3-LUT
(with vias going to top layers)
Configuration pins

Figure 2.4: Different possibilities for medium-grained logic block

A third way of creating logic blocks is to use coarse-grained blocks such as Programmable Logic Arrays (PLAs). The number of inputs, outputs, and product terms for the PLA are architectural parameters that are fixed during initial fabrication. The connections between the AND and OR plane, as well as the connections

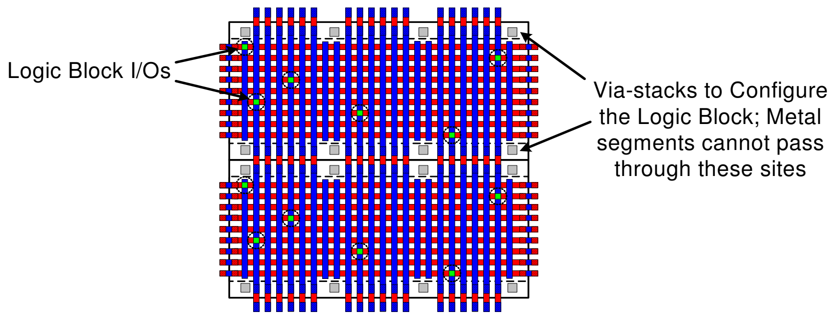

Logic Block I/Os
Via-stacks to Configure the Logic Block; Metal segments cannot pass through these sites

Figure 2.5: Impact of higher-level configuration-vias on routability

between the PLA and the interconnect fabric can be configured either by making the connections using lower-level vias or by bringing the configuration points to higher layers.

2.3 Prior Work on Structured ASICs

2.3.1 Academic Efforts

There has been only a moderate amount of academic research related to Structured ASICs. In each case, specific logic blocks and routing fabrics have been proposed and evaluated.

Ran and Sadowska have proposed a VPSA [94–97]. Their proposed logic block is shown in Figure 2.6. The logic block is made up of Via-Configurable Cells (VCC), which are composed of vertically aligned transistor pairs and n-/p-diffusion strips [94]. Metal 1 (M1), M2, and via 1 (V1) layers are used to define the cell. The M1 and M2 layers are fixed whereas V1 is customizable. Placing vias at various intersections of M1 and M2 segments allows VCC to implement combinational gates, sequential gates, SRAM cells and different arithmetic units such as adders and multipliers. Four VCCs are grouped to form a via-configurable logic block (VCB). The routing fabric is a crossbar structure that is laid on top of the VCB using M3 and above. All the metal is fixed and only the vias between the intersecting wires of the crossbar are used to route the circuits. They show that when a crossbar structure with only M3 and M4 is used for routing, the area increase is 4x and the delay increase is 1.5x, relative to a standard cell implementation. They also consider a routing fabric with four metal layers (M3, M4, M5, and M6) and show that if the configurability is reduced to only V3 (the via layer between M3 and M4), an area and delay penalty is incurred. This is because some of the metal in M4 is now dedicated to provide a connection between M3 and M5/M6, reducing the number of M4 segments available for routing. This results in an area increase up to 46% and delay penalty up to 25%, compared to the case when all the via layers (V1–V5) are configurable.

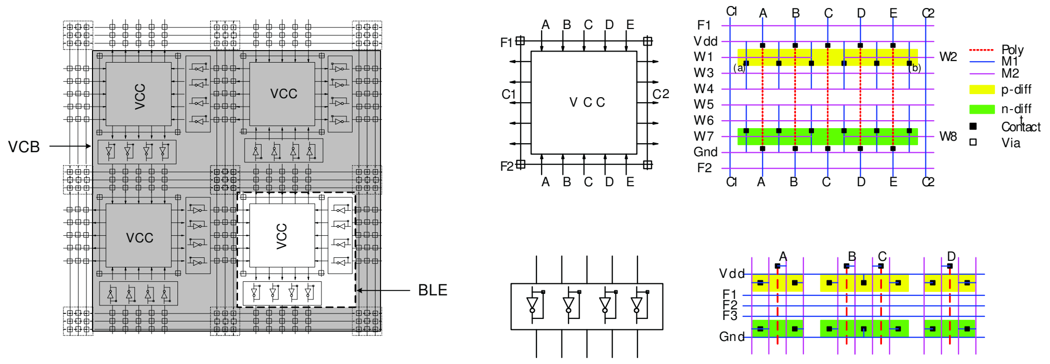


Figure 2.6: Logic block proposed by Ran and Sadowska [94]

In [92], Pileggi et al. compare a VPSA using LUTs to standard cell designs.

Each basic cell in the VPSA fabric consists of a via-programmable LUT, two input-invertable three-input NAND gates, seven inverters and one flip-flop. This fabric is improved for enhanced performance and better density by Koorapaty et al. who proposed a logic block consisting of a XOR gate, a three-input NAND gate, 2-to-1 MUXes and inverters [71]. The logic block is configured using only lower-layer vias.

Kheterpal et al. have explored different routing architectures that can be used with a VPSA [69]. They compared the performance of structured routing and a via-configurable routing fabric to ASIC routing. In structured routing, metal segments can be customized but they conform to a strict grid whereas in the via-configurable routing, the metal segments are fixed and form a crossbar structure. Experiments were conducted for a process with 6 metal layers where four metal layers are available for routing. They show that structured routing degrades the performance by 5% and 6% relative to the ASIC routing solution for a datapath circuit and a network switch circuit, respectively. In contrast, the performance loss for via-configurable routing was 24% and 21%, respectively, for the same two circuits.

Veredas et al. have proposed a MPSA called Zelix [116, 117]. Their goal is to reduce the large area overhead of FPGAs and not to improve the performance. Zelix is based on mask-configurable look-up tables and a regular routing fabric. The logic architecture has the same topology and gate-level logic elements as a CLB in the Xilinx Virtex-II Pro FPGA. The switch blocks and connection blocks utilize fully populated crossbars and are configured by vias. Internal signals, clocks and flip-flop control signals are routed using M1, M2 and M3 layers. The power grid is implemented in M5. The configuration of Zelix is done by customizing M3, M4 and the vias between these layers. The interconnect is based on length-1 wires and there is a buffer for every wire. It is reported that, with 30 tracks per channel, the Zelix area is 82% smaller than a Xilinx Virtex-II Pro.

Nakamura et al. have proposed a VPSA known as VPEX, which is designed for electron-beam (EB) direct writing [31, 88]. The VPEX logic block consists of an exclusive OR and an inverter. The XOR is implemented as NOR and a AOI (AND-OR-INV) gate. The logic block can implement all the 2-input functions and some 3-input functions. All the metal layers in VPEX are fixed and the logic block is configured by the via layer between M1 and M2. The routing is done using M3 and M4 layers and the via layer between M3 and M4 is used to configure the routing fabric. Figure 2.7 illustrates the logic and interconnect architecture for VPEX. The architecture is evaluated against a standard cell implementation for small circuits such as a full adder and a 4-bit multiplier.

Finally, Chau et al. have proposed a via-programmable logic cell called CULG [46]. The CULG consists of two complementary NMOS pull-down networks, two cross-coupled PMOS transistors, and two inverters. The logic block can implement all 3-input functions and some 4 or 5 input functions. The logic block is configured using the via between M3 and M4. The performance of CULG is evaluated against a transmission gate (TG) based logic block and a differential cascode voltage switch with pass gate (DCVSPG) logic block. CULG requires fewer transistors than TG and DCVSPG to implement lookup tables with 3 or

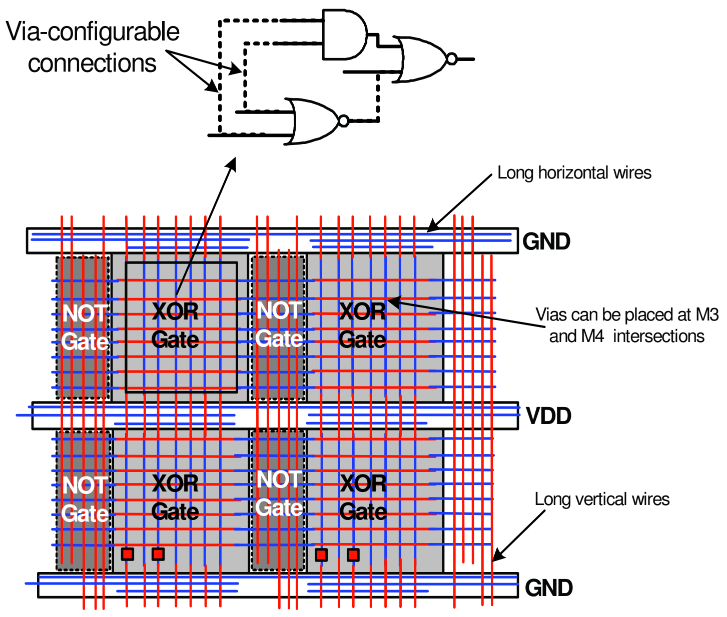


Figure 2.7: Logic and interconnect architecture for VPEX [31]

more inputs. The power consumption of CULG is shown to be better than TG and DCVSPG, but the delay is worse than DCVSPG. CULG was evaluated using small circuits such as full adder, 8-bit multiplier, flip-flop, and a 3-input NAND. It is not clear whether CULG was intended for MPSAs or VPSAs, since the authors do not describe the interconnect architecture in detail.

Our work is different in that all of these previous efforts focus on point solutions. They consider a certain type of logic block with a routing fabric that has a fixed amount of configurability, and compare it against standard cell implementation or another architecture. In our work, we do not consider a fixed amount of customization. We vary the customization and study the effect it has on the overall performance of a Structured ASIC.

2.3.2 Commercial Efforts

There are a number of commercial vendors who have offered Structured ASIC products, including both MPSAs and VPSAs. Some of these products provide a migration path for existing FPGA designs to improve power dissipation and unit cost while others are designed for general System-On-a-Chip (SoC) based designs. Table 2.2 shows several of these products. Unfortunately, detailed information about most of these products is not published. We have categorized these products into three classes: products that are currently available at the current process node (Active), products that can be seen at the company's web-page but are not offered in the latest process technology (Semi-Active), and products that have been completely discontinued (Defunct). It can be seen that most of the commercial products are MPSAs, whereas the academic work has focussed mainly on VPSAs. The commercial products have a wide range of configurability that varies from single-via to six-metal and six-vias. Interestingly, the products that had a high amount of configurability have been discontinued.

2.4 Yield Modeling

Noise or defects in the IC manufacturing process lead to chips that do not work. The percentage of fault-free chips produced by a given fabrication process is known as *yield*. Depending on the nature of the defects, the yield can be categorized into two classes: *random defect yield*, and *systematic yield*.

The random defect yield is due to the defects in the manufacturing materials or unwanted chemical and airborne particles being deposited during the various

Table 2.2: Commercial Structured ASICs

State	Company	Product	Type	Custom Layers (M: metal, V: via)	Comments
Active	Altera	Hardcopy Series	MPSA	2M	Aimed at FPGA-to-ASIC conversion [1]
	eASIC	Nextreme Series	VPSA	1V	4-6 week turnaround time [4]
Semi-Active	ChipX	CX6200	MPSA	2–4M	[2]
	Faraday	MPCA	MPSA	3M + 2V	Targets SoC based communication systems [5]
	ON Semiconductor	Xpress Array-II	MPSA	?	150nm Structured ASIC aimed at FPGA-to-ASIC migration [14]; formerly AMI Semiconductor
	ViASIC	ViaMask, DuoMask	VPSA	1–2V	Targets SoC systems [21]
	Virage Logic	ASAP	MPSA	3–4M	Metal programmable cell libraries [22]
Defunct	Fujitsu	AccelArray	MPSA	3–4M	[7]
	Lightspeed	-	MPSA	2M+2V to 6M+6V	[11]
	LSI Logic	RapidChip	MPSA	all-M + all-V	Only the transistors are fabricated, all the remaining layers can be customised [12]
	NEC	ISSP	MPSA	2M	Targets SoC systems [89]
	Tier Logic	-	-	-	Essentially a 3D FPGA in which the configuration memory was placed on top of the device. The device could be converted into an ASIC by replacing the configuration memory with a metal layer [19]

steps of IC fabrication [72]. These defects are usually local and are random in nature. It has been shown that a negative binomial distribution provides a good fit for these random defects [110]. For a defect i, if λ_i and α_i denote number of faults per chip and the spread of the defect respectively, then the yield due to defect i, Y_i, can be expressed as [110]:

$$Y_i = \left(1 + \frac{\lambda_i}{\alpha_i}\right)^{-\alpha_i} \tag{2.1}$$

The total random defect yield, Y_R, can then be expressed as [110]:

$$\begin{aligned} Y_R &= \prod_{i=1}^{m} \left(1 + \frac{\lambda_i}{\alpha_i}\right)^{-\alpha_i} \\ &\simeq \left(1 + \frac{\lambda}{\alpha}\right)^{-\alpha} \end{aligned} \tag{2.2}$$

where α is defined as the clustering parameter and λ is the average number of faults per chip. The clustering parameter is a measure of the degree to which the defects are clustered together. The average number of faults per chip is related to the area of the chip (A_{die}) and the defect density (D_o); it can be modeled as:

$$\lambda = A_{die} \times D_o \tag{2.3}$$

Systematic yield occurs due to systematic defects. The systematic defects involve processing problems such as scratches on wafers due to mishandling, mask misalignments, over- and under-etching, excessive variation in temperatures etc. [72, 110]. These defects are large, and result in parts of wafers, or entire wafers to be faulty. The systematic yield is usually modeled by a constant factor that is multiplied with the random defect yield to obtain total yield [32, 72]. The systematic yield improves as the process matures.

If Y_o represents systematic yield, the total die-yield Y_{die} can be expressed as:

$$Y_{die} = Y_o \times \left(1 + \frac{A_{die} \times D_o}{\alpha}\right)^{-\alpha} \tag{2.4}$$

2.5 CAD for FPGAs and CBICs

Structured ASICs have many similarities to FPGAs and CBICs. The CAD for Structured ASICs would therefore also share some of the techniques from FPGA and CBIC CAD. In this section, we review three of the most important CAD stages; namely Placement, Routing, and Whitespace Insertion. We highlight the important differences between FPGAs and CBICs for these steps and review the techniques used in some of the open-source CAD tools.

2.5.1 Placement

The placement problem involves assigning physical locations to the logic blocks while minimizing certain objectives such as wirelength, delay etc. In CBICs, the logic blocks are fine-grained basic cells that have the same height (hence, called standard cells) and are arranged in rows. The cells have different widths, depending on the output drive strength and the logic function. A standard cell can be placed anywhere within the row. In contrast, the FPGA logic blocks are clustered

LUTs that are arranged in a grid on the FPGA die. As a result, there are two important differences between the FPGA and CBIC placement problems. First, the number of placeable blocks in CBICs is much larger compared to FPGAs. Second, the fixed grid locations in an FPGA make the placement more constrained. In addition to the basic logic blocks, large macro blocks such as embedded memories and other IP blocks for datapath or DSP operations are included both in FPGAs and CBICs which makes the placement problem more challenging.

The most commonly used academic placement tool for FPGAs is VPR [34, 35]. The placement algorithm in VPR is based on simulated annealing [70]. The annealing schedule is adaptive in that the initial temperature, exit criterion, and the temperature update scheme are calculated from circuit parameters. The cost function is based on total wirelength and timing cost.

There are a number of available CBIC placers [44, 50, 81, 99, 111, 118]. These placers incorporate different types of algorithms to perform the placement while optimizing for wirelength. The techniques used in most of these placers are described in [56]. We used CAPO [99] in our experiments. CAPO combines recursive min-cut bi-partitioning and floorplanning to perform placements for mixed-size net-lists. Depending on the size of the partitioning instance, three different min-cut partitioners are used: (1) an optimal branch-and-bound partitioner for small partitioning instances [39]; (2) a heuristic based, middle range Fiduccia-Mattheyses partitioner [40]; and (3) a multilevel Fiduccia-Mattheyses partitioner [41]. To handle large-sized IP blocks during placement, CAPO uses a fixed outline floorplanner called Parquet [26]. When the size of the partitioning bin is close to the size of a macro block, Parquet is invoked instead of partitioning.

2.5.2 Routing

The routing problem involves making non-overlapping connections between the placed logic blocks while minimizing the wirelength. Additional objectives such as delay and/or power minimization can also be included in the optimization. In CBICs, routing is typically divided in two steps. The first step finds approximate routes, or regions through which the route for each net would pass. This step is known as *global routing*. It limits the search space for the next step, known as *detailed routing*. The detailed routing step accounts for the complex design rules and assigns metal tracks for each net inside a global route. It also makes connections to the logic block inputs and outputs.

On the other hand, in FPGAs all the wires are prefabricated and the connections between the logic blocks are made simply by configuring the switch-boxes and the connection-boxes. Because the switch-boxes and the connection-boxes have limited connectivity, it is difficult to correctly model the routing capacities for a global router. Hence, the routing process in FPGAs is typically a single step, detailed routing.

VPR is the most widely used academic FPGA routing tool. It uses A^* maze-search [55] along with PathFinder algorithm [83] to remove congestion. The PathFinder algorithm is an iterative rip-up and re-route technique that starts with shortest possible routes allowing overuse of routing resources, and then gradually increases the congestion penalty for each routing iteration. The cost function used in the PathFinder algorithm is:

$$c_n = (b_n + h_n) * p_n \tag{2.5}$$

where c_n is the congestion cost for a routing resource n, b_n is the base cost, h_n reflects congestion history, and p_n is the congestion penalty for current congestion. VPR uses the following slightly modified form of the cost function:

$$c_n = b_n * h_n * p_n \tag{2.6}$$

This is done to avoid normalization of b_n and h_n to the same range. VPR routes multi-sink nets by finding the path for each sink separately. The route to the first sink is found using A^* search; then for each additional sink, A^* search is performed around the currently routed portion.

In CBICs, the success and quality of final routes is mostly determined by global routing [98]. The main goal of detailed routing is to satisfy the design rules of the process. Thus, despite some of the research efforts on detailed routing [33, 45, 62, 114], most of the recent research on CBIC routing has focused on the global routing problem [54, 58, 86, 91, 98]. The global router we use in our experiments is known as FGR [98]. FGR builds a Rectilinear Minimal Spanning Tree (RMST) or a Rectilinear Steiner Minimal Tree (RSMT) for each net and then rips-up and re-routes congested, two-point portions of these trees. The technique used to remove congestion is very similar to the PathFinder algorithm, and utilizes the following cost function:

$$c_n = b_n + (h_n * p_n) \tag{2.7}$$

2.5.3 Whitespace Insertion

Producing placements in which the logic blocks are very tightly packed often makes the design un-routable. It is therefore desirable to leave some unused space in the placement. This is known as *whitespace*. In CBICs, there are several other reasons for including whitespace [27]:

1. To reach timing closure, some of the placed gates may need to be resized, some of the nets might require additional buffering resources, or some local portion of the circuit might need to be re-synthesized
2. Tightly packing gates in a given region can violate power density and temperature constraints
3. It may be desirable to use a previously designed and rigorously simulated power grid. Such a power grid would determine the die size, which may be larger than the area needed by the logic

The whitespace insertion process has been studied previously, both for CBICs [27, 76, 100, 105] and for FPGAs [53, 106, 112, 113]. However, it still remains an active area of research [9].

There are different approaches to handle whitespace insertion in CBICs. Cells in the congested regions can be artificially bloated [105]; congestion estimates can be used to change partition sizes in min-cut partitioning [76]; unconnected *fake-cells* can be added to the net-list to handle large amounts of whitespace [27]; or, the available whitespace can be recursively divided as part of min-cut partitioning to uniformly distribute it [27]. CAPO, the placer we use in our experiments, performs whitespace insertion by using fake cells and recursive division of available whitespace during min-cut partitioning.

In CBICs, the whitespace can be inserted anywhere within a standard-cell row, but in FPGAs it must be inserted at fixed grid locations, and in units of logic blocks. This makes the problem particularly hard for FPGAs. Inserting whitespace by leaving entirely empty logic blocks, requires a very accurate congestion estimate. The only successful approach to insert whitespace in this fashion is [106], which used a router in the inner most loop of simulated annealing. This makes this approach impractically slow. Most of the other FPGA techniques achieve whitespace insertion at the clustering stage by *de-populating* the logic blocks [53, 112, 113]. In these techniques, the logic block is not fully packed and some of its inputs and outputs remain unused. This helps to reduce the routing demand in the congested regions.

Chapter 3

Cost Model for Structured ASICs

3.1 Overview

In this chapter we describe a cost model to estimate the manufacturing cost for a Structured ASIC die. The model expresses the cost as a function of die-area and the number of routing layers that can be configured by one or more metal and/or via masks. We identify different parameters that affect the cost and then derive detailed equations in terms of these parameters. We then use typical values of these parameters to assess per-die cost of Structured ASICs.

As described in Section 2.2.3, a Structured ASIC can either be configured by metal-and-via layers (MPSA), or only by via layers (VPSA). We provide cost equations for both MPSAs and VPSAs and study how MPSAs and VPSAs compare in terms of cost.

3.2 Factors Affecting Structured ASIC Cost

In most FPGA and ASIC research, it is common to use die area as a proxy to the cost of a device. However, this is not sufficient for Structured ASICs. In Structured ASICs, the cost per die depends on the number of routing layers in addition to the die area; a larger die with fewer routing layers can be less expensive than a smaller die with more configurable layers.

There are a number of other factors that affect the cost of Structured ASICs. These factors are described below.

3.2.1 Die-Yield

The number of *good* dies obtained from a wafer depends on the yield of the fabrication process. A high-yield process would produce more working chips. Since the cost to process a wafer is fixed, high yield lowers the cost per chip and vice versa.

3.2.2 Device Volume Requirements

The volume in which a Structured ASIC device is produced is a very important factor that affects the per-die cost. There are two types of volumes associated with a Structured ASIC device. One is the total device volume, which is the number of partially fabricated, un-configured Structured ASIC devices of a particular family. These Structured ASICs can be used for implementing many different designs. The other type of volume is the per-customer volume, which is the number of Structured ASICs that have been configured to implement one particular design for a given customer.

3.2.3 Mask-set Cost

The cost of producing the masks for each layer of the Structured ASIC device impacts the die-cost. The mask-set cost can be divided into two components. One component is for the masks of layers that are fixed; this component of cost is amortized over the total device volume (total among all customers). The other component deals with preparing masks that are configurable; this component is amortized over the volume per customer.

3.2.4 Processing Costs

The cost to process a wafer is another component that affects the die-cost. Like mask-set cost, this cost has two components: one for processing the wafer up to the first configurable routing layer, and the other for processing the remaining portion of the device.

In the following section, we describe a detailed cost model that uses these factors to express Structured ASIC cost as a function of die-area and number of routing layers.

3.3 Die-cost Formulation

The cost per die, C_{die}, can be expressed as:

$$C_{die} = C_{base} + C_{custom} + C_{proto} + C_{pkg} + C_{test} \tag{3.1}$$

where C_{base} is the cost of the partially fabricated device (i.e., the cost shared across all the customers), C_{custom} is the cost to customize the pre-fabricated chip to implement a particular circuit, C_{proto} is the prototyping cost to manufacture test wafers before the final spin, C_{pkg} is the packaging cost, and C_{test} is the testing cost.

In this dissertation, our main goal is to express C_{die} as a function of number of routing layers and die area. In this regard, C_{pkg} and C_{test} are independent of user's design and do not vary when we consider a range of different Structured ASIC architectures. Therefore, we consider them as constant and do not use them in our cost calculations.

The base, customization, and prototyping costs can be subdivided into three components:

1. a non-recurring cost of preparing the mask sets;

2. the cost associated with processing a wafer; and,

3. the cost of setting up the fab line.

In the following subsections, we provide equations for C_{base}, C_{custom}, and C_{proto}.

3.3.1 Base Cost

The base cost (C_{base}) is associated with the portion of Structured ASIC that will not be customized. It includes costs of preparing masks for fixed layers and processing the wafers up to, but not including, the first configurable layer. This cost is amortized over the total volume of all the different customers. C_{base} can be expressed as:

$$C_{base} = \underbrace{\left(\frac{C_{sm_l}N_{fm_l} + C_{sm_u}N'_{up\_fix}}{V_{tot}}\right)}_{\text{Mask costs}} + \underbrace{\left(\frac{C_{wpm}N'_{fab1} + C_{sw}}{N_{gdpw}}\right)}_{\text{Wafer costs}} + \underbrace{\left(\frac{C_{fs_1}}{V_{tot}}\right)}_{\text{Fab setup cost}} \tag{3.2}$$

where N_{fm_l} is the number of lower fixed masks, N'_{up_fix} is the number of fixed upper-level masks, C_{sm_l} is the average cost for a single lower-level mask (e.g., polysilicon mask, M1 mask), C_{sm_u} is the average cost for a single upper-level mask (e.g., M4 mask), V_{tot} is the expected total volume, C_{wpm} is the wafer processing cost for a single mask, N'_{fab1} is the number of layers up to which the wafer has been processed, C_{sw} is cost of single unprocessed wafer, N_{gdpw} is the number of good dies per wafer, and C_{fs_1} is the fab setup cost of the Structured ASIC device for all customers.

The fixed upper-level masks (N'_{up_fix}) can include fixed masks that are required for power grid etc. It can also include masks for some of the routing layers that are fixed in the case of VPSAs.

Lower-level masks (e.g., diffusion and polysilicon masks) have higher costs than higher-level masks. Acquiring cost values for each individual mask is not always possible and using a different cost value for every mask would also complicate the cost model. Therefore, we decided to use one average cost number for lower masks, C_{sm_l}, and a lower average cost value for higher masks, C_{sm_u}.

3.3.2 Customization Cost

The customization cost (C_{custom}) is the cost that is associated with the portion of a Structured ASIC that is customized for a particular application. It includes costs for preparing the custom masks and processing the wafer from the first configurable layer onwards. C_{custom} can be calculated as:

$$C_{custom} = \underbrace{\left(\frac{C_{sm_u} N'_{config}}{V_c}\right)}_{\text{Mask costs}} + \underbrace{\left(\frac{C_{wpm} N'_{fab2}}{N_{gdpw}}\right)}_{\text{Wafer costs}} + \underbrace{\left(\frac{C_{fs_2}}{V_c}\right)}_{\text{Fab setup cost}} \tag{3.3}$$

where N'_{config} is the number of custom masks needed to configure the device, V_c is the volume per customer, N'_{fab2} is the number of layers that need to be processed during the second, customer-specific phase of fabrication, and C_{fs_2} is the fab setup cost for customization phase.

3.3.3 Prototyping Cost

Due to the complexity of large hardware designs, it is usually necessary to manufacture a number of spins, where each spin requires a new set of custom masks. Assuming N_s is the total number of customer silicon spins including the final version and C_{fs_3} is the fab setup cost for prototyping phase, the prototyping costs (excluding the final spin) are calculated as:

$$C_{proto} = \underbrace{(N_s - 1)\left(\frac{C_{sm_u} N'_{config}}{V_c}\right)}_{\text{Mask costs}} + \underbrace{\left(\frac{N_s - 1}{V_c}\right)\left(C_{wpm}\left(N'_{fab1} + N'_{fab2}\right) + C_{sw}\right)}_{\text{Wafer cost}}$$
$$+ \underbrace{(N_s - 1)\left(\frac{C_{fs_3}}{V_c}\right)}_{\text{Fab setup cost}} \tag{3.4}$$

In C_{proto}, we include the cost to manufacture one complete wafer for every prototype spin, excluding the final spin. Although minimum lot sizes required by the foundry may require several wafers to be manufactured at once, a Structured ASIC vendor should be able to mix wafers from several customers to fill a single lot. Furthermore, a Structured ASIC vendor may offer a multi-project wafer, where each customer uses less than a full wafer. This could reduce the wafer cost component of the prototype even further than assumed here.

3.3.4 Good Dies per Wafer (N_{gdpw})

The number of good-dies-per-wafer (N_{gdpw}) depends on number of dies per wafer (N_{dpw}) and die yield (Y_{die}). It can be expressed as:

$$N_{gdpw} = N_{dpw} \times Y_{die} \tag{3.5}$$

The number of dies per wafer (N_{dpw}) depends on the area taken up by a die on the silicon wafer. There are three types of areas associated with each die; namely, *core area*, *pad area*, and *scribe area*. The core area is the area of the Structured ASIC fabric. The pad area surrounds the core area with input and output pads. The scribe area is a ring around the die reserved for wafer testing and die-cutting; it mostly influences the area of very small dies. The three components of die-area are illustrated in Figure 3.1.

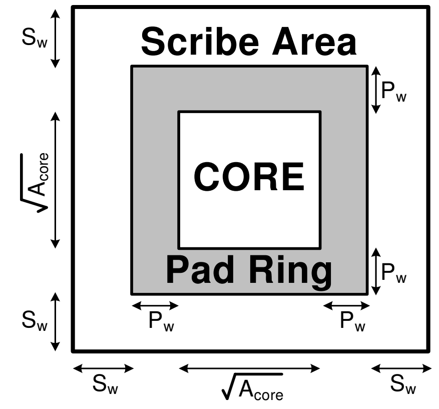


Figure 3.1: Core, pad and scribe area

If P_w and S_w represent the pad width and scribe width, respectively, then the pad area (A_{io}) and scribe area (A_{scribe}) can be calculated as:

$$A_{io} = 4P_w\left(P_w + \sqrt{A_{core}}\right)$$

$$A_{scribe} = 4S_w\left(S_w + 2P_w + \sqrt{A_{core}}\right)$$

The number of dies per wafer can then be approximated as [61]:

$$N_{dpw} = \frac{\pi \times \left(\frac{D_{waf}}{2}\right)^2}{A_{core} + A_{io} + A_{scribe}} - \frac{\pi \times D_{waf}}{\sqrt{2\left(A_{core} + A_{io} + A_{scribe}\right)}} \tag{3.6}$$

where A_{core} is the core area, and D_{waf} is the wafer diameter. The first term in this equation counts the number of dies that can fit a diameter of D_{waf}, and the second term subtracts the clipped dies that fall along the circumference.

We use Eq. 2.4 to calculate die-yield:

$$Y_{die} = Y_0 \times \left(1 + \frac{(A_{core} + A_{io}) \times D_0}{\alpha}\right)^{-\alpha} \tag{3.7}$$

where Y_0 is the multiplier to account for material and systematic yield, D_0 is the defect density, and α is the cluster factor.

The yield may be affected by the number of routing layers. It is possible that every additional layer may reduce the yield. On the other hand, the regularity in fixed layers of Structured ASICs can help to improve the yield. It is not known which of these conflicting effects would be significant. We currently assume both of these to have negligible effect on Y_{die}.

3.3.5 Die-Cost Equation

Most of the parameters in Eqs. 3.2 to 3.4 are specific to a particular manufacturing process. The only architecture-dependent parameters are N'_{up_fix}, N'_{config}, N'_{fab1}, and N'_{fab2}. These parameters are either constants or depend on the number of routing layers and the number of custom masks. Table 3.1 describes these parameters.

We define N_{ml} as the number of metal layers used for routing and use it as a measure for the number of routing layers in a Structured ASIC architecture. These

Table 3.1: Architecture parameters used in the cost model

Parameter	Description
N'_{up_fix}	Number of fixed upper level masks
N'_{config}	Number of configurable masks
N'_{fab1}	Number of layers fabricated in phase 1 (pre-fabrication)
N'_{fab2}	Number of layers fabricated in phase 2 (customization)

metal layers can be configured by one or more via, or metal-and-via masks.

By substituting the values of C_{base}, C_{custom}, C_{proto} from Eqs. 3.2 to 3.4 into Eq. 3.1, C_{die} can be expressed in the following form:

$$C_{die} = \frac{K_0}{N_{gdpw}} + N_{ml}\left(\frac{K_1}{N_{gdpw}} + K_2\right) + K_3 + C_{pkg} + C_{test} \tag{3.8}$$

where K_0, K_1, K_2, and K_3 are constants that depend on the volume requirements, Structured ASIC types, and various foundry costs.

Most of the constant parameters in Eqs. 3.2–3.4, 3.6, and 3.7 are confidential information of a foundry. The cost numbers (such as C_{sm_l}, C_{sm_u} and C_{wpm}) can also vary from one foundry to another. The values used for these parameters in our experiments are shown in Table 3.2. We obtained and confirmed this data from various sources, including several news articles and contacts in industry. In Chapter 5, we provide a detailed sensitivity analysis of the die-cost model to various parameters of Table 3.2.

Table 3.2: Values of parameters used in the cost model

Param.	**Value**	**Comments**
N_{fm_l}	18	*Fixed masks below the configurable masks* (1) A 10-metal, 90nm process requires 34 masks [90]; we assume 45nm also requires 34 masks[a] (2) Device fabricated up to M2, and subsequent layers require single mask
C_{sm_l}	\$107k	*Average single mask cost for lower layers* (1) 45nm mask set costs \$2.5M [6] (2) Lower-level mask cost = 3× Upper-level mask cost
C_{sm_u}	\$36k	*Average single mask cost for upper layers* [90]
V_{tot}	2M	*Total volume*
C_{wpm}	\$220	*Wafer processing cost per mask* Cost to process a 45nm wafer = \$8000; Total masks= 34
D_{waf}	300mm	*Wafer diameter*
P_w	150μm	*Pad width*
S_w	100μm	*Scribe width*
N_s	2	*Number of silicon spins* One prototype plus one re-spin
V_c	100k	*Per customer volume*
N_{fm_u}	2	*Fixed masks above the configurable masks (e.g., for power grid)*
Y_0	0.9	*Material and systematic yield* [10]
N_m, N_v	1	*Number of masks per routing layer* One mask each for metal and via
D_0	1395 per m^2	*Defect rate* [10]
α	2.0	*Cluster factor* [10]
$\frac{C_{fs_1}}{V_{tot}}, \frac{C_{fs_2}}{V_c}, \frac{C_{fs_3}}{V_c}$	\$0	C_{fs_1}, C_{fs_2}, C_{fs_3}*: Fab line setup costs* These can be ignored, esp. when divided over the volume
C_{sw}	\$0	*Cost of a single, unprocessed wafer* Assumption: Cost of an unprocessed wafer is negligible compared to the processing cost
C_{pkg}, C_{test}	\$0	*Packaging cost, Testing cost* Not considered because these are independent of die area and N_{rl}

[a] Even with N_{fm_l} as high as 36 (52 total masks), the results are not significantly different.

3.4 Cost Modeling for Different Structured ASICs

In this section, we derive equations for K_0–K_3 for different types of Structured ASICs and compare their cost trends.

3.4.1 MPSA Die-Cost

In MPSAs, the fixed upper masks are usually the masks that define the power grid; let N_{fm_u} denote the number of such masks. The routing is done using using one or more upper-level masks for each configurable metal and via layer; let N_m and N_v denote the number of masks needed for each metal and via layer respectively, and N_{ml} denote the number of metal layers as described in Section 3.3.5. Finally, the MPSA device can only be fabricated up to, and not including, the first configurable layer. Thus, for MPSAs, the architecture-dependent parameters N'_{up_fix}, N'_{config}, N'_{fab1}, and N'_{fab2} can be written as:

$$N'_{up\_fix} = N_{fm_u}$$

$$N'_{config} = N_{ml} \times (N_m + N_v)$$

$$N'_{fab1} = N_{fm_l}$$

$$N'_{fab2} = N_{ml} \times (N_m + N_v) + N_{fm_u}$$

Using the above values, the constants K_0–K_3 in Eq. 3.8, can be expressed as:

$$\begin{aligned}
K_0 =& C_{wpm}(N_{fm_l} + N_{fm_u}) + C_{sw} \\
K_1 =& C_{wpm}\,(N_m + N_v) \\
K_2 =& \frac{N_s C_{sm_u}\,(N_m + N_v)}{V_c} + \left(\frac{N_s - 1}{V_c}\right) C_{wpm}\,(N_m + N_v) \\
K_3 =& \frac{C_{sm_l} N_{fm_l} + C_{sm_u} N_{fm_u}}{V_{tot}} + \left(\frac{N_s - 1}{V_c}\right)\left(C_{wpm}(N_{fm_l} + N_{fm_u}) + C_{sw}\right) \\
& + \frac{C_{fs_1}}{V_{tot}} + \frac{C_{fs_2} + (N_s - 1) C_{fs_3}}{V_c}
\end{aligned}$$

3.4.2 VPSA Die-Cost

In VPSAs, all the metal-masks are fixed and only the via-masks need to be generated for a particular customer. In the VPSA architectures that we consider in our experiments, we assume that the via layer that connects the logic block inputs and outputs to the first routing layer is also fixed. This allows the device to be fabricated up to the first routing layer. The values of the architecture-dependent parameters for VPSAs are given as:

$$N'_{up\_fix} = N_{ml}N_m + N_v + N_{fm_u}$$

$$N'_{config} = N_v\left(N_{ml} - 1\right)$$

$$N'_{fab1} = N_{fm_l} + N_m + N_v$$

$$N'_{fab2} = \left(N_{ml} - 1\right)\left(N_m + N_v\right) + N_{fm_u}$$

The constants K_0–K_3 in Eq. 3.8 can then be written as:

$$\left.\begin{aligned} K_0 =& C_{wpm}(N_{fm_l} + N_{fm_u}) + C_{sw} \\ K_1 =& C_{wpm}(N_m + N_v) \end{aligned}\right\} \quad \text{Same as MPSA}$$

$$\begin{aligned} K_2 =& \frac{C_{sm_u}N_m}{V_{tot}} + \frac{N_s C_{sm_u} N_v}{V_c} + \left(\frac{N_s - 1}{V_c}\right) C_{wpm}(N_m + N_v) \\ K_3 =& \frac{C_{sm_l}N_{fm_l} + C_{sm_u}(N_v + N_{fm_u})}{V_{tot}} - \frac{N_s C_{sm_u} N_v}{V_c} \\ &+ \left(\frac{N_s - 1}{V_c}\right)\left(C_{wpm}(N_{fm_l} + N_{fm_u}) + C_{sw}\right) \\ &+ \frac{C_{fs_1}}{V_{tot}} + \frac{C_{fs_2} + (N_s - 1)C_{fs_3}}{V_c} \end{aligned}$$

3.4.3 Die-Cost for VPSAs with Single-Via Configurability

It is possible that the routing fabric of a VPSAs can be configured by only a single via layer. In this case, only the mask(s) for a single via layer need to be generated to configure the device. The location of the configurable via layer determines what proportion of the device can be pre-fabricated.

Depending on the routing fabric architecture, the die-cost can also be affected by the number of metal layers used for routing. The cost to process a wafer increases with the number of routing metal layers. For example, although there is only one configurable via layer, each metal layer on top used for routing requires processing and must be paid for.

In the single-via configurable fabric used in our experiments, the via layer above the first routing metal layer is configurable. The values of the architecture dependent parameters are:

$$N'_{up\_fix} = N_{ml}N_m + (N_{ml}-1)\,N_v + N_{fm_u}$$

$$N'_{config} = N_v$$

$$N'_{fab1} = N_{fm_l} + N_m + N_v$$

$$N'_{fab2} = (N_{ml}-1)\,(N_m + N_v) + N_{fm_u}$$

The values of K_0–K_3 in Eq. 3.8 are as follows.

$$\left.\begin{aligned} K_0 &= C_{wpm}(N_{fm_l} + N_{fm_u}) + C_{sw} \\ \\ K_1 &= C_{wpm}(N_m + N_v) \end{aligned}\right\} \text{Same as MPSA}$$

$$K_2 = \frac{C_{sm_u}(N_m + N_v)}{V_{tot}} + \left(\frac{N_s - 1}{V_c}\right) C_{wpm}(N_m + N_v)$$

$$\begin{aligned} K_3 = &\frac{C_{sm_l}N_{fm_l} + C_{sm_u}(N_{fm_u} - N_v)}{V_{tot}} + \frac{N_s C_{sm_u} N_v}{V_c} \\ &+ \left(\frac{N_s - 1}{V_c}\right)\left(C_{wpm}(N_{fm_l} + N_{fm_u}) + C_{sw}\right) \\ &+ \frac{C_{fs_1}}{V_{tot}} + \frac{C_{fs_2} + (N_s - 1)C_{fs_3}}{V_c} \end{aligned}$$

3.4.4 MPSA and VPSA Cost Trends

Using the parameter values from Table 3.2, the typical values of constants K_0–K_3 to be used with Eq. 3.8 for different types of Structured ASICs are shown in Table 3.3.

Table 3.3: Typical values for the cost model constants

Type	K_0	K_1	K_2	K_3
MPSA	$4400	$440	$1.4444	$1.0430
VPSA			$0.7424	$0.341
VPSA-SV			$0.0404	$1.745

The impact of the differences of these constants on die-cost is illustrated in Figs. 3.2 and 3.3, where the output of the cost model is shown for a range of values of A_{core} and N_{ml}. The figure shows that a considerable reduction in die-

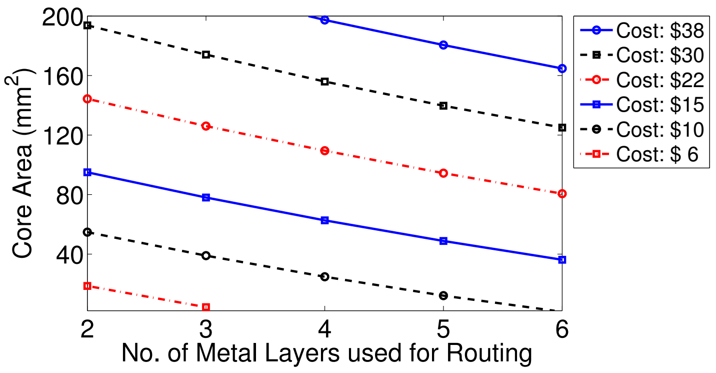
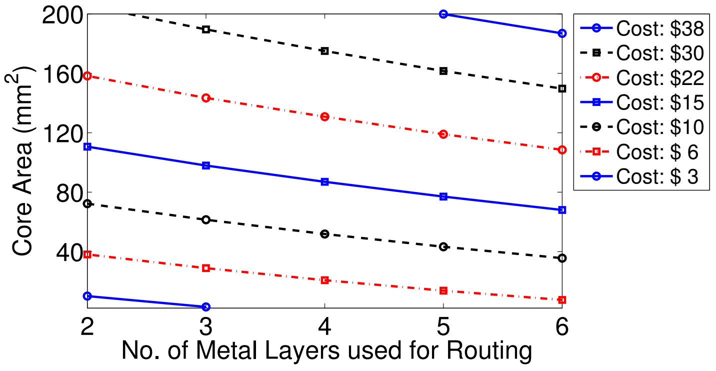
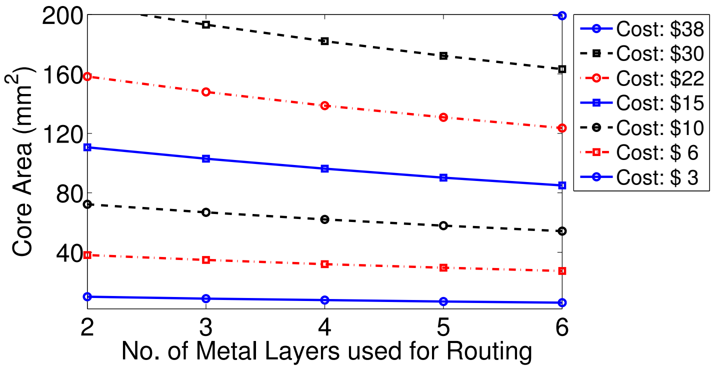


Figure 3.2: Cost trends for MPSAs



(a) Custom vias between routing layers



(b) Single custom-via layer

Figure 3.3: Cost trends for VPSAs

area is required for an additional routing layer to be cost effective, especially for MPSAs and for VPSAs with all custom vias. For example, to maintain a constant cost of \$15, each additional metal routing layer in a MPSA and a VPSA must save $15mm^2$ and $10mm^2$ of die-area, respectively. This is because of the large mask and wafer processing costs associated with each additional layer. However, it is less expensive (in terms of die-area) to add an extra metal routing layer in VPSAs with single-via configurability, where a reduction of only $6mm^2$ per additional layer is needed to maintain a constant cost of \$15. This is because VPSAs with single custom-via layer amortize the cost across the much larger total device volume (V_{tot}) rather than the per-customer volume (V_c).

The die-yield and die-cost of CBICs and different types of Structured ASICs is shown in Figure 3.4 for $100k$ dies. In calculating the CBIC cost, we assumed six routing layers and every mask to be custom. We also assume that a CBIC requires a re-spin where all the CBIC masks are changed, whereas a Structured ASIC requires a re-spin where only the configurable masks (N'_{config}) are changed. It is possible that a CBIC re-spin can be completed without modifying all the masks by employing some of the Engineering Change Order (ECO) techniques [59]. However, we do not take this into account.

In Fig. 3.4, it is better to compare the area values of Structured ASICs and CBICs for a given cost, rather than comparing the cost values for a given area since the areas will generally be different for a given design. As an example, at a cost of \$45, a CBIC can use only $10mm^2$, whereas a Structured ASIC implementation can use more than $200mm^2$. However, this difference becomes smaller as the die-cost

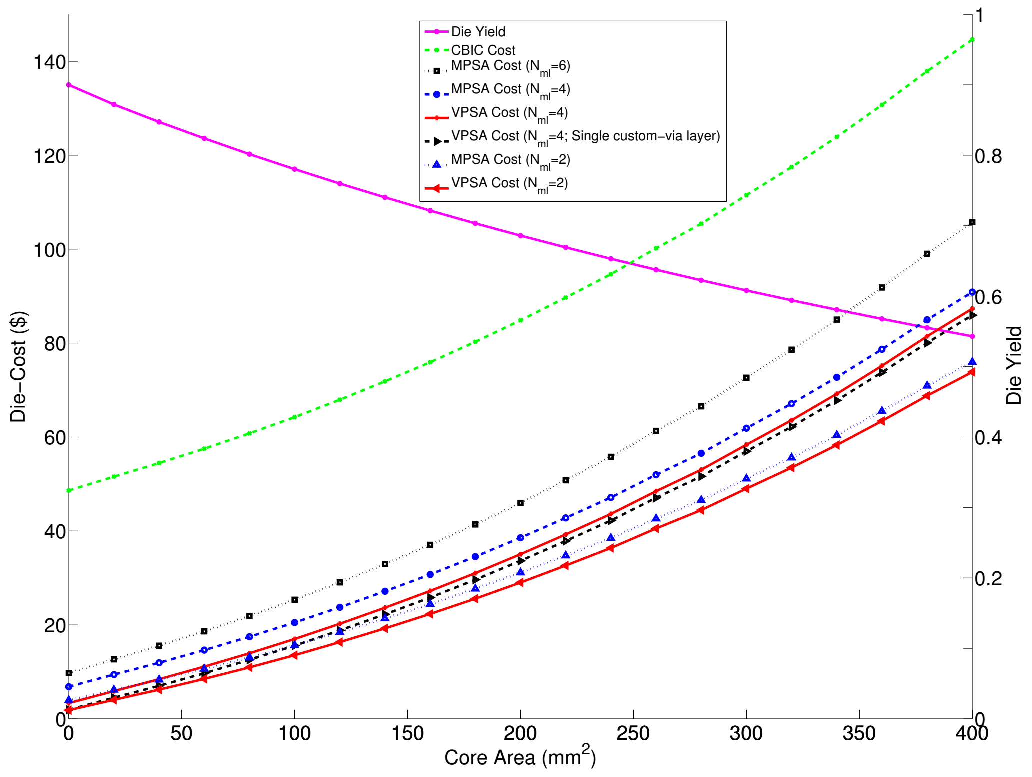


Figure 3.4: Yield and die-cost (total CBIC volume = V_c)

increases. Thus, for large die-sizes, Structured ASICs must be very area efficient to compete with CBICs.

3.5 Summary

In this chapter, we described the Structured ASIC die-cost model. The cost model expresses the die-cost as a function of the number of configurable routing layers and die-area of Structured ASICs. We used the cost model to study the die-cost trends for MPSAs and two different types of VPSAs for a range of configurable layers and die-area values. Using specific assumptions about volume (100*k* units), we have shown that VPSAs with single-via configurability are most cost effective per mm^2 and CBICs are least cost effective. Although not shown here, we have also studied the sensitivity of die-cost to various cost model parameters (Table 3.2). This analysis is presented in Chapter 5.

Chapter 4

Framework and CAD

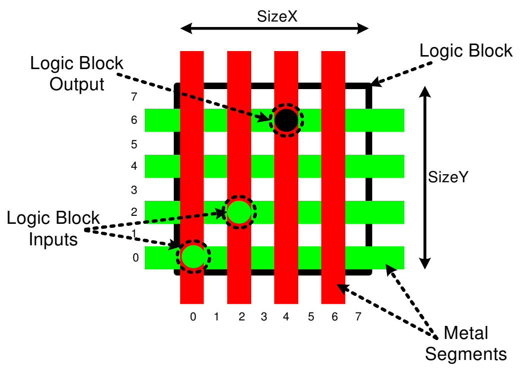
4.1 Overview

This chapter describes the experimental framework used to conduct the experiments for this thesis. We start by describing how the logic and interconnect fabrics are modeled. This defines the architecture space that our experiments can cover. This is followed by a description of the metrics used to evaluate potential architectures. Finally, we explain the CAD flow used in the experiments. We describe the placement and routing steps and explain the differences in the CAD flow for MPSAs and VPSAs.

4.2 Architecture Modeling

4.2.1 Logic Fabric

As described in Sec. 2.2.3, the Structured ASIC logic blocks can vary in granularity. This affects their physical size and the number of inputs and outputs. In our framework, we model each logic block as a rectangular region with a given number of pins. The logic block size (height and width) and the position of pins are specified in terms of wire half-pitches. This high-level abstraction allows us to model a large architecture space without worrying about the low-level, layout-related details. The modeling process for a simple 2-input logic block is illustrated in Figure 4.1.



```
BLK MODEL
{
   SizeX: 8
   SizeY: 8
   Inputs: 2
   InLoc: (0,0) (2,2)
   Outputs: 1
   OutLoc: (4,6)
}
```

Figure 4.1: Logic block modeling

4.2.2 Routing Fabric

The routing fabric in a Structured ASIC can be metal-and-via programmable (MPSA), or via-only programmable (VPSA). Each routing layer in an MPSA is modeled as a set of equally-wide and equally-spaced horizontal or vertical wires. We use the minimum width and minimum spacing possible in our technology to calculate the total number of wires that can pass through a given region. This number is used to determine the routing capacities for the global router.

The VPSAs are modeled by first identifying a basic routing tile. The basic routing tile is the smallest unit of the routing fabric that repeats itself both in the *x*- and *y*-directions. For each routing layer in the basic routing tile, the start and end location for each metal segment and the sites for the underlying fixed or configurable vias is specified. This scheme allows us to model the different routing fabrics described in Sec. 2.2.3.

4.3 Evaluation Metrics

In this section, we describe the evaluation metrics used to compare different configurability choices and different architectures. The configurability here refers to the number of custom metal and/or via masks that are required to implement a particular application on a Structured ASIC. The evaluation metrics we use are core area, delay, power and manufacturing cost. Each of these metrics is described below.

4.3.1 Area

We use the core area (A_{core}) as the area metric. If the logic block size (in units of half wire-pitches) is $L_x \times L_y$, the wire width and wire spacing are denoted by w_w and w_s, respectively, and the placement grid size is $N_x \times N_y$, then the core area is calculated as:

$$A_{core} = \left[N_x \times \frac{L_x}{2} \times (w_w + w_s)\right] \times \left[N_y \times \frac{L_y}{2} \times (w_w + w_s)\right]$$

4.3.2 Delay

We use the Elmore delay model to estimate the delay of an implementation [101]. For each net, we calculate the delay to each sink and average these values to obtain a net delay value. We then average all the net delays to obtain *average net delay* and use it as our delay metric. We use the average net delay, as opposed to critical path delay, for three reasons. First, the number of routing layers affects only the interconnect and this effect is captured in the average net delay. Second, it allows us to compare different configurability choices without knowing the internal details of the logic blocks such as the input-to-output delays or the location of flip-flops. Third, our CAD flow is not timing driven.

4.3.3 Power

For the power metric, we use the dynamic power dissipated in the interconnect since this is the primary component of power that would change as we vary the number of routing layers. We assume that a change in the number of routing layers has a negligible impact on glitching activity. We use the total interconnect (metal and via) capacitance as a first order estimate for power.

4.3.4 Cost

The manufacturing cost of the die is used to calculate the cost metric. The core area and the number of routing layers are used to estimate the manufacturing cost using the cost model described in Chapter 3.

4.4 CAD Flow

The CAD flow used to conduct the experiments is shown in Figure 4.2. The flow starts by reading in a technology-mapped circuit. The first step is to initialize the placement by reading in the physical size (height and width) of a logic block, the location of logic block inputs and outputs, and location of I/O pads of the circuit. The placement grid is set to a minimum square (i.e., if the technology mapped circuit has N blocks, then the initial grid size would be $\lceil\sqrt{N}\rceil \times \lceil\sqrt{N}\rceil$). After initialization, the placement is performed where each circuit block is assigned a location on the placement grid. We then route the placed circuit. If there is any congestion, rendering the circuit unroutable, the placement grid size is increased to create whitespace and the circuit is re-placed and re-routed. These steps are repeated until all the congestion is removed. The placement and routing steps are described in the following subsections.

4.4.1 Placement and Whitespace Insertion

To choose the most appropriate placer for our framework, we investigated two state-of-the-art standard-cell placers, CAPO [99] and NTUPlace [50], and an FPGA placer, VPR [34, 35]. The two important factors we used to decide which

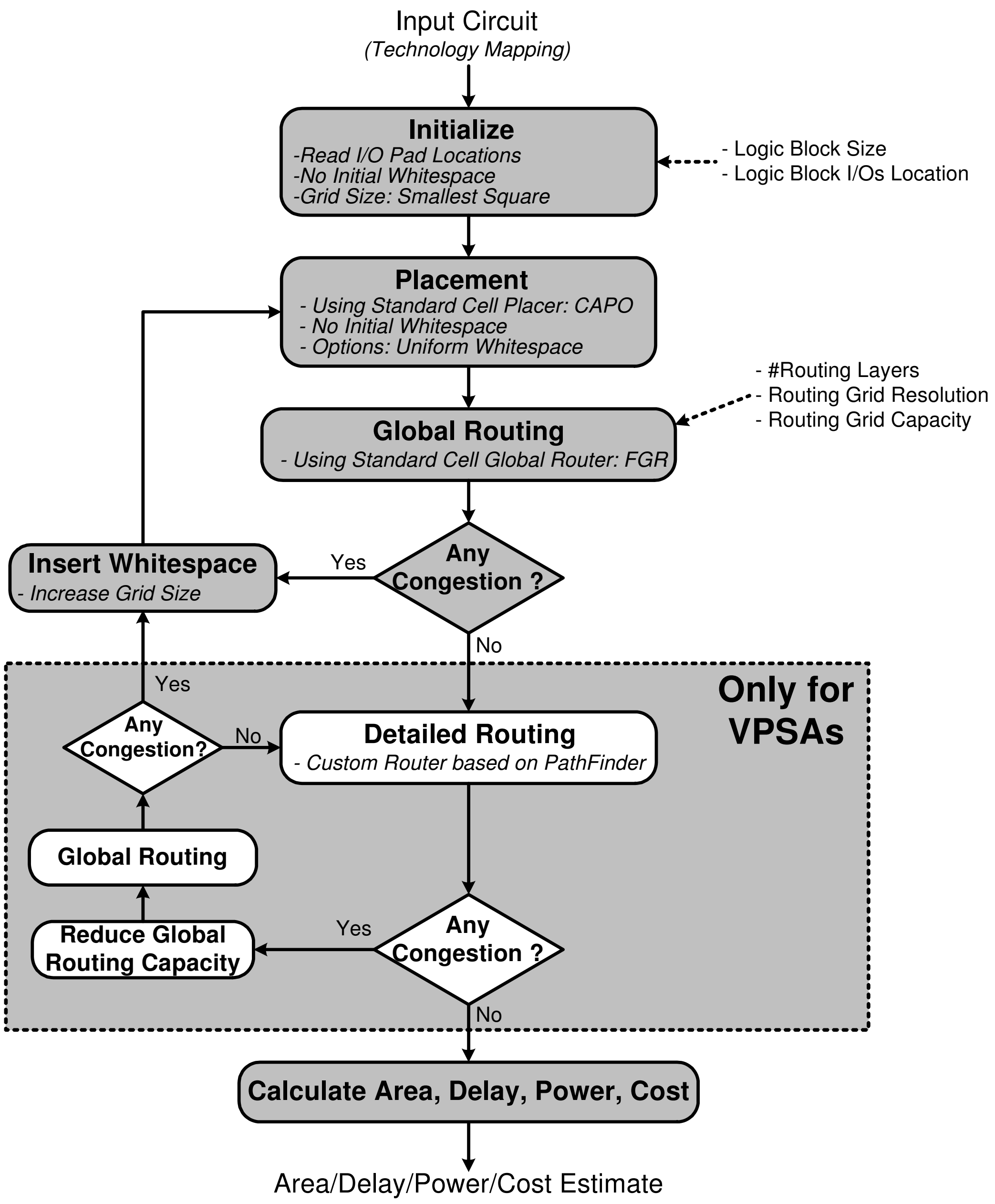


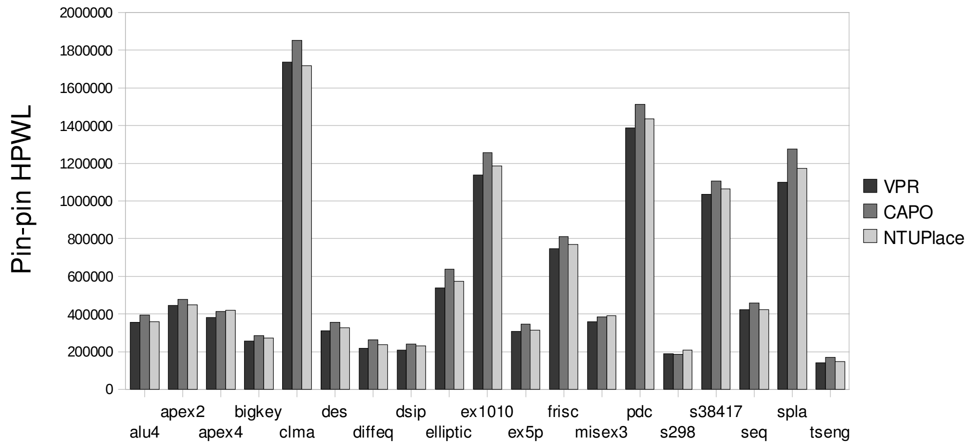
Figure 4.2: CAD flow

placement tool was appropriate for our experiments were runtime and the ability to insert whitespace to remove congestion. Runtime is important because Structured ASICs can have in excess of one million placeable blocks [104], especially if the logic blocks are fine grained. The whitespace insertion is crucial because with small logic blocks (compared to FPGAs), and the routing being done on top of them, Structured ASICs are more likely than FPGAs to experience congestion.

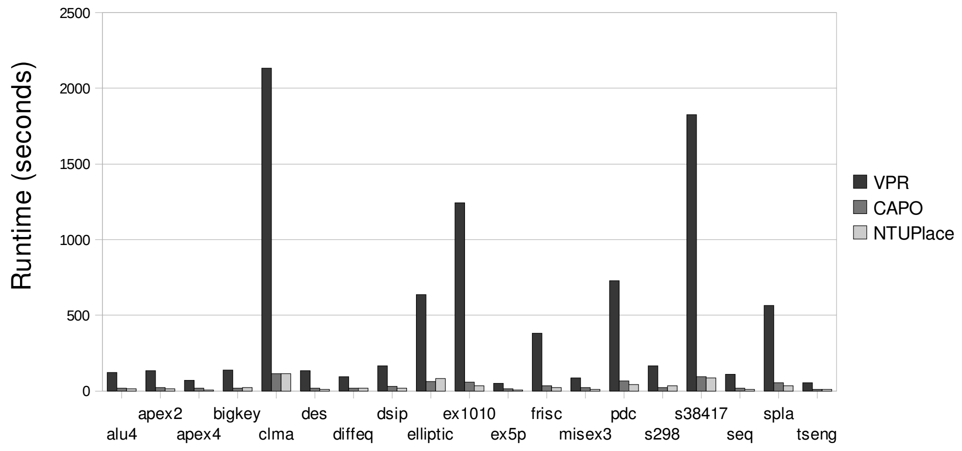
In Structured ASICs, the pre-fabricated logic blocks are similar to each other and are arranged on a grid. This is very similar to FPGAs, and implies that an FPGA placer may be suitable. However, even with circuits containing less than 15000 logic elements, we found VPR to be both slow, and unable to insert whitespace to remove routing congestion. The wirelength and runtime comparison of the three placers for nineteen largest Microelectronics Centre of North Carolina (MCNC) benchmarks is shown in Fig. 4.3. Compared to VPR, the standard cell placers, CAPO and NTUPlace result in 10% and 4% larger wirelength, respectively. However, they are faster than VPR by factors of 12× and 14×, respectively.

The comparison for whitespace insertion is shown in Fig. 4.4. The figure shows the placed blocks for the largest circuit (clma). The placement grid is double the minimum size. It can be seen that the wirelength-based cost function of VPR keeps all the logic blocks together. Although NTUPlace spreads out the logic blocks, it still has large clusters of tightly packed logic blocks towards the center of the placement grid. Thus, VPR and NTUPlace would not be able to remove congestion. Therefore, we use CAPO to perform placement in our CAD flow.

CAPO has different options for whitespace insertion; we use the uniform whitespace distribution. To eliminate congestion, we increase the grid size, thus creating whitespace, and then re-place the circuit, resulting in a better distribution of whitespace. Some circuits require a large amount of whitespace, therefore, to speed up the flow we use a binary search to find the minimum routable grid size.



(a) Half-perimeter wirelength comparison



(b) Runtime comparison

Figure 4.3: FPGA and ASIC placers: wirelength and runtime comparison

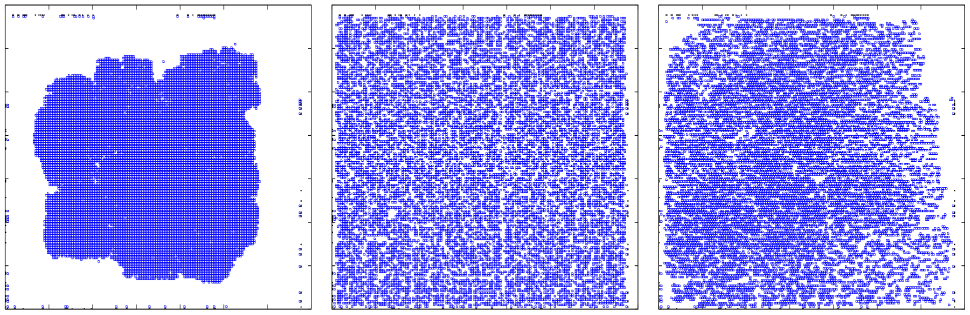

(a) VPR **(b)** CAPO **(c)** NTUPlace

Figure 4.4: FPGA and ASIC Placers: whitespace insertion

We use multiple passes of the placer for circuits with hard macro blocks such as memories and register files. In the first pass, we perform the placement without imposing any constraints on the positions of the different blocks. This global placement is then legalized by moving each macro block to its nearest, empty legal site in the MPSA device architecture. The block's position is then locked and not modified in the next pass. In the second pass, with all the hard macro blocks locked to a legal position, we re-place the logic blocks.

If the logic fabric has dedicated flip-flops, a third pass can substantially improve wirelength. We consider these flip-flops as hard macro blocks and do not change their position in the second pass. The second pass only changes the position of logic blocks. Then, in the third pass, we fix the logic blocks and other hard macros, and re-place the flip-flops. Alternative approaches, e.g., placing flops before logic, were found to give inferior results. Additional passes (e.g., repeating passes 2 and 3) were found to improve the wirelength by 10%, but this roughly doubles runtime.

Recently, a new open-source structured ASIC placer, RegPlace, has been released [43]. RegPlace attempts to assign hard macro blocks to their legal sites and it also takes into account multiple clock domains. However, RegPlace is not directly applicable in our case because of its inability to insert whitespace. In fact, in its "Wirelength Recovery" step, it explicitly tries to bring connected cells closer to each other which is likely to cause more routing congestion.

4.4.2 Routing

After placement, the next step is to route all the nets in order to estimate the wirelength. In our flow, we use the FGR global router [98]. In addition to the list of nets to route, the inputs to the router include the number of available layers for routing, the resolution of the global routing grid (number of logic blocks encapsulated in a global routing tile), and the grid capacity (number of metal wires that can pass through the global routing tile). We use different approaches to perform detailed routing for MPSAs and VPSAs. These are described below.

MPSA Detailed Routing

The MPSA routing problem is very similar to the ASIC routing problem. Detailed routing for ASICs confines the connections to the given global routing path and deals mainly with meeting the design rules [33]; in general, the quality of the routing results is dictated mostly by the global route. Therefore, to simplify the flow we do not perform detailed routing for MPSAs. We constrain the global router so that it can only use up to 85% of the available tracks. We assume that this accounts for the overhead of satisfying the design rules. As is typical with ASICs, we assume that a successful global routing result can always be detail routed with negligible wirelength overhead.

VPSA Detailed Routing

In VPSAs, the nets have to be routed on pre-designed metal segments. During routing, when a portion of a net needs to switch from one track position to another on the same layer (e.g., due to congestion, or to a reach a specific logic block pin) it consumes a metal segment from the above or below layer. This can result in two problems: (1) the number of segments available for routing on the adjacent layers is reduced, which can make a circuit unroutable despite a successful global routing solution, and (2) the length of each net could be much larger than the length of its global route. Therefore, to ensure a valid routing solution, and to accurately estimate the wirelength and delay of a circuit implementation, we perform detailed routing for VPSAs in our CAD flow.

The detailed routing problem in VPSAs is very similar to FPGAs — the routing resources (metal segments) are fixed and the connections can be made between these resources by selectively inserting or removing a via. However, there are two issues in directly using the FPGA router built into VPR [35]. First, the VPSA routing fabric is very flexible compared to FPGAs; a via can be placed/removed from any intersection of the metal segments, making it a fully connected crossbar. Using a single routing resource graph to represent the entire routing fabric, as is done in VPR, can have a huge memory footprint.[1] Second, the high flexibility of the routing fabric implies a large search space for the router, which can affect the runtime of the router. Therefore, we have developed our own router to perform detailed routing.

The routing algorithm used by our router is similar to the one used by VPR. However, there are some enhancements to reduce memory footprint and improve runtime. To reduce memory footprint, we only create a graph for one basic routing tile [51]; during the shortest path search, only this small graph is used. The runtime is improved by only expanding along the global route. It is possible to restrict the search path to the global route because of the high flexibility of the VPSA routing fabric.

The inputs to the router include the global routes, and the position of the fixed metal segments that repeat over the die. We use this specification to create the routing resource graph of the basic routing tile. We use large penalties for negotiating congestion; a value of 4000 is used for present congestion cost, and 0.5 is used for history congestion cost. With smaller penalties, the runtime increases significantly without improving the routing quality. Using large penalties allows us to find a valid routing within 8 iterations. If the congestion is not eliminated completely by then, we terminate the detailed routing, insert more whitespace, and then start with the placement phase again.

[1]A possible connection between two routing segments is represented by an edge in the routing resource graph. With a fully connected crossbar, a routing resource graph for a circuit with only 2000 logic elements can contain more than 30M edges. This would require about 1GB of memory only to store the edges.

4.5 Summary

In this chapter we described our experimental CAD flow and the architecture space it can cover. The CAD flow makes use of open-source CBIC global placement and global routing tools. It includes a custom detailed placer to perform legalization for Structured ASIC architectures containing different types of logic blocks. It also includes a custom detailed router to perform detailed routing for VPSAs. The CAD flow provides area, delay, and power estimates for a benchmark circuit when implemented on a particular Structured ASIC architecture.

Chapter 5

Metal-Programmable Structured ASICs

5.1 Overview

In this chapter, the experimental results for MPSAs are presented. The experiments were conducted on two different suites of benchmark circuits. The first benchmark suite consists of circuits that contain only one type of logic block. We define this suite as *homogeneous circuits*. The second benchmark suite consists of circuits that contain embedded IP blocks (block Random Access Memory (RAM), register files, etc.) in addition to the logic cells. We define these circuits as *heterogeneous circuits*. For each circuit in these benchmark suites, we vary the number of routing layers, and use the CAD flow described in Chapter 4 to collect area, delay, power, and cost statistics. Using these statistics, we study how the number of routing layers impacts the cost and performance of an MPSA.

We also study how sensitive the MPSA cost results are to the various constants used in the cost model. To analyze this sensitivity, we compare the MPSA and CBIC costs for a range of values of various parameters given in Table 3.2.

Finally, we study the impact of the whitespace insertion algorithm on MPSA die-cost results. The CAD flow employs uniform whitespace distribution. This can result in large die-areas. We estimate the die-area and die-cost savings in MPSAs due to the use of an intelligent whitespace insertion algorithm that inserts whitespace only in the congested regions.

5.2 Homogeneous Circuits

We use circuits from the MCNC benchmark suite as homogeneous circuits. The twenty largest MCNC benchmark circuits have commonly been used in the research on FPGAs [35] and Structured ASICs [97]. We use the nineteen largest circuits.[1] One of the circuits, s38584.1, contains a net with more than 3000 pins which was too large for the global router; we chose to exclude the benchmark rather than modify it or the CAD flow.[2]

To study the performance and cost trends, we consider a range of logic block architectures. The logic blocks have different number of inputs and outputs, and they also vary in their physical sizes. In the following subsections, we describe our approach for technology mapping, the method used to calculate the physical

[1]Characteristics of these benchmarks are shown in Appendix A.

[2]For such high-fanout nets, a different type of a router (e.g., a clock router) would typically be used, or the generating logic would be duplicated.

sizes of the logic blocks, and the performance and cost trends.

5.2.1 Technology Mapping

The input to our CAD flow is a technology-mapped circuit. The technology mapping depends on the internal structure of each logic block in the MPSA. In order to focus our attention on the interconnect architecture, we abstract the contents of the block by representing only its input and output pins, and the block area. This means that an exact technology mapping is impossible. Instead, we perform a clustering step to produce an interconnect netlist that approximates a real technology-mapped netlist. Our benchmark circuits are expressed in terms of 2-input gates. We cluster these basic gates such that each cluster has a specific number of inputs and outputs that matches the number of inputs and outputs of a particular logic block architecture. Such a clustered netlist has many of the properties (such as fan-in and fan-out distributions, Rent parameter, etc.) of a real technology-mapped circuit. We use T-VPack ([35]), an FPGA clustering algorithm, for this purpose.

Because we avoid real technology mapping, we must be careful not to compare the results obtained using two different logic blocks (I/O counts) directly. Hence, we do not draw any conclusions about which logic block is better. Instead, we average results across logic blocks and only compare the results for different layout areas.

5.2.2 Logic Block Dimensions

Our experimental methodology requires the pin locations and the layout area (height and width) for each logic block. We randomly assign pin locations within each cell such that all the logic blocks that have the same number of inputs and outputs have the same pin locations.

The layout area for a particular logic block depends upon the contents (number of gates) and the effort of the layout artist, both of which are hard to estimate precisely. Instead, we determine the minimum and maximum area values for each logic block architecture and sweep through five equally spaced points in that range. The minimum cell area represents a very dense layout. We use the number of logic block pins (p) to calculate the minimum cell area. The minimum area (in units of wire half-pitches) to fit p pins is $2\lceil\sqrt{p}\rceil \times 2\lceil\sqrt{p}\rceil$. However, we have seen that it is not possible to connect to such a dense arrangement of pins. Therefore, we assume the minimum layout area to be $4\lceil\sqrt{p}\rceil \times 4\lceil\sqrt{p}\rceil$.

For maximum layout area, we find an area number (A) for an "average" gate by averaging the areas of different basic standard cells such as NAND, NOR, MUX, etc. If the logic block has o outputs, then we assume the maximum area to be $\lceil\sqrt{A\cdot o}\rceil \times \lceil\sqrt{A\cdot o}\rceil = A\cdot o$.

Table 5.1 shows the different logic block *types* (I/O counts) and the corresponding layout area values used in our experiments.

Table 5.1: Logic blocks used in experiments

Type		Block Layout Area in Half-Pitches (Width×Height)				
		High Density		···		Low Density
IN	*OUT*	*Min.*	*Small*	*Medium*	*Large*	*Max.*
2	1	8×8	12×12	15×15	19×19	22×22
4	2	12×12	17×17	22×22	27×27	32×32
6	3	12×12	19×19	26×26	33×33	39×39
8	4	16×16	23×23	30×30	37×37	44×44
10	5	16×16	25×25	33×33	42×42	50×50
12	6	20×20	29×29	37×37	46×46	54×54
14	7	20×20	30×30	40×40	50×50	59×59
16	8	20×20	31×31	42×42	53×53	63×63

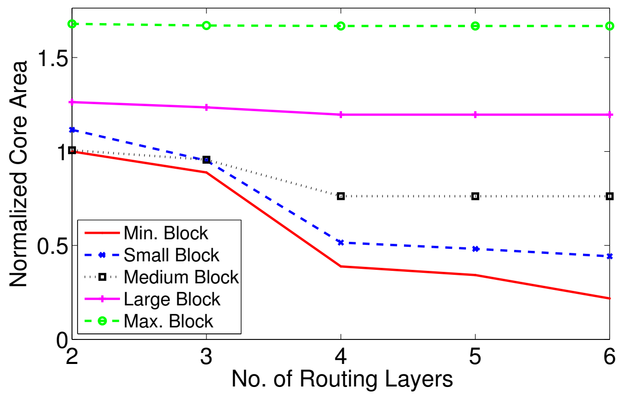
5.2.3 Performance and Cost Trends

The trends for area, delay and power as a function of the number of routing layers, averaged over all the MCNC circuits, are shown in Figure 5.1. We study the trends for 2–6 routing layers. With an odd number of layers, there are more metal segments along one direction (horizontal or vertical) than the other direction. However, despite the asymmetry, such fabrics help in reducing congestion (e.g., see Figure 5.5).

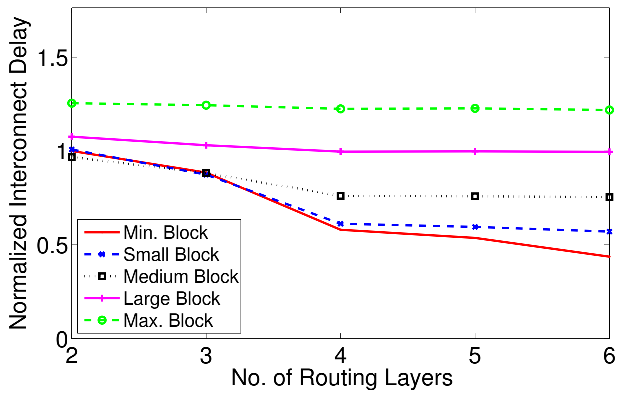
The plots show averaged (geometric mean) data of all the different logic block types for all the circuits. The plots are normalized to the values of minimum block layout area with two routing layers. We define the *nominal area* to be the geometric mean of the core area of the minimum block layout area with two routing layers. The nominal area in these plots is $0.008mm^2$. There are four important observations. First, for larger block layouts, the area, delay, and power does not change as we increase the number of routing layers. This is because the blocks are so large that even with two layers there is no congestion, therefore, there is no effect of adding subsequent routing layers. Second, for smaller block layouts, the improvements in area, delay and power are quite small after four routing layers. Third, in some cases, given the same number of routing layers, the core area with larger blocks can be smaller than the core area with small blocks (e.g., core areas for “Medium” and “Small” blocks with two routing layers in Figure 5.1(a)). This is primarily because of the uniform whitespace distribution scheme used during placement. The total whitespace required for small blocks is more than the whitespace inserted for larger blocks, which increases the core area. The use of an intelligent whitespace insertion algorithm (one that inserts whitespace only at the congested areas) could alleviate this problem. Finally, area is the most sensitive to the addition of extra routing layers, while power is the least sensitive. These trends are similar when the averaged data shown in Figure 5.1 is examined for individual logic block types (I/O counts), but these data are not shown due to space constraints.

Next, we estimated the die-cost by applying the cost model described in Chapter 3. However, the homogeneous circuits we used are quite small. Such small circuits are not realistic; this artificially increases (N_{gdpw}) significantly, reducing Equation 3.8 to $N_{ml}K_2 + K_3$. Because of this, we scaled the core area to a realistic value before applying the cost model.[3] To do this scaling, we multiplied the core areas by a common factor such that the nominal core area matches a desired area value. We considered three desired values for the nominal core area. These values

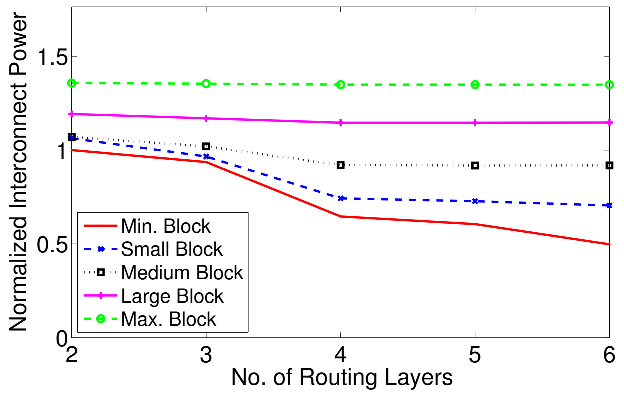
[3]In section 5.3, we show results on larger circuits that do not require any scaling.



(a) Area

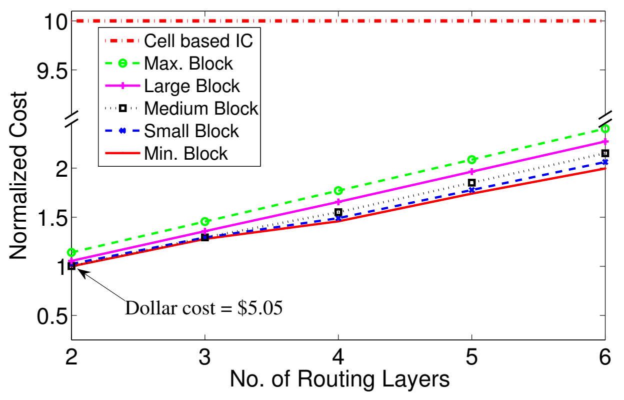


(b) Delay

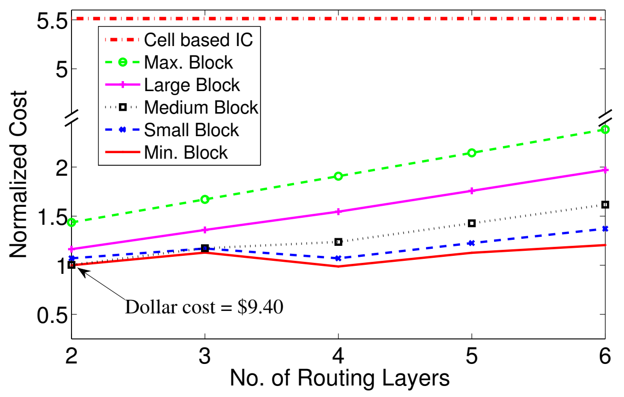


(c) Power

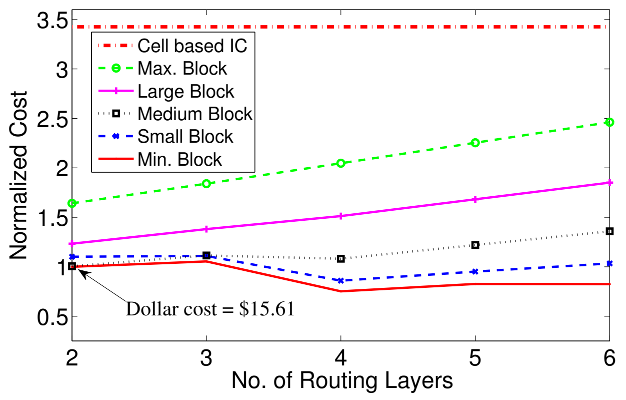
Figure 5.1: Area, delay, and power trends for MCNC circuits. The nominal core area (core area of Min. Block with 2 routing layers) at 45nm is only 0.008mm^2



(a) Nominal core area = $10mm^2$



(b) Nominal core area = $50mm^2$



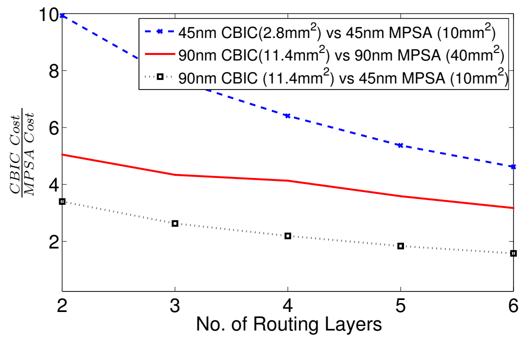
(c) Nominal core area = $100mm^2$

Figure 5.2: MCNC circuits: trends for die-cost at 45nm

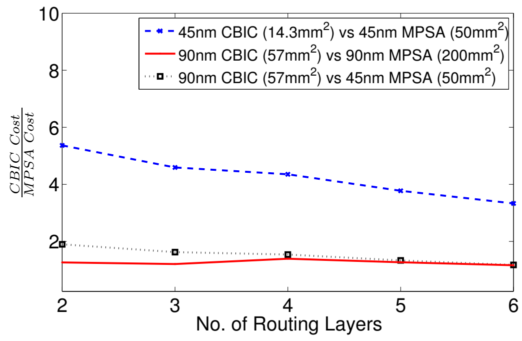
are 10mm^2, 50mm^2, and 100mm^2. The resulting cost plots are shown in Figure 5.2. It can also be seen from Figure 5.2 that, for small die sizes, the minimum cost is achieved with only two routing layers; the cost of adding an extra layer is almost always greater than any cost savings due to area reduction. However, for large die sizes, additional routing layers reduce cost modestly for only the most dense block layouts.

We also show the estimated CBIC cost in Figure 5.2, produced using the core area of a "Min" block layout area, six routing layers, and all custom masks. It can be seen that, despite the small area of CBICs, there is a significant gap between the cost of an MPSA and a CBIC for smaller dies. This difference, however, diminishes as the die sizes grow, suggesting that CBICs may be cost-effective for extremely large designs (i.e., greater than 100mm^2).

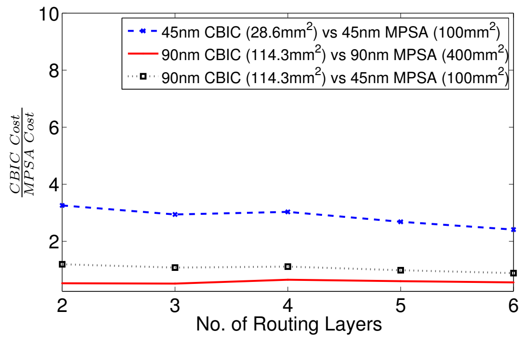
Finally, we compare the cost of implementing a design in an MPSA and a CBIC using different process technologies. We consider 90nm and 45nm process technologies. The area of the 90nm implementation is 4$\times$ the area of 45nm implementation. For MPSA costs we assumed a "Medium" block layout area whereas for CBIC we assumed "Min" block layout area. With these assumptions, the MPSA implementation of a design requires 3.5$\times$ more area than the CBIC implementation in the same process technology. The ratio of CBIC costs to MPSA costs are then shown in Figure 5.3. It can be seen that, for smaller dies, the MPSAs are more cost effective than CBICs despite a 3.5$\times$ area penalty. The cost effectiveness improves as we scale to finer process geometries. The figure also shows the comparison of a 90nm CBIC implementation versus a 45nm MPSA implementa-



(a) 45nm MPSA Core Area=10mm^2; 90nm MPSA Core Area=40mm^2



(b) 45nm MPSA Core Area=50mm^2; 90nm MPSA Core Area=200mm^2



(c) 45nm MPSA Core Area=100mm^2; 90nm MPSA Core Area=400mm^2

Figure 5.3: Cost advantage of MPSAs over CBICs at 90nm and 45nm (higher value means MPSA is lower cost)

tion. Again, MPSAs are much cheaper than a CBIC implementation, especially for small die sizes. This suggests that the MPSAs can make modern technologies more affordable than older CBIC technologies. Interestingly, we can also see that MPSAs are not cost-effective against CBICs for large dies when both are implemented in 90nm; this may partly explain the slower than anticipated adoption rate of Structured ASICs to date.

5.3 Heterogeneous Circuits

For heterogeneous circuits, we used circuits that were released by eASIC as part of a placement contest [3]. These circuits are modified versions of large industrial designs and contain up to approximately one million logic blocks. The circuits utilize four different types of logic elements: ecells (logic block), flip-flops, block RAMs, and register files. The circuits have been technology mapped to an architecture (described below) that contains these different types of logic blocks. The internal architecture of the different blocks is not disclosed. There are multiple clock domains in these circuits, however, for our experiments we only assume a single clock domain.

In the following subsections, we describe the eASIC device architecture, our approach for technology mapping and logic block area calculation, and the performance and cost trade-offs.

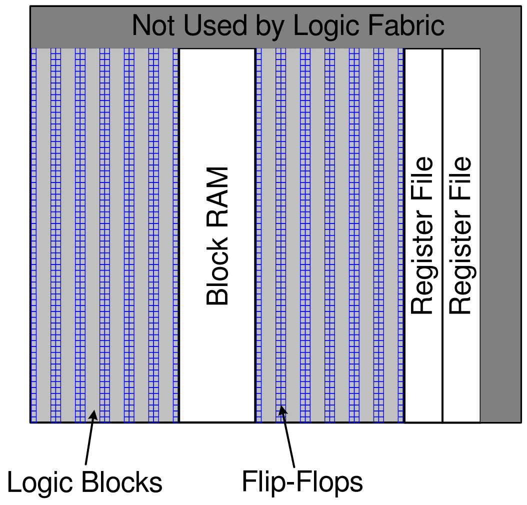


Figure 5.4: Basic logic fabric group of the eASIC device

5.3.1 Device Architecture

The architecture of the eASIC device is similar to a column-based FPGA. The basic building block is called a "group" which consists of columns of logic blocks and flip-flops, block RAMs, and register files. There is a fixed site for each block type in a group and a group can have four different clocks. The chip is made up of an array of groups and can have 32 different clocks. Figure 5.4 shows the organization of the eASIC group.

5.3.2 Technology Mapping

The benchmark circuits are mapped to the eASIC device, however, the original technology mapping of the circuits is very sparse. Table 5.2 shows the characteristics of the original eASIC benchmark circuits. The logic block has 9 pins (7 input pins and 2 output pins), but the circuits, on average, are only using 3 pins.

Because of such a sparse technology mapping, there is no congestion and all the circuits were routable with only two layers. In this case, the results were similar to the results of MCNC benchmarks with “Max” block layout area (Figure 5.1). Therefore, we modified the circuits by clustering the logic blocks to make the mapping more dense. We used the T-VPack algorithm ([35]) for clustering. The characteristics of the clustered circuits are shown in Table 5.3. On average, approximately three blocks are clustered together without violating the input and output constraints of the eASIC logic block (7 inputs and 2 outputs). In some cases, the number of blocks clustered together is as high as twenty. The resulting packed netlists are fairly dense where circuits on average use six out of nine available pins of the logic block. We use these packed netlists for our experiments.

5.3.3 Logic Block Dimensions

To conduct the experiments, we need an estimate of the layout area for different circuit elements in the eASIC architecture. The smallest circuit element is the logic block and the area of the other blocks can be expressed in terms of the logic block area. The relative area of different circuit components can be found from the benchmark files. We estimated the block RAM layout area from its size (36kb dual-port memory), and used that to determine the layout area of a logic block. We defined this logic block as “Medium Block” and it has a layout area (in units of wire half-pitches) of 69×69. We also consider two other logic blocks: one with a $0.5\times$ layout area and the other with a $2\times$ layout area of “Medium Block”. We define these as “Small Block” and “Large Block” respectively. The layout

Table 5.2: Characteristics of eASIC benchmarks

Circuit	Inputs	Outputs	Nets	Logic Blocks	Flip Flops	Block RAM	Register File	Avg. Used Pins per Logic Block (Total Pins: 9)
easic1	151	311	930,116	832,824	87,052	110	172	3.18
easic2	83	28	873,483	812,200	45,478	175	686	3.08
easic3	421	1	1,016,364	961,063	52,780	192	0	2.99
easic4	123	292	126,224	102,038	23,330	0	44	3.06
easic5	26	149	1,010,422	913,853	84,505	145	262	3.14

Table 5.3: Characteristics of packed eASIC benchmarks (modified technology mapping)

Circuit	Nets	Logic Blocks	Max. Blocks per Cluster	Avg. Blocks per Cluster	Avg. Used Pins per Cluster
easic1_packed	723,455	318,448	14	2.64	5.99
easic2_packed	658,846	304,520	14	2.67	5.87
easic3_packed	533,161	261,366	19	3.68	6.41
easic4_packed	82,822	31,746	13	3.22	6.27
easic5_packed	738,587	332,341	20	2.77	5.99

area values of these blocks, in terms of wire half-pitches, are 50×50, and 96×96, respectively.

5.3.4 Performance and Cost Trends

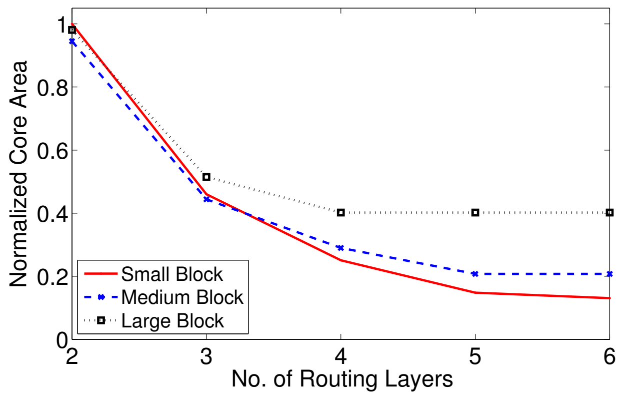
We pass four of the circuits through the CAD flow. The placement grid for the smallest circuit, easic4, is limited by the register files rather than the logic blocks, so we do not use this circuit in our experiments. We collect area, delay, and power statistics for different numbers of routing layers and use the cost model described in Chapter 3 to calculate the die-cost. The plots for the average (geometric) area, delay, and power trends are shown in Figures 5.5 (a), (b), and (c) respectively. All the plots are normalized to the values for "Small Block" with two routing layers.

There are four major observations. First, the area, delay, and power performance improves with more routing layers. The bulk of the improvement occurs in going from 2 to 4 layers; for example, Small Block area and delay reduces by 75% and power reduces by 50%. For the same block size, the improvement in area, delay, and power from 4 to 6 layers is only 12%, 11%, and 13% respectively. The trends for other block sizes are similar.

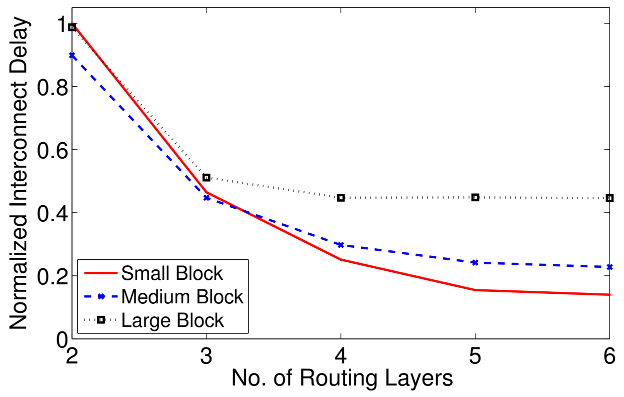
Second, with fewer routing layers, the difference between the different block sizes is small but it grows with more routing layers. With two routing layers, the difference in area, delay, and power of a Structured ASIC containing Small Blocks and one containing Large Blocks is 2%, 1%, and 0%, respectively. The same difference with 4 layers is 15%, 20%, and 13% respectively; with 6 layers, the difference grows to 27%, 31%, and 26%, respectively.

Third, area is most sensitive to the number of routing layers, whereas power is least sensitive. The reduction in area and power in going from 2 to 6 layers with Small Blocks is 90% and 60%, respectively.

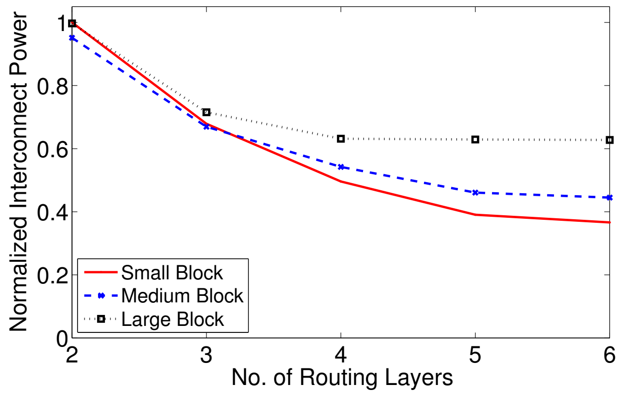
Finally, we see that with 2 routing layers, the area, delay and power with Small Blocks are more than the Medium Block. We also see that in going from 2 to 3 layers, there is a significant performance improvement. The reason for both these observations is related to the use of uniform whitespace insertion algorithm in our CAD flow. The available whitespace gets distributed across the core rather than just at the congested regions. As a result, a large amount of whitespace needs to be inserted to successfully route highly congested designs, which is the case with a small number of routing layers and/or small block sizes. In Section 5.5, we



(a) Area



(b) Delay



(c) Power

Figure 5.5: Packed eASIC circuits: area, delay and power trends (core area with Small Block and two routing layers = 162mm^2)

provide an insight into the improvement that might be obtained from the use of an intelligent whitespace insertion algorithm.

Next, we estimate the die-cost using the area values of Figure 5.5a. The resulting plot is shown in Figure 5.6. The plot shows that the decrease in core area with more routing layers does not reduce the die-cost by the same proportion. It can be seen that the reduction in die-cost obtained by having more than 3 routing layers is very small and there is almost no cost advantage of having more than 4 routing layers. The reason for this behavior is the large cost associated with the mask-set; cost savings resulting from smaller die sizes are offset by the increase in cost due to the use of additional custom masks.

We also compare the MPSA die-cost of heterogeneous circuits to the corresponding CBIC cost. We estimated the CBIC cost using the MPSA area value (with Small Block and 6 routing layers) and consider all masks as custom. The resulting cost is also shown in Figure 5.6. It can be seen that 2-layer MPSAs have a $2\times$ cost advantage over CBICs, and with 4 routing layers it grows to about $4\times$.

5.4 Cost Sensitivity Analysis

In this section we study the sensitivity of the die-cost to some of the parameters of Table 3.2. In particular, we look at the effect of different volume requirements (V_c and V_{tot}), mask-set prices (C_{sm_l} and C_{sm_u}), and number of fixed lower masks (N_{fm_l}). We have noticed that the trends for different MPSAs (different logic block sizes and different number of routing layers) are largely insensitive to these parameters. However, the cost of MPSAs relative to CBICs does change. Therefore,

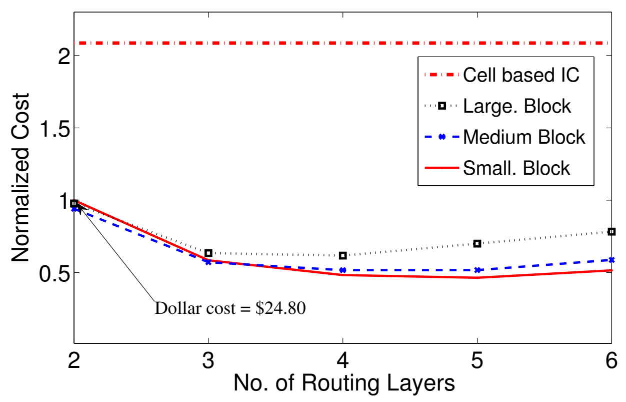


Figure 5.6: Die-cost trend for packed eASIC circuits (normalized to cost values for “Small Block”)

we only compare the die-cost of 45nm CBICs against the 45nm MPSA with Small Block and 2 routing layers and show the results for heterogeneous circuits.

The sensitivity of die-cost to volume requirements is shown in Figure 5.7. We considered a range of values for customer volume (V_c) and total device volume (V_{tot}). The results show that the die-cost is much more sensitive to V_c than V_{tot}. This is because CBIC mask-set cost is amortized over the customer volume only. For small volumes, MPSAs are therefore very cost effective.

Next, we look at the impact of mask-set cost. There are two factors in the mask-set cost: (1) the total mask-set cost, and (2) the ratio of the cost of the lower and upper masks (C_{sm_l} and C_{sm_u}, respectively). We considered these two factors and also considered different device volumes. The results are shown in Figure 5.8. It can be seen that the higher mask-set costs favor MPSAs, especially for smaller volumes. Additionally, the current trend shows that as technology scales, $C_{sm_l} : C_{sm_u}$ ratio becomes larger [90]. Figure 5.8 shows that these trends

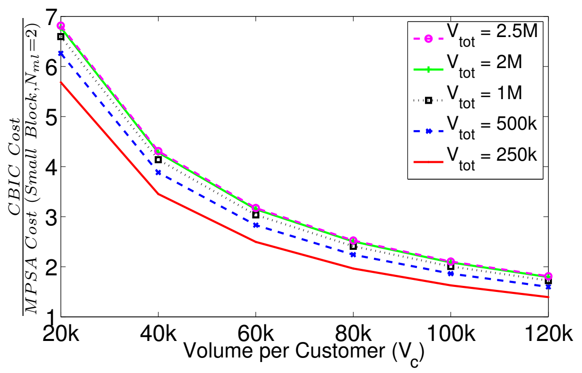


Figure 5.7: Die-cost sensitivity to volume requirements

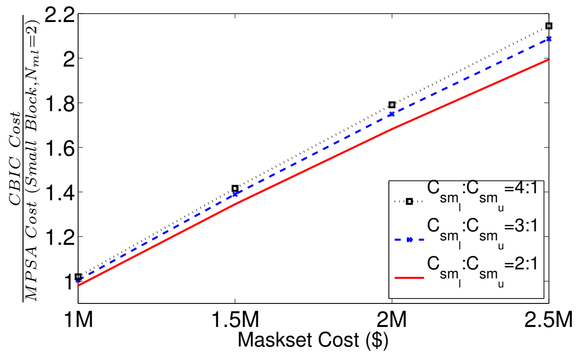
also favor MPSAs.

Finally, we also modeled different processes in which the number of masks needed to manufacture the fixed portion of the device (N_{fm_l}) may differ. The results, shown in Figure 5.9, illustrate that a larger value of N_{fm_l} would favor MPSAs over CBICs. This is because, with large N_{fm_l}, a larger portion of the cost of the mask-set is amortized over total device volume (V_{tot}). This lowers the per-die cost of MPSAs.

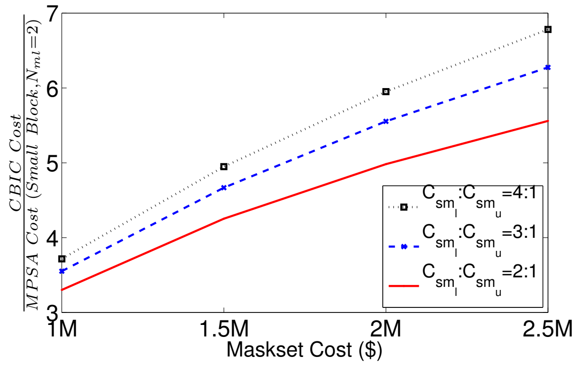
5.5 Impact of an Intelligent Whitespace Insertion Algorithm

We use uniform whitespace insertion in our CAD flow. As described in Section 5.3, one of the problems with this approach is that a significant amount of whitespace needs to be inserted before all congestion is removed. This results in a large die-area and increased wirelength which degrades delay and power.

The nature of the whitespace insertion problem in MPSAs is similar to that



(a) $V_c = 100k, V_{tot} = 2M$



(b) $V_c = 20k, V_{tot} = 500k$

Figure 5.8: Die-cost sensitivity to mask-set cost

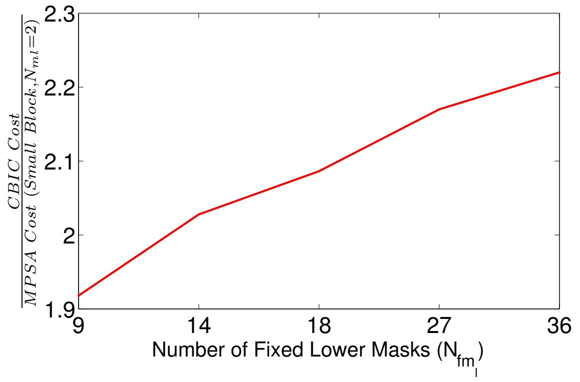


Figure 5.9: Die-cost sensitivity to number of fixed masks (N_{fm_l})

of FPGAs. As described in Chapter 2, the use of empty CLBs as whitespace has not been very successful in FPGAs. Instead, most published techniques rely on depopulating the logic blocks (using fewer LUTs than are available in each logic block) [112][113][53]. In our MPSA logic block model, we are assuming a fully packed logic block. Therefore, this technique is not directly applicable. In our experiments we have noted that some of the congestion-aware placement options available in the existing academic placers were not able to produce routable placements, especially when there are few metal layers available for routing. Developing a new suitable whitespace allocation algorithm is beyond the scope of this dissertation. Instead, in this section, we *estimate* the impact that an intelligent, congestion-aware whitespace insertion algorithm would have on our results.

Our approach for this estimation is as follows. Assume that the minimum number of routing layers for which a given circuit can be routed without any whitespace insertion is L. For architectures with fewer than L routing layers, not all nets can be routed due to congestion. To remove this congestion, an intelligent whitespace insertion algorithm would leave selected logic blocks empty; if this is done correctly, then the circuit can be routed using fewer than L routing layers, since each empty logic block is accompanied by a set of routing tracks, and these tracks can be used to route nets in the circuit. For an architecture with L' routing layers, where $L' < L$, our approach is to estimate the number of logic blocks N that need to be left empty such that the total number of available routing tracks in the new architecture with L' routing layers is same as the total number of available routing tracks in the architecture containing L routing layers. The die area and

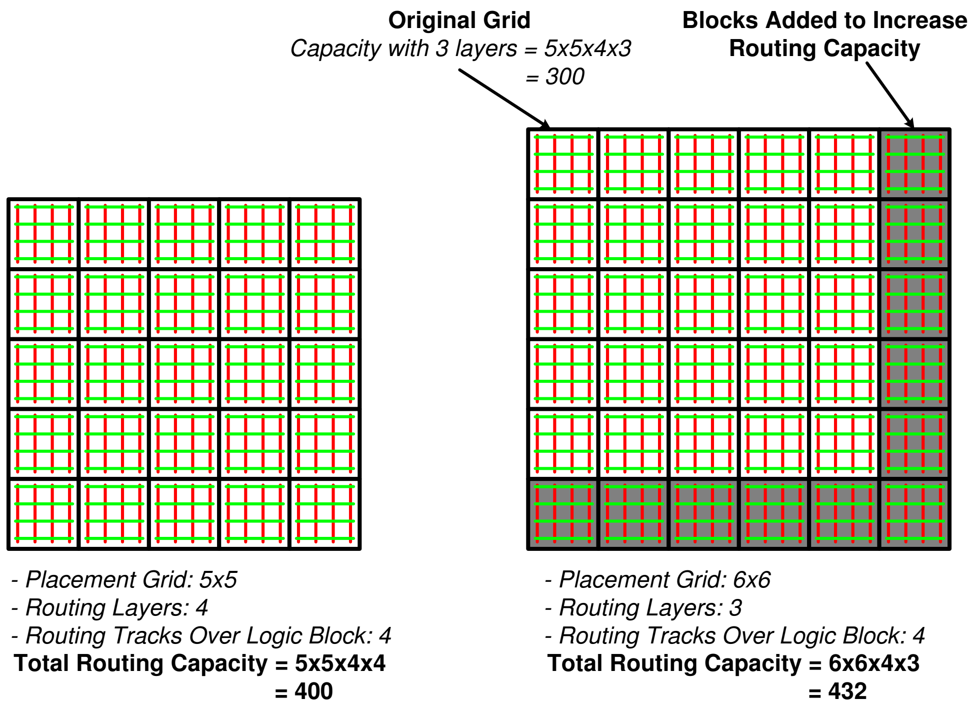

Original Grid
Capacity with 3 layers = 5x5x4x3
= 300
Blocks Added to Increase Routing Capacity
- Placement Grid: 5x5
- Routing Layers: 4
- Routing Tracks Over Logic Block: 4
Total Routing Capacity = 5x5x4x4
= 400
- Placement Grid: 6x6
- Routing Layers: 3
- Routing Tracks Over Logic Block: 4
Total Routing Capacity = 6x6x4x3
= 432

Figure 5.10: Estimating die-area with use of an intelligent whitespace insertion algorithm

dollar cost of an architecture with N additional logic blocks but only L' routing layers can then be computed using our previous techniques.

To calculate N, we do the following. Consider a placement grid of $X \times Y$ logic blocks that is routable without any whitespace using L routing layers. If the size of a logic block is such that t $(= t_x = t_y)$ routing tracks can pass over it in one layer, then the total routing capacity T is $T = X \times Y \times t \times L$. If the number of routing layers is reduced by ΔL, then the total reduction, R, in the routing capacity is $R = X \times Y \times t \times \Delta L$. We then use R to calculate N as follows:

$$N = \frac{R}{(L - \Delta L)t} = X \times Y \times \frac{L - L'}{L'}$$

and consequently the new placement grid size is: $\lceil \sqrt{(X \times Y) + N} \rceil \times \lceil \sqrt{(X \times Y) + N} \rceil$. This process is illustrated in Figure 5.10 for $X = Y = 5$, $L = 4$, $t = 4$, and $\Delta L = 1$. This estimation technique is optimistic in that it shows the "best case" benefit that might be achieved. In practice, the benefit will likely be less.

The die-area values obtained by using the above technique for heterogeneous circuits are shown in Figure 5.11 along with the original area values. To gather these results, we found the minimum L for which the circuit can be routed, and iterated for all values $L' < L$, each time calculating the area as above. For each point, if the estimated area turns out to be more than the area obtained from the CAD flow, we use the CAD-area instead for the current and subsequent area calculations. It can be seen from the graph that an intelligent whitespace insertion algorithm has the potential to provide significant savings in die-area, especially when there are few layers available for routing. The most potential for area savings is with an architecture containing a Small Block where the estimated die-area for 2 routing layers is 60% less than what was obtained using uniform whitespace allocation. This difference reduces to 7% if 4 routing layers are available.

The die-cost values corresponding to the estimated area values are shown in Figure 5.12. As the graph shows, the 60% area saving (due to improved whitespace insertion) for the Small Block with 2 layers, translates to a 55% cost reduction. However, the area reduction in going from 2 to 4 layers does not translate into any cost advantage. Another observation is that the layout area of the logic block now has an impact on the die-cost when there are only two layers in the routing fabric. There is 12% difference between the die-cost of Small and Medium Blocks, and a difference of 20% between Medium blocks and Large Blocks. Finally, it can also be seen that the minimum cost point for Small and Medium Blocks has moved from 5 or 4 layers, respectively, to 3 layers.

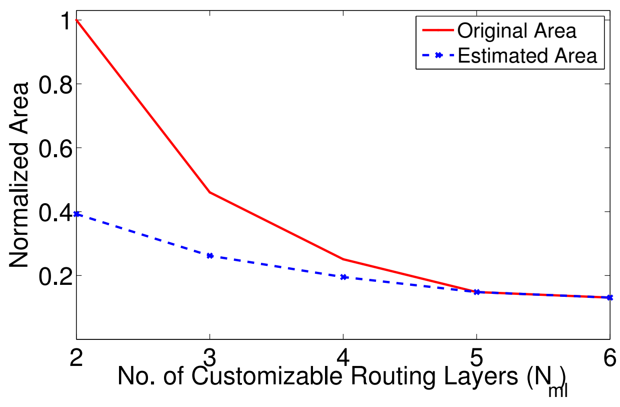
From Figure 5.5, we see that the trends for delay and power are similar to area, therefore we expect the impact of the whitespace insertion algorithm on delay and power to be similar to that of area.

These results show that a significant reduction in the die-area and die-cost can be made by improving the CAD flow. With the current whitespace insertion techniques, there is very little advantage of having a small, densely laid out logic block, especially with few routing layers. In the future, however, as better CAD techniques evolve, densely laid-out logic blocks will become advantageous.

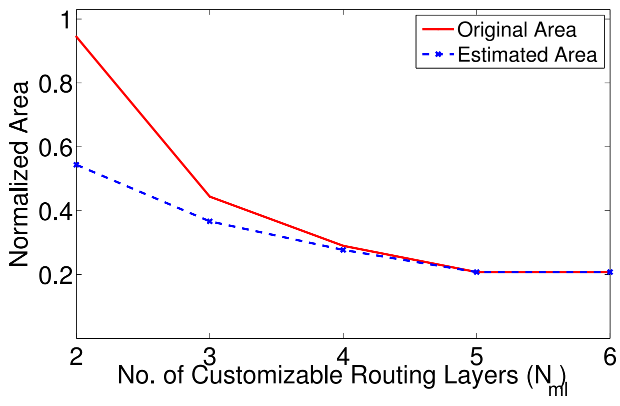
5.6 Summary

Area, delay, power and cost trends for MPSAs were presented in this chapter. Area is the most sensitive, whereas the power is the least sensitive to the number of routing layers. The sensitivity also varies with logic block layout density; high-density layouts have greater sensitivity than low-density layouts.

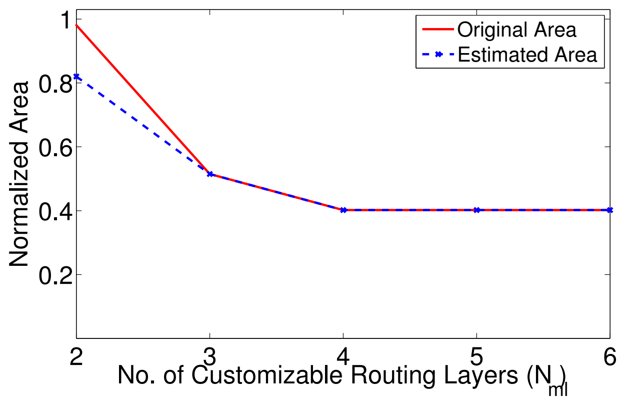
We experimented with two different suites of benchmark circuits that consisted of homogeneous and heterogeneous circuits. In case of homogeneous cir-



(a) Small Block

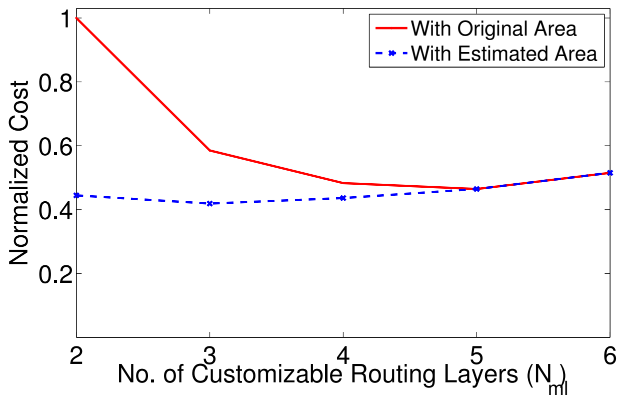


(b) Medium Block

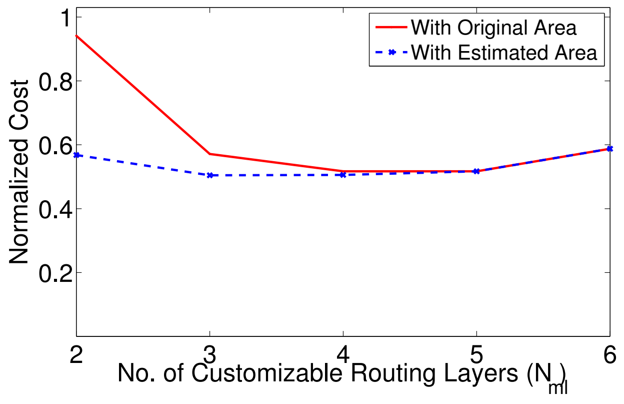


(c) Large Block

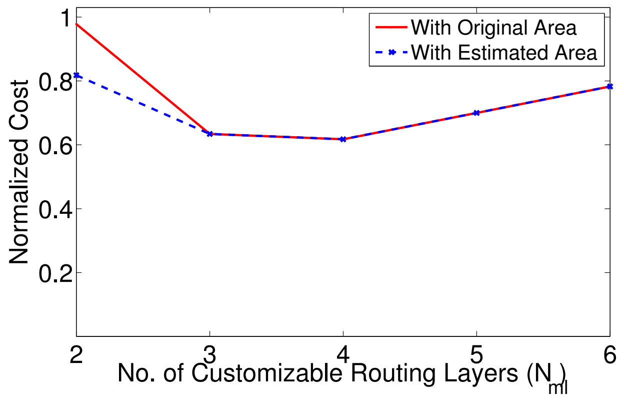
Figure 5.11: Estimated die-area with use of an intelligent whitespace insertion algorithm (packed eASIC benchmarks)



(a) Small Block



(b) Medium Block



(c) Large Block

Figure 5.12: Estimated die-cost with use of an intelligent whitespace insertion algorithm (packed eASIC benchmarks)

cuits, minimum cost is achieved with the minimum number of routing layers. The minimum number of routing layers also lead to the minimum cost in heterogeneous circuits if the technology mapping is sparse. With a dense technology mapping, the minimum cost is achieved with three or four routing layers. The delay and power for both the homogeneous and heterogeneous circuits improve with additional routing layers. However, the improvement with more than four routing layers is not significant. The maximum improvement occurs in heterogeneous circuits with Small Blocks. In this case, 90% of the delay reduction, and 80% of the power reduction occurs with four routing layers.

The die-cost of MPSAs was compared against CBICs. Small circuits with a core area of up to $10mm^2$ in a 2-layer 45nm MPSA can be $10\times$ cheaper than a corresponding $2.8mm^2$ 45nm CBIC. For large designs with embedded macro blocks, the cost difference (between a 2-layer 45nm MPSA and a corresponding CBIC) is $2\times$. This cost advantage grows to $4\times$ for a 4-layer MPSA.

The CAD flow currently uses uniform whitespace insertion. This inflates area and cost when there are few routing layers. With the use of an intelligent whitespace insertion algorithm, the MPSA cost advantage with two routing layers can be improved by more than 50%. This would also make the difference between 45nm CBIC and 2-layer MPSA grow to $4\times$, and there would be little or no additional cost advantage to increase the MPSA from 2 layers to 4 layers.

Chapter 6

Via-Programmable Structured ASICs

6.1 Overview

In this chapter, we present experimental results for Structured ASICs in which all the metal masks are fixed and only the custom via masks are needed to configure the device (VPSAs). We conducted experiments using three different types of via-programmable routing fabrics: one of the fabric uses *crossover wiring*, another uses *jumper wiring*, and the third type of fabric employs multiple metal routing layers but only uses a *single via* layer for customization. We also studied the impact of the logic block pin positions on the performance of VPSAs. We explored three different schemes for pin positions. These include assigning pins at the periphery of logic blocks, on two intersecting diagonals across the logic block, and on a single diagonal across the logic block. Using the best pin position scheme for each routing fabric, we study the power, delay, area, and cost trends for VPSAs. We also compare these trends to MPSAs. Finally, we outline some of the limitations of our detailed router when routing large heterogeneous circuits and mention some of the techniques that we explored to improve its performance.

6.2 Experimental Setup

We use the CAD flow described in Chapter 4 to conduct the experiments. In the following subsections, we describe the benchmarks and our approach for technology mapping, and the method used to calculate the physical sizes of the logic blocks.

6.2.1 Benchmark Circuits and Technology Mapping

We conducted the experiments for VPSAs using nineteen largest homogeneous circuits from the MCNC benchmark suite.[1] As described in Chapter 5, one of the circuits, s38584.1, contains a net with more than 3000 pins which was too large for the global router. We chose to exclude the benchmark rather than modify it or the CAD flow. We could not use heterogeneous circuits from the eASIC benchmark suite ([3]) for VPSA experiments. This limitation is discussed in Section 6.6.

We perform technology mapping as described in Section 5.2.1.

[1]Characteristics of these benchmarks are shown in Appendix A.

6.2.2 Logic Block Types and Dimensions

We used the same types of logic blocks (number of I/O pins) that were used in the MPSA experiments with homogeneous circuits (Table 5.1).

The layout area for a particular logic block depends upon the contents (number of gates) and the effort of the layout artist, both of which are hard to estimate precisely. Instead, we sweep the area across a range of values. We determine minimum and maximum area and use three equally spaced points within that range. The maximum cell area corresponds to a logic block that is laid out with little effort, and therefore has a sparse layout. We define this logic block as “Sparse”. The minimum cell area corresponds to a hand-crafted cell that has a dense layout. We define this as “Dense”. Between these values, we define the mid-point as “Medium”.

We determine maximum and minimum cell area values for each logic block type (I/O count). To estimate the area of a “Sparse” implementation, we find the smallest logic block size that can be routed with a two-layer Crossover fabric without any whitespace. We determine this value for each benchmark circuit and then select the largest value of the set. Similarly, we estimate the area of “Dense” implementation by calculating the smallest logic block size that is routable (using whitespace) with a two-layer Crossover fabric. In Table 6.1, we show the layout area values (in units of half metal pitches) used in our experiments for different logic block types. The minimum and maximum cell areas for each individual MCNC benchmark circuit are shown in Appendix C.

Table 6.1: Logic blocks used in experiments

Type		**Block Layout Area(Width×Height)**		
IN	***OUT***	***Dense***	***Medium***	***Sparse***
2	1	12×12	20×20	28×28
4	2	20×20	36×36	50×50
6	3	22×22	42×42	62×62
8	4	26×26	50×50	72×72
10	5	30×30	56×56	82×82
12	6	32×32	62×62	92×92
14	7	38×38	72×72	104×104
16	8	40×40	74×74	106×106

6.3 Fixed-Metal Interconnect Fabrics

In VPSAs, all the metal segments in the routing layers are pre-designed. Thus, the metal masks are fixed and only the via masks need to be generated to customize the device. This reduces the device cost, but may increase the delay and power consumption since the device is not as flexible as an MPSA. We conduct our experiments using three different types of fixed-metal fabrics that have been proposed in [96, 97]. We define these fabrics as *Jumper*, *Crossover*, and *SingleVia* fabrics. Each of these fabrics is described in detail below.

6.3.1 Jumper Fabric

The Jumper fabric is a simple interconnect fabric containing routing segments and jumpers on each layer. The routing segments span the entire logic block whereas the jumpers are small segments that are orthogonal to the routing segments and are used to extend routing segments in adjacent layers. The routing segments (and jumpers) of two adjacent layers are orthogonal to each other and form a crossbar structure. The fabric is configured by placing vias at desired points between intersecting segments (routing segments or jumpers) in adjacent layers. It is possible to include long routing segments in the fabric that span more than one logic block. The metal architecture for two adjacent layers of a jumper-based fabric for a tile of 2×2 logic blocks is shown in Figure 6.1.

The jumper fabrics used in our experiment contain two types of routing segments: *regular segments* that span a single logic block, and *long segments* that span four logic blocks. The long segments are staggered across the tile. Based on the ratio of regular and long segments, the following two variants of the jumper-based fabric are used in the experiments:

1. *Jumper20*: 20% routing segments are long segments

2. *Jumper40*: 40% routing segments are long segments

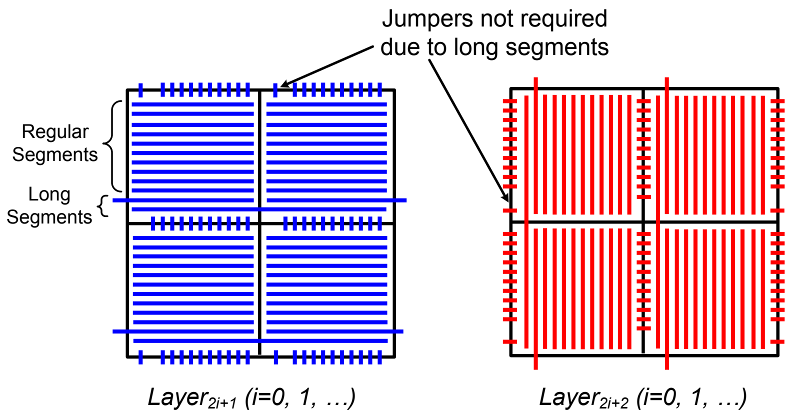

Jumpers not required due to long segments
Regular Segments
Long Segments
Layer$_{2i+1}$ (i=0, 1, …)
Layer$_{2i+2}$ (i=0, 1, …)

Figure 6.1: Jumper-based routing fabric

6.3.2 Crossover Fabric

Crossover fabric is a routing structure which does not contain any dedicated jumpers and only contains routing segments. In one layer, all the routing segments passing over a logic block are orthogonal to the segments passing over the neighboring logic blocks. The segments in two adjacent layers are also orthogonal to each other and form a crossbar structure. Figure 6.2 illustrates the Crossover fabric.

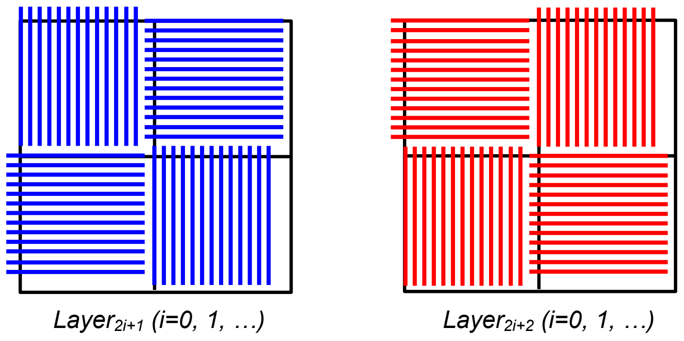


Figure 6.2: Crossover routing fabric

The Crossover fabric has certain advantages over the Jumper fabric [96, 97]. First, since there are no dedicated jumpers, it can accommodate an additional routing segment in the same area. Second, extending a segment along the same direction in a Jumper-based fabric requires the signal to go through two vias (jumper ends), whereas in Crossover fabric it only goes through a single via.

6.3.3 SingleVia Fabric

In both Jumper and Crossover fabrics, all the metal masks are fixed and all the via masks are customizable. The SingleVia fabric has been proposed to further reduce the VPSA cost by making only a single via layer configurable and fixing all the other via masks [96, 97].

The basic structure of the SingleVia fabric is as follows. The metal segments in Layer 1 and Layer 2, the first two routing layers of the fabric, form a Crossover structure.The via layer between Layer 1 and Layer 2 is configurable. Each additional layer consists of staggered long segments that connects to the second routing layer with fixed vias at each end. All the long segments in a layer have the same orientation (horizontal or vertical), and segments in adjacent layers are orthogonal. Not all the segments on second routing layer can be used for routing; some of these segments, known as *accessing segments*, are only used to provide a connecting path for segments from layer three onwards to the configurable via layer. The number of accessing segments depends on the number of routing layers above layer two, the length of segments in these layers, and the orientation of long segments relative to the corresponding accessing segment.[2] Figure 6.3 illustrates the SingleVia fabric with four routing layers (Layer 1 is not shownl it is similar but orthogonal to Layer 2).

As illustrated in Figure 6.3, the SingleVia fabric used in our experiments uses long segments that span four logic blocks.

6.4 Impact of Logic Block Pin Positions

The logic block pins (inputs or outputs of a logic block) connect to the first layer of the routing fabric. The VPSA performance can be affected by the logic block

[2]If a long segment is parallel to layer-2 segments and the long segment spans *n* logic blocks, then every *n*th segment per logic block is an accessing segment. If a long segment is perpendicular to the accessing segment, then only two segments per logic block act as accessing segments.

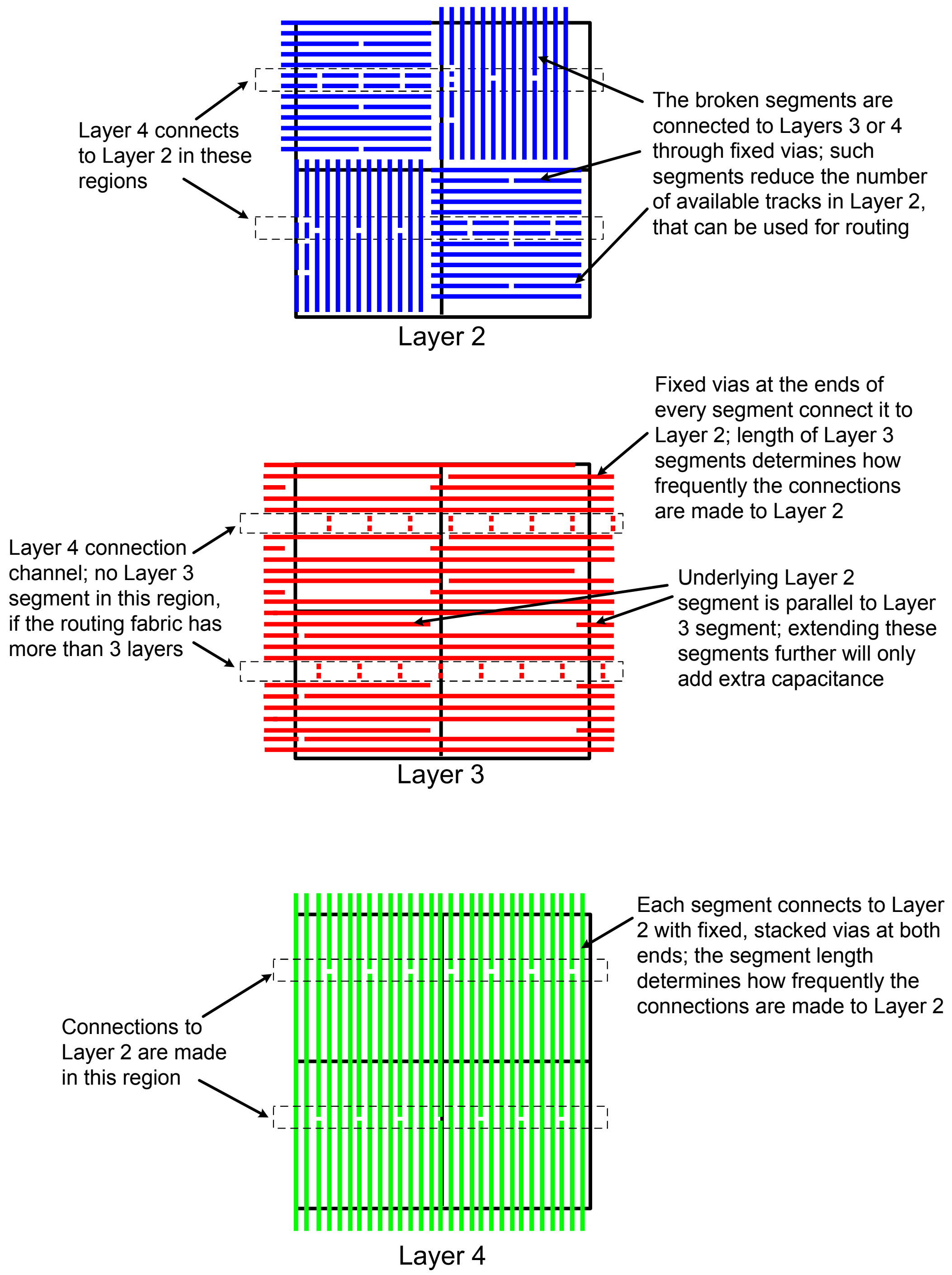
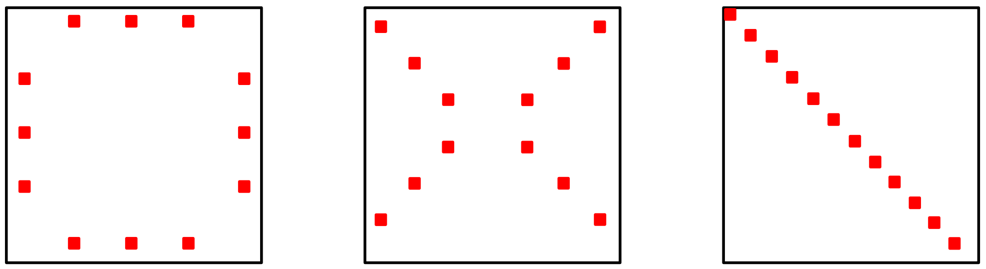

Layer 4 connects to Layer 2 in these regions
The broken segments are connected to Layers 3 or 4 through fixed vias; such segments reduce the number of available tracks in Layer 2, that can be used for routing
Layer 2
Fixed vias at the ends of every segment connect it to Layer 2; length of Layer 3 segments determines how frequently the connections are made to Layer 2
Layer 4 connection channel; no Layer 3 segment in this region, if the routing fabric has more than 3 layers
Underlying Layer 2 segment is parallel to Layer 3 segment; extending these segments further will only add extra capacitance
Layer 3
Each segment connects to Layer 2 with fixed, stacked vias at both ends; the segment length determines how frequently the connections are made to Layer 2
Connections to Layer 2 are made in this region
Layer 4

Figure 6.3: SingleVia routing fabric

pin positions. Depending on the pin organization, a number of tracks that connect to the pins cannot be used for routing any other signal. On one extreme, all the pins can be assigned on to a single track by dividing the track into sub-segments such that every sub-segment connects to only one pin. The other extreme is to assign one pin per track. In the first case, although most of the tracks can be used for routing, each pin can only connect to a small number of tracks of the second routing layer. This can lead to reduced routability. In the second case, connecting to the pins can occupy many tracks of the first layer, which can create routability problems. It is not clear which scheme is better.

To study the impact of the pin positions on the performance of VPSAs, we investigated three different pin position schemes, which we refer to as *PinsPer*, *PinsX*, and *PinsDiag*. In the PinsPer scheme, the pins are assigned at the periphery of the logic block such that each side of the block has approximately the same number of pins. Pins are distributed uniformly on each side and the routing segments with more than one pin are subdivided into smaller sub-segments such that each subsegment connects with only one pin. In the PinsX scheme, pins are assigned on two intersecting diagonals (in the shape of the letter X). A maximum of two pins are assigned per track in the PinsX scheme, which again requires subdividing the segment. Finally, in the PinsDiag scheme, pins are assigned on a diagonal with one pin occupying an entire routing track. These pin position schemes are illustrated in Figure 6.4 for a logic block with 12 pins.

To find the best pin position scheme, we evaluated each pin position schemes described above with Crossover, Jumper20, and SingleVia fabrics. We looked at



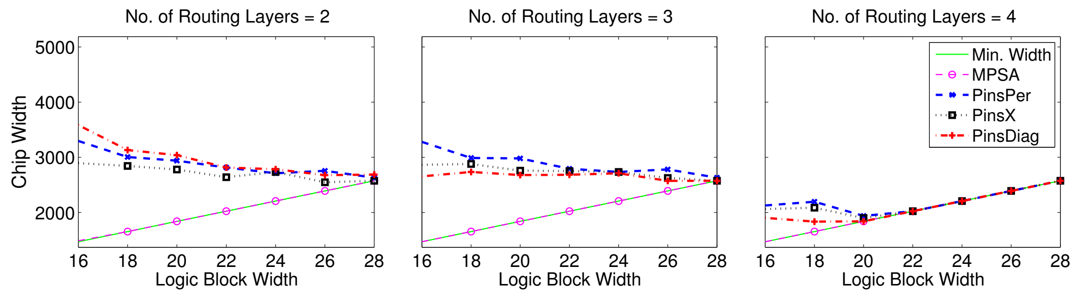
(a) PinsPer **(b)** PinsX **(c)** PinsDiag

Figure 6.4: Different schemes for logic block pin positions

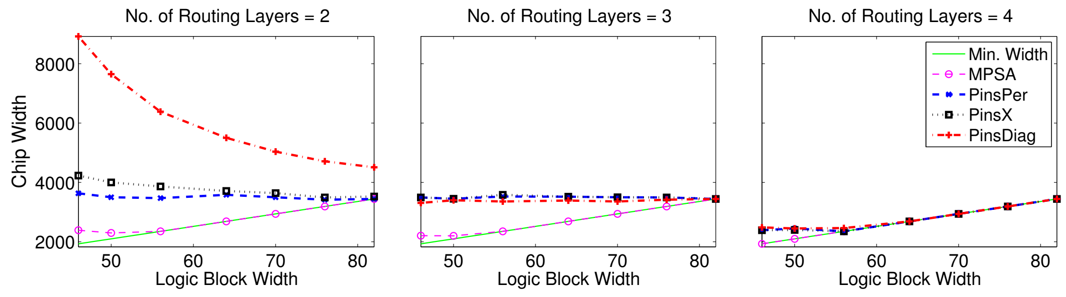
different logic block types (number of I/O pins), as shown in Table 6.1. We analyzed each logic block type with a range of logic block sizes (layout area) and with routing fabrics consisting of two, three, and four routing layers. The resulting trends[3] for the most congested MCNC benchmark circuit (*pdc*) are shown in Figures 6.5, 6.6, and 6.7 for Crossover, Jumper20, and SingleVia fabrics respectively. In each plot, the horizontal axis shows logic block width (in units of wire half-pitches) and the vertical axis represent the width of the square VPSA device that is obtained after performing place and route (logic block width $\times$ placement grid-x). Each plot shows the performance of three pin position schemes. For comparison, we also include the minimum width (no whitespace) and the width of an MPSA device. The MPSA device width indicates the lower bound for the width of VPSA devices.

There are three main observations. First, there is a significant difference between the performance of different pin position schemes, especially when the routing fabric consists of only two layers. When there are more routing layers

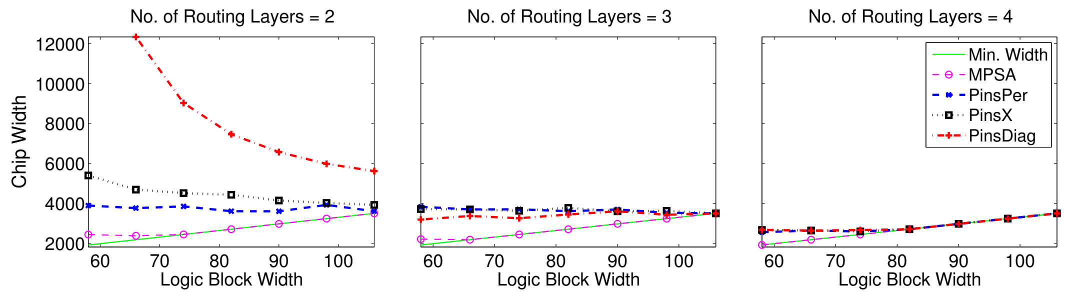
[3]The trends for only three logic blocks types (2-input, 1-output; 10-input, 5-output; and, 16-input,8-output) are shown here. The results for remaining types are shown in Appendix B.



(a) 2-input, 1-output Logic Block

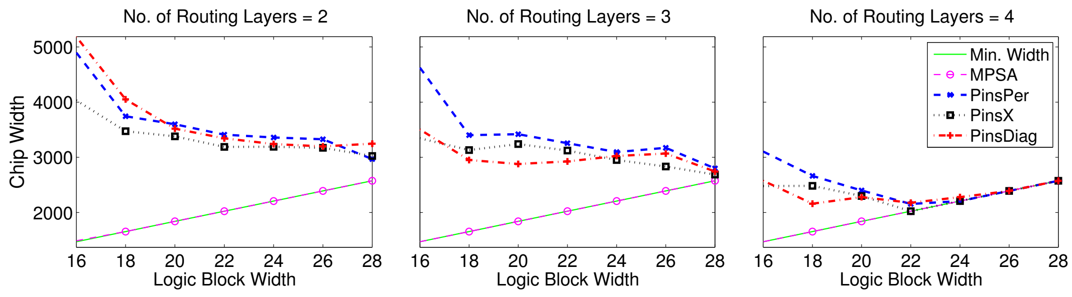


(b) 10-input, 5-output Logic Block

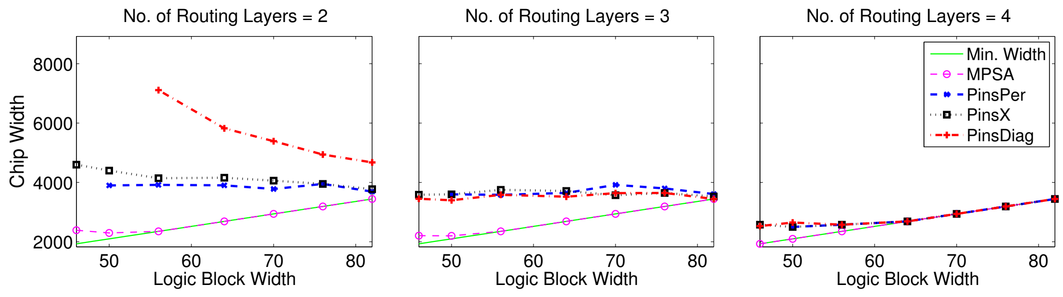


(c) 16-input, 8-output Logic Block

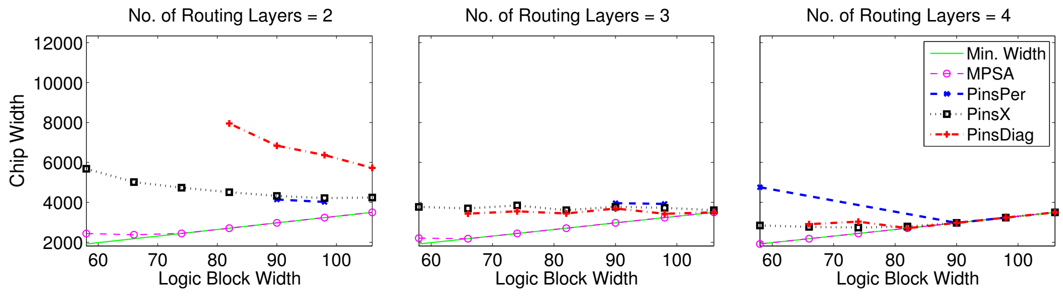
Figure 6.5: Impact of logic block pin positions on *Crossover* routing fabric. The plots show trends for the most congested circuit — *pdc*.



(a) 2-input, 1-output Logic Block

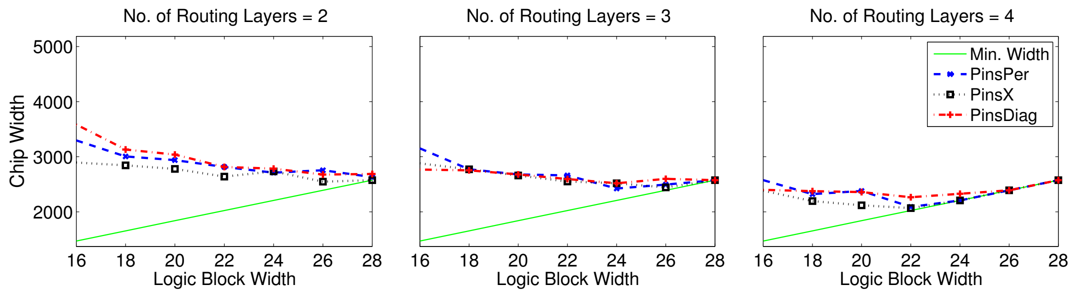


(b) 10-input, 5-output Logic Block

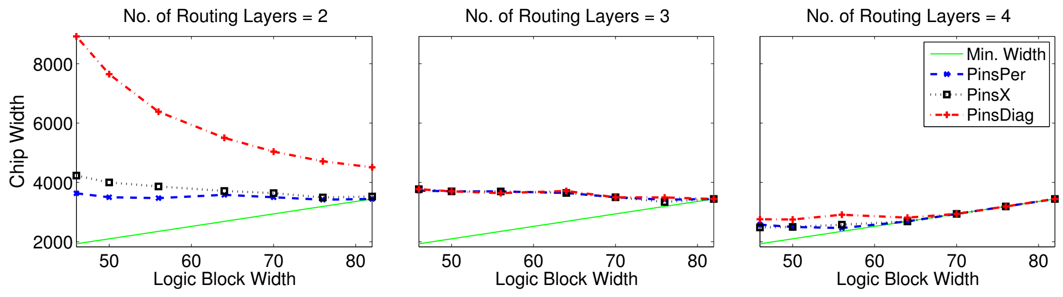


(c) 16-input, 8-output Logic Block

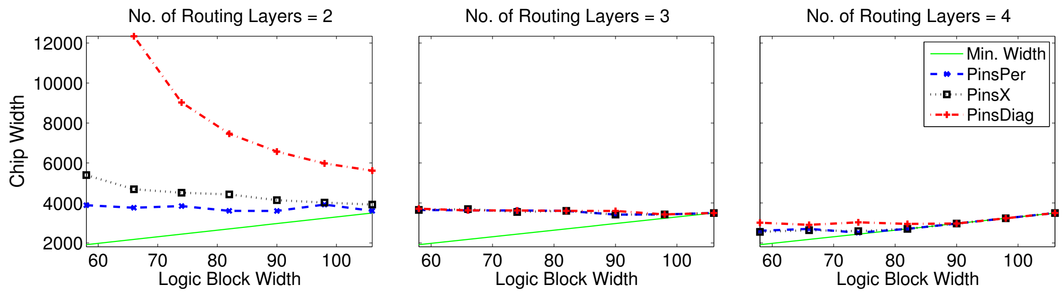
Figure 6.6: Impact of logic block pin positions on *Jumper20* routing fabric. The plots show trends for the most congested circuit — *pdc*.



(**a**) 2-input, 1-output Logic Block



(**b**) 10-input, 5-output Logic Block



(**c**) 16-input, 8-output Logic Block

Figure 6.7: Impact of logic block pin positions on *SingleVia* routing fabric. The plots show trends for the most congested circuit — *pdc*.

available in the routing fabric, the difference is not as significant. This trend can be seen in Figures 6.5 and 6.6. The reason is that the number of available routing resources (metal segments) are limited when there are only two routing layers in the routing fabric. Different pin position schemes can result in a significant difference between the device widths, highlighting the importance of choosing the correct scheme for the correct circumstances. With more routing layers, or wider logic blocks, the number of routing resources increase and the impact of different pin position schemes is not as significant.

Second, the number of routing layers in the routing fabric can affect the performance of a particular pin position scheme. Figures 6.5 and 6.6 show that with two routing layers, the device width with PinsDiag scheme is significantly worse than PinsPer and PinsX schemes, especially when the logic block contains more than three pins. For example, in case of a 16-input, 8-output logic block with two-layer Crossover fabric (Figure 6.5c), the difference between the device widths with PinsDiag and PinsPer schemes is more than $3\times$. Similarly, for Jumper20 fabric, there is a big gap between PinsDiag, and PinsX and PinsPer schemes. In fact, in many cases, the use of PinsDiag scheme with two-layer Jumper20 fabric makes the circuit unroutable. However, the PinsDiag fabric performs better with both Crossover and Jumper20 fabrics when there are more than two layers in the routing fabric. This is because the PinsDiag scheme connects an entire routing track to each pin. The use of PinsPer and PinsX schemes allow multiple pins per routing track, and therefore these schemes perform better when there are only two routing layers. As the number of routing layers increases, the ability of PinsPer and PinsX schemes to connect pins to only a small number of layer-two segments becomes a limiting factor and PinsDiag scheme performs better in this case.

Finally, the third observation is that the performance of a particular pin position scheme is also dependant on the architecture of the routing fabric. For two-layer Crossover fabric, PinsPer scheme has the best performance except when the logic block has only three pins (Figure 6.5). As mentioned earlier, the PinsPer scheme aims to distribute pins equally on each side of the logic block; with three pins, the PinsPer scheme ends up with a single pin per routing track. Hence, the PinsX scheme which assigns two pins per routing segment, performs better in the case of three pins. Unlike the Crossover fabric, the PinsPer scheme is not routable with two-layer Jumper20 fabric in many cases (Figure 6.6). The results for SingleVia fabric are shown in Figure 6.7. The plots for two-layer SingleVia fabric and two-layer Crossover fabric are similar because a two-layer SingleVia fabric is identical to two-layer Crossover fabric. It can be seen that unlike the Crossover and Jumper20 fabrics, the PinsDiag scheme does not perform better than PinsPer and PinsX schemes for three- and four-layer SingleVia fabrics. This is because the third and fourth layers in SingleVia fabric are not completely flexible; the connections can only be made at end points and they also consume many segments from layer two (accessing segments). The PinsDiag scheme consumes many routing segments of the flexible first layer, hence it degrades the performance.

For our experiments, we used the best-case pin position scheme for each routing fabric depending on its architecture and the number of routing layers. Table 6.2 shows the pin position schemes used with different fabrics in our experi-

Table 6.2: Pin position schemes used in the experiments

Routing Fabric	Pin Position Schemes		
	PinsPer	**PinsX**	**PinsDiag**
Crossover	Pins $>$ 3, and Routing layers $=$ 2	Pins $=$ 3, and Routing layers $=$ 2	Remaining cases
Jumper20 *Jumper40*	Not used	a) Routing layers $=$ 2, or b) Pins cannot be assigned using PinsDiag scheme (logic block is too small to accommodate all the pins with 1 pin/track)	Remaining cases
SingleVia	Not used	All cases	Not used

ments.

6.5 Performance and Cost Trends

In this section, we present the experimental results for fixed-metal routing fabrics. We look at the trends for routing fabrics with two, three, and four metal layers. With three metal layers, the number of wire segments along one direction (horizontal or vertical) are roughly twice as large as the other direction. Such asymmetric fabrics may be useful in the following cases:

1. logic blocks have a large number of input/output pins and many metal segments from the first layer are consumed in making connections to these pins;

2. there is too much congestion; or,

3. when logic blocks have non-square aspect ratios (this work only considers

square aspect ratios).

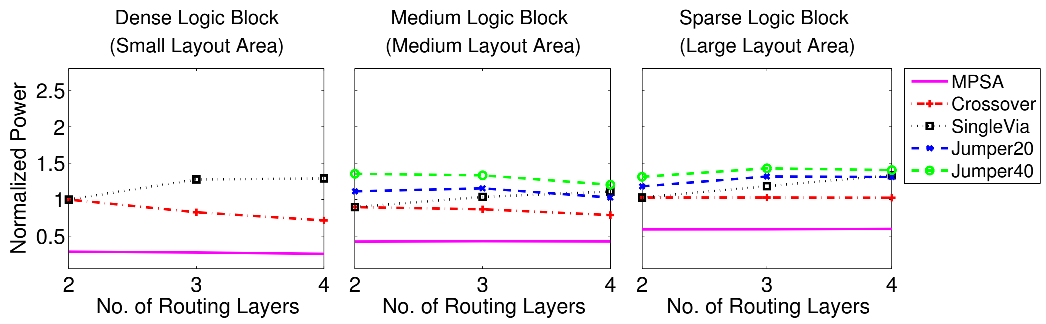
We present power, delay, area, die-cost trends for four different VPSA routing fabrics. We also compare the VPSA performance against MPSAs. The MPSA data is calculated as in Section 5.2 (we use the same layout area values and number of routing layers as used for VPSAs, but do not perform detailed routing).

6.5.1 Power Results

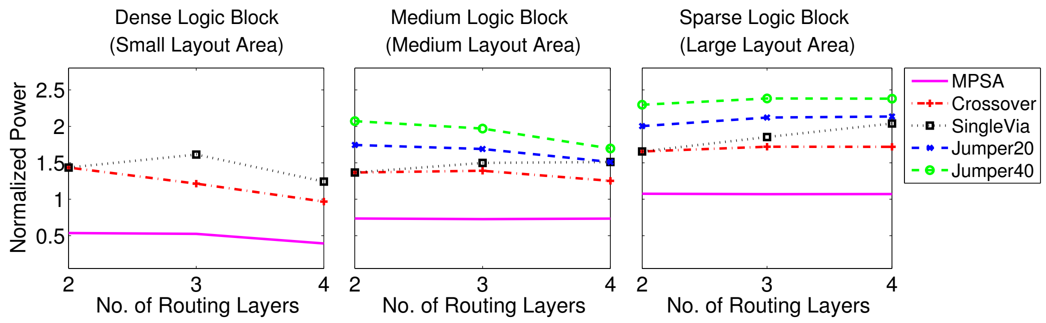
To look at the effect of a certain routing architecture on VPSA performance, we first look at the dynamic power dissipated in the interconnect. The results for four different VPSA routing fabrics are shown in Figure 6.8 along with the results for an MPSA routing fabric. The horizontal axis in each graph shows the number of metal layers that can be used for routing. Depending on the type of the fabric, different number of via layers are customizable. For example, for the Crossover and Jumper fabrics, n routing layers would imply that there are $n-1$ configurable via layers; for SingleVia fabric, the number of configurable via layers is always one, irrespective of the number of routing layers. The vertical axis in each of the plots shows interconnect power that we estimate from total interconnect capacitance. The values in all the graphs are normalized to the interconnect power of a VPSA having two-input, one-output "Dense" logic blocks and one customizable via layer with the Crossover routing fabric. We show separate plots for different logic block types and different block layout areas. Figures 6.8a, 6.8b, and 6.8c show the plots corresponding to the first, fifth, and last rows of Table 6.1. Some of the congested circuits could not be routed with Jumper20 and Jumper40 fabrics when “Dense” logic blocks were used, so the entire curve is omitted. The trends for the remaining logic block types are similar and the plots are shown in Appendix D.

There are three main observations. First, there is a significant range in interconnect power consumption of different via programmable routing fabrics. The SingleVia, Jumper20, and Jumper40 fabrics dissipate 0–85%, 15–30%, and 28–54% more power than Crossover fabric respectively. The Jumper40 fabric also dissipates 7–22% more power than the Jumper20 fabric. The relative power dissipation between the different fabrics is mostly similar across different logic block layout densities (each individual plot in Figures 6.8a, 6.8a, and 6.8c). It is interesting to see that a small architectural change, varying the number of long segments from 20% to 40%, can affect the power consumption by 7% to 22%.

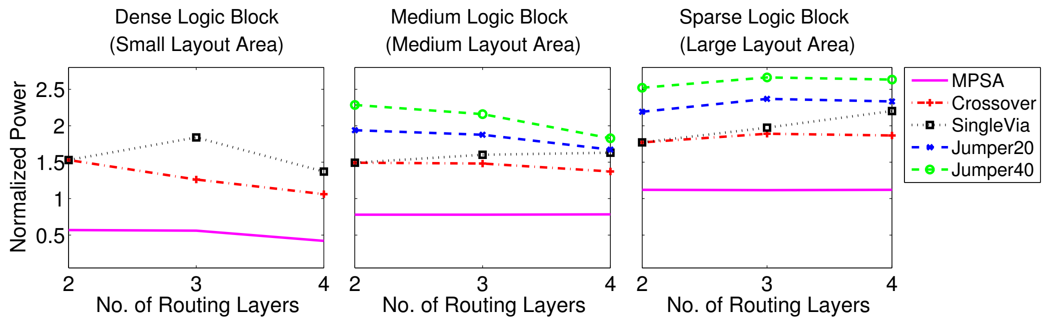
Second, except for the SingleVia fabric, the power dissipation generally improves as more routing layers are available and this improvement diminishes as the layout area of the logic block increases. This can be seen from the interconnect power of Crossover across the three plots in Figure 6.8a. For “Dense” logic block (left plot), the power consumption with three customizable via layers is 29% lower than the power consumption with a single customizable via layer. For “Sparse” logic block (right plot), this variation is only 0.3%. A similar conclusion can be drawn from Figures 6.8b and 6.8c. The reason for reduction in power dissipation with more routing layers is due to the fact that, with more routing layers, a design can be successfully routed with less whitespace, and connections between logic blocks can be made with shorter nets. In the case of the SingleVia fabric, the



(a) 2-input, 1-output logic block



(b) 10-input, 5-output logic block



(c) 16-input, 8-output logic block

Figure 6.8: VPSA power trends

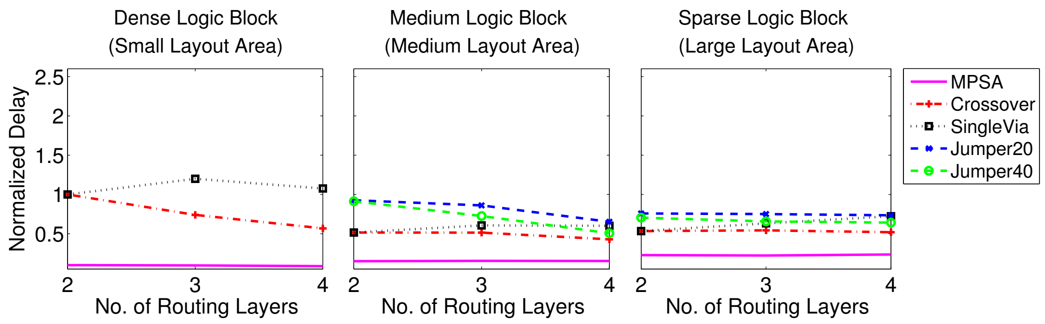
power dissipation does not improve with more routing layers. There are two reasons for this. First, the third and fourth routing layers in the SingleVia fabric are restrictive; the connections can only be made at the end points of each segment and the connections also consume many segments from layer two (accessing segments). Second, it is possible that some of the accessing segments are used to make local connections (e.g., connection between two layer-one segments). This can increase the power consumption since these segments are connected to layer-three or layer-four segments with fixed vias.

Finally, the third observation is regarding the gap between MPSAs and VPSAs. It can be seen that using only via layers for customization increases power dissipation. The Crossover fabric, which has the smallest power consumption among the VPSA fabrics, has a power dissipation that is 1.5–3.5$\times$ larger than an MPSA. The gap between MPSAs and VPSAs reduces when the number of routing layers increase. The gap also reduces when the layout area of the logic blocks increase.

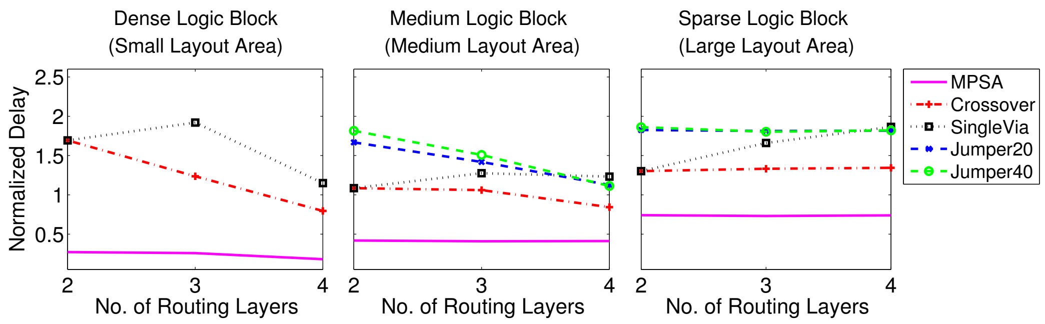
6.5.2 Delay Trends

The delay results are shown in Figure 6.9. These plots are similar to Figure 6.8, except that the vertical axis now shows interconnect delay. As in the case of power, the plots are normalized to the interconnect delay of a two-layer Crossover routing fabric with two-input, one-output "Dense" logic blocks. As mentioned in Chapter 4, average net delay is used to estimate interconnect delay. The main observations from the delay plots are as follows.

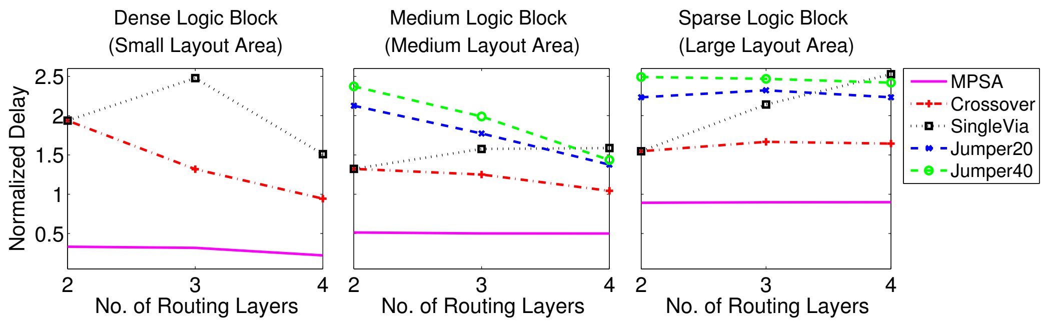
First, like power, delay has significant range across different routing fabrics.



(a) 2-input, 1-output logic block



(b) 10-input, 5-output logic block



(c) 16-input, 8-output logic block

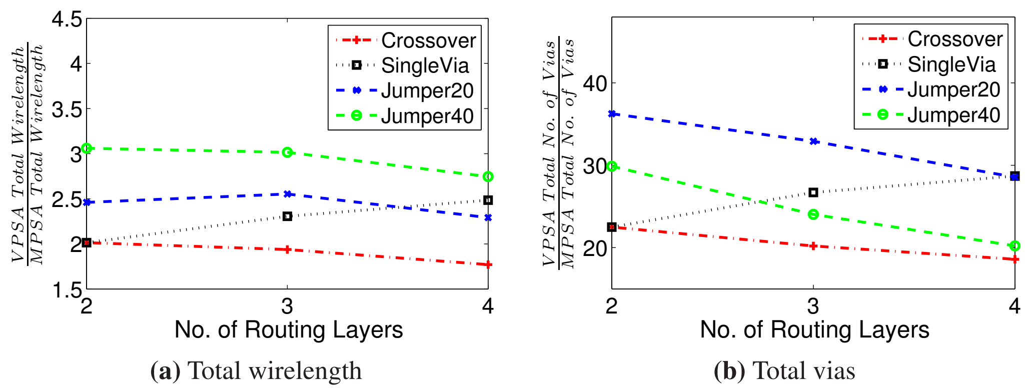
Figure 6.9: VPSA delay trends

The SingleVia, Jumper20, and Jumper40 fabrics are 0–89%, 32–80%, and 18–79% slower than the Crossover fabric respectively. As in the case of power, these trends are mostly independent of the number of available routing layers.

Second, also as in the case of power, the increase in the number of routing layers generally improve the delay, except for the SingleVia fabric. This is also because more routing layers reduce congestion, allowing the circuits be routed with less whitespace. This reduces the wirelength for each net and improves delay. The additional layers in the SingleVia fabric are not as flexible as in other fabrics and some of the long segments can be used to make local connections. This results in an increase in delay. A router that intelligently selects the routing segments from layer-two can improve the delay performance of the SingleVia fabric.

A third observation is regarding the difference between the performance of the Jumper20 and Jumper40 routing fabrics. Jumper20 has fewer long segments than Jumper40 and it was consistently better in terms of power. However, in terms of delay, Jumper40 fabric is better than Jumper20 in some cases (e.g., Figure 6.9a). This shows an interesting trade-off; having more long segments in a fabric can improve delay but can make power worse.

Finally, the last observation is regarding the comparison of VPSA and MPSA interconnect delay. As in the case of power, the VPSA delay is worse than that of an MPSA but it improves as the layout area increases. However, it is important to observe the magnitude by which the delay gets worse in VPSAs. The increase in the delay for the Crossover fabric compared to MPSAs ranges from 2–10$\times$. To investigate this, we looked at the total wirelength and total number of vias for each



(**a**) Total wirelength (**b**) Total vias

Figure 6.10: MPSA and VPSA delay comparison for a 2-input, 1-output logic block with medium layout density

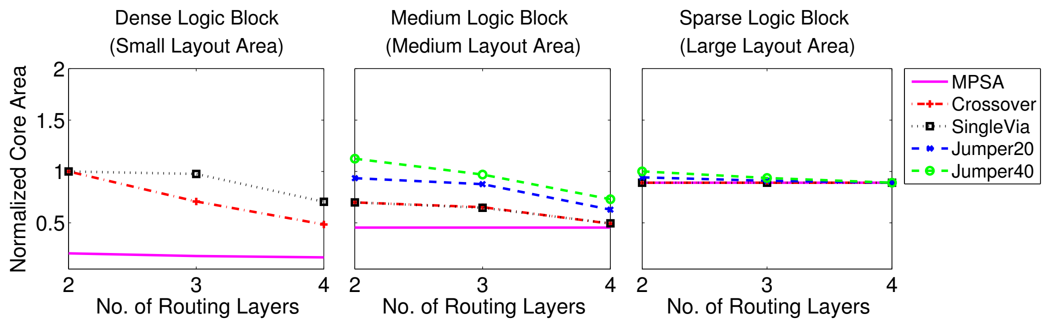
of the fabrics relative to the MPSA. The plots for two-input, one-output Medium logic block are shown in Figure 6.10. It can be seen that with two routing layers, the increase in wirelength is only 2× to 3×, whereas the number of vias has a dramatic increase of 22× to 36×. This causes the delay to increase significantly. The reason for this behavior is that VPSAs need to go through a via whenever a wire has to extend in any direction; depending on the architecture of the routing fabric, it may have to go through more than one via in order to extend by just one more segment (e.g., in case of Jumper20 and Jumper40 routing fabrics).

The relatively poor performance of VPSAs compared to MPSAs may be improved in two ways. First, double vias could be used to make connections between metal segments at the expense of area. Second, VPSAs may respond more strongly to buffer insertion due to the larger resistance and capacitance of the interconnect.

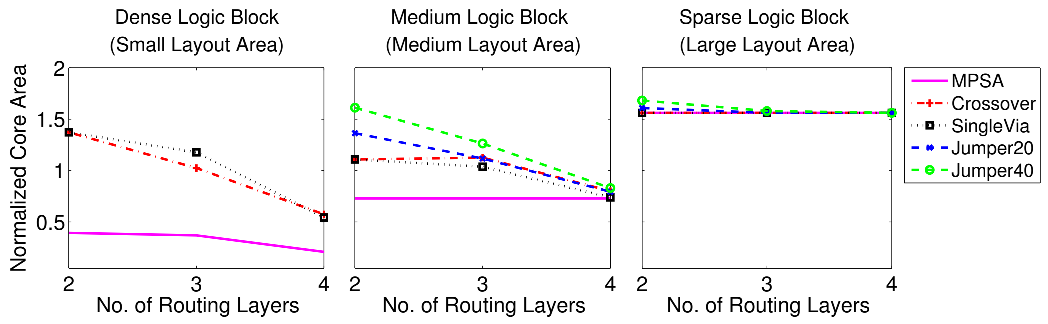
6.5.3 Area Trends

The plots for core area are shown in Figure 6.11. The important observations are as follows. First, like power and delay, there is a significant range in the core area among the different routing fabrics, especially when there are a small number of routing layers. The core areas with the SingleVia, Jumper20, and Jumper40 fabrics are 0–46%, 0–34%, and 0–60% larger than the core areas with the Crossover fabric, respectively.

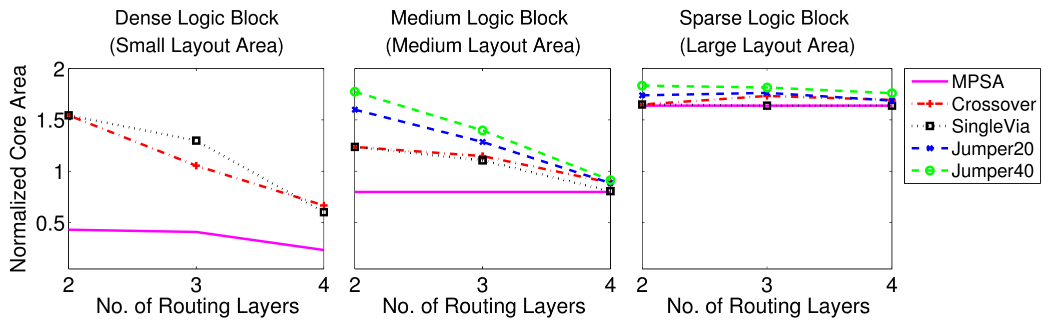
Second, the availability of more routing layers reduces the core area. However, this reduction becomes small as the logic block layout area increases. For example, from Figure 6.11a, it can be seen that the core area for the Crossover fabric for the "Dense" logic block reduces by 50% when the number of routing layers increases from two to four; however there is no area reduction when the "Sparse" logic block is used. Similar trends can observed for other logic block types (Figures 6.11b and 6.11b). Another interesting observation is that with fewer routing layers the core area with "Dense" logic blocks is larger than the core area with "Medium" logic blocks. This is because there is a lot of congestion when the number of routing layers, or the layout area, is small. We are using uniform whitespace allocation in our CAD flow, which inserts whitespace everywhere instead of targeting the congested regions. This results in a significant increase in the core area. The use of an "intelligent" whitespace allocation algorithm, one that inserts whitespace only at congested regions, may help to reduce this problem. In the absence of such an algorithm, there is very little advantage of having a logic block with small layout area, especially when the number of routing layers



(a) 2-input, 1-output logic block



(b) 10-input, 5-output logic block



(c) 16-input, 8-output logic block

Figure 6.11: VPSA area trends

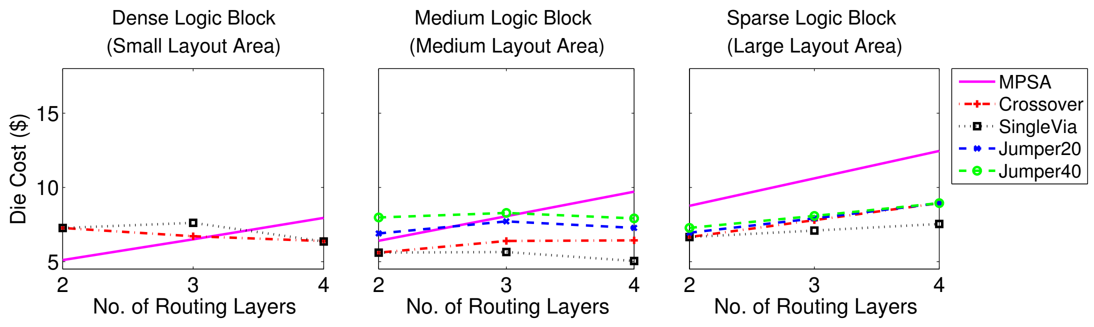
is small.

Finally, there is a significant area gap between MPSAs and VPSAs. The Crossover fabric has the least area among the different VPSA routing fabrics and its area is 1–5$\times$ larger than the MPSA area. The difference between the MPSA and the VPSA area is reduced as the number of routing layers increases or when the logic block layout area increases. Improving the CAD flow (e.g., congestion aware whitespace allocation) can also reduce the gap between MPSAs and VPSAs.

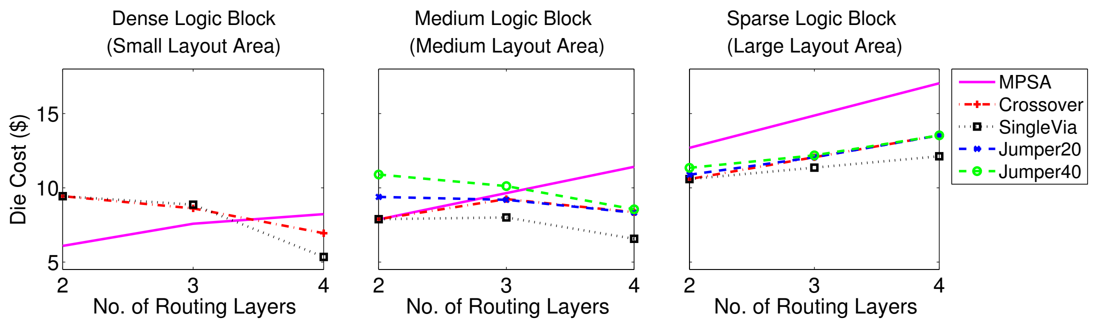
6.5.4 Cost Trends

Finally, we apply the cost model described in Chapter 3 to estimate the die-cost. However, because the MCNC benchmarks are small, we first scale the core area to a reasonable value. To do this, we scale all the core areas in such a way that the core area for an MPSA with two-input, one-output "Dense" logic blocks and having two routing layers is $10mm^2$. The resulting cost plots are shown in Figure 6.12. Instead of showing relative values, we show absolute die-cost values. The important observations are described below.

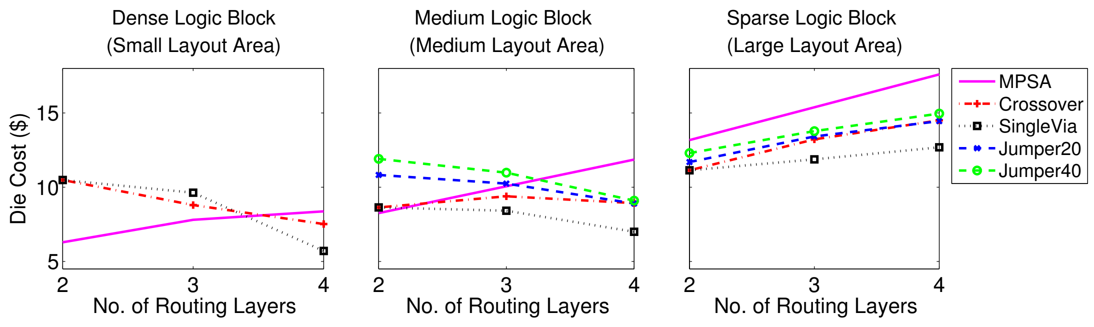
The routing fabric architecture affects die-cost. In some cases, the MPSA cost is better than VPSA cost whereas in other cases the VPSA is better than an MPSA, despite the fact that the area of a VPSA is never less than that of an equivalently capable MPSA. For example, the core area of the two-layer Crossover and SingleVia fabrics with "Medium" two-input, one-output logic blocks is 54% more than MPSA (Figure 6.11a); however, in terms of die-cost the VPSA is 15% less



(a) 2-input, 1-output logic block



(b) 10-input, 5-output logic block



(c) 16-input, 8-output logic block

Figure 6.12: VPSA cost trends

expensive than the corresponding MPSA. Similarly, in the same example with four routing layers, the VPSAs with SingleVia and Crossover fabrics are 48% and 34% cheaper than MPSAs, respectively. For densely laid out logic blocks, an MPSA is cheaper when fewer routing layers are available. With more routing layers, VPSAs become cheaper. This is easily seen in Figures 6.12a and 6.12b for the “Dense” logic block. As the layout area of each logic block increases (layouts become sparse), VPSAs become more cost effective even with fewer routing layers. It can also be seen that sparsely laid out logic blocks may lead to a more cost effective die than densely laid out logic blocks; this may be caused by inferior whitespace allocation performed during placement.

The cost plots in Figure 6.12 also demonstrate the cost effectiveness of the SingleVia fabric. In most cases, the SingleVia fabric results in similar core area compared to the Crossover fabric, however, the die-cost is significantly lower, especially with four routing layers. This is because the cost of the mask-set is the dominant factor that determines die-cost. In SingleVia fabrics, only a single mask needs to be customized which makes SingleVia fabrics superior to other fabrics in terms of cost.

6.6 Limitations of Detailed Routing

Our detailed router could not route the large heterogeneous circuits from the eASIC suite in a reasonable amount of time. There were two main issues. First, the time to route a single net sometimes becomes very large, which causes a single routing iteration to take about 36 hours. Second, all the congestion could not be removed within eight routing iterations for a range of different global routing constraints.

We tried several techniques to improve the runtime of the detail router. These included pruning the router search space, changing the net ordering scheme, and modifying the rip-up and re-route strategy. To prune the search space, we only put the 10 lowest cost neighbors of a segment on the heap during wavefront expansion, compared to the original algorithm which puts all the neighbors on the heap. We also changed the net ordering such that the longest nets are routed first. Long nets take the most time to route, and routing them early, when there is little congestion, should improve the runtime. Finally, after each routing iteration, we ripped-up and re-routed only those nets that pass through congested global cells rather than ripping-up and re-routing all the nets.

After making the above changes, the runtime for a single routing iteration reduced to about 7 hours (for the smallest eASIC circuit). However, the circuit could still not be routed successfully. This suggests that an algorithm different from the PathFinder algorithm might be more appropriate for performing detailed routing of such large circuits, at least when routing has to be done many times in order to find smallest routable grid size. This is an avenue for future research.

6.7 Summary

In this chapter, we studied the power, delay, area, and die-cost trends for VPSAs. We investigated three types of routing fabrics: one that uses crossover wiring (Crossover), another that uses jumper wiring (Jumper), and a fabric that uses multiple routing layers but in which only a single via layer can be configured (Single-Via). We also explored three different schemes for the logic block pin positions.

The results show that the logic block pin positions have a significant impact on the VPSA performance. In general, with limited routing resources (e.g., fewer metal layers), we found that it is better to assign multiple pins to a single metal track; whereas, in other cases it is better to assign a single pin to an entire metal track.

In terms of delay and power, the performance of Crossover and Jumper fabrics improves as the number of routing layers increases. As the layout area of logic blocks increases, the improvement diminishes. The performance of the SingleVia fabric generally gets worse as the number of routing layers increases. This is because the SingleVia fabric is not as flexible as the other fabrics. For each routing fabric, the die-area improves as the number of routing layers increases. However, as in case of delay and power, the improvement is insignificant when the logic blocks have a large layout area. The die-cost also improves with the number of routing layers, except logic blocks with a large layout area.

We also compared VPSAs against MPSAs. In terms of area, delay, and power, MPSAs perform better than VPSAs. The maximum difference between MPSAs and VPSAs for area, delay, power is $5\times$, $10\times$, and $3.5\times$ respectively. We also found that MPSAs are more cost effective than VPSAs when Dense logic blocks (small layout area) and two or three routing layers are used. However, in all other cases, MPSAs are up to $2\times$ more expensive than VPSAs. One of the reasons for the improved delay and power performance of MPSAs is the large number of serially connected vias in VPSAs. This also shows an important aspect of the interconnect architecture: for better performance, a routing fabric should use fewer serially-connected vias to make connections between the fixed metal segments. Hence, the Crossover fabric performs better than the Jumper fabric.

There is a significant difference between the performance of VPSAs using the different routing fabrics. A VPSA using the Crossover fabric has the best power, delay, and area performance. The power, delay, and area of VPSAs with other fabrics is up to 85%, 89%, and 60% worse than VPSAs built using Crossover fabric, respectively. However, in terms of die-cost, the SingleVia fabric is best. The die-cost of a VPSA using a SingleVia fabric is up to 36% lower than one using other fabrics. These results show that the interconnect architecture plays a very important role in the overall efficiency of VPSAs and it should be thoroughly researched.

Finally, the efficiency of VPSAs also depends on the associated CAD algorithms. For example, although we have not investigated this, we expect that the use of timing- or power-driven placement and routing algorithms may reduce the power and delay penalty of SingleVia fabrics. Similarly, our results suggest that a congestion-driven whitespace insertion, and buffer insertion, are both important. Logic blocks with dense layouts (small layout area) offer little advantage without such an algorithm, especially when the number of routing layers is small.

Chapter 7

Conclusions

7.1 Research Observations

In this dissertation, we have investigated power, delay, area, and die-cost trends for Structured ASICs. We focussed our attention on Structured ASIC interconnect and studied how the performance and cost is affected by the number of custom metal and/or via masks.

The die-cost of Structured ASICs depends on die-area as well as on the number of custom masks. The cost due to the custom masks is a significant component of die-cost. Increasing the number of custom masks must result in a considerable reduction of die-area to be cost-effective. For example, according to our cost model described in Chapter 3, a $95mm^2$ MPSA in $45nm$ with two configurable layers (two metal and two via) costs \$15; each additional custom layer in this MPSA must reduce the die-area by more than $15mm^2$ for the die-cost to remain

\$15. In contrast, the corresponding two-layer VPSA with a SingleVia fabric (two metal layers, but only a single via mask required for customization) that costs \$15 has an area of $110mm^2$; each additional metal routing layer in this VPSA only needs to save about $6mm^2$ of die-area to remain cost-effective.

Table 7.1 summarizes the important observations regarding the delay, power, area, and cost trends of MPSAs and VPSAs. The area, delay, and power improve as the number of routing layers increases for both MPSAs and VPSAs. However, as the layout area of the logic block increases, the improvement diminishes. In terms of die-cost, small MPSA dies (die-area $< 100mm^2$) have the lowest cost with 2 custom layers; the lowest cost in large MPSA dies (die-area$> 100mm^2$) is obtained with 3 to 4 custom layers. The lowest cost in VPSAs is achieved with 4 routing layers, except when sparse logic blocks (large layout area) are used. MPSAs are up to $10\times$, $3.5\times$, and $5\times$ better than VPSAs in terms of delay, power, and area respectively. However, in terms of die-cost, VPSAs are up to 50% less expensive.

The area, delay, power, and die-cost of VPSAs is sensitive to the logic block pin positions and the architecture of the fixed-metal routing fabric. We investigated three different types of VPSA routing fabrics: (1) Crossover fabric, (2) Jumper fabric, and (3) SingleVia fabric. Compared to a VPSA with Crossover fabric, VPSAs with Jumper or SingleVia fabrics perform up to 89% worse in terms of delay, up to 85% worse in terms of power, and up to 60% worse in terms of area. In terms of die-cost, VPSAs with the SingleVia fabric are up to 36% less expensive than VPSAs employing other types of fabrics. This significant difference

Table 7.1: Summary of MPSA and VPSA trends

		Delay Trends	**Power Trends**	**Area Trends**	**Cost Trends**
MPSAs (Chapter 5)	Homogeneous Circuits	Dense Logic Block: 3 or 4 custom layers for best performance Sparse Logic Blocks: 2 custom layers for best performance			Min. Cost: 2 custom layers
	Heterogeneous Circuits	Best performance with 3 or 4 custom layers; additional layers offer little advantage			Min. cost obtained with: - 2 custom layers with sparse tech. mapping - 3 or 4 custom layers with dense tech. mapping
VPSAs (Chapter 6)	Crossover Fabric	–	–	–	–
	Jumper Fabric	0–89% worse than Crossover	0–85% worse than Crossover	0–60% worse than Crossover	–
	SingleVia Fabric				SingleVia is 0–36% cheaper than other VPSA fabrics
MPSAs vs. VPSAs (Chapter 6)		MPSAs 1–10× better than VPSAs	MPSAs 1–3.5× better than VPSAs	MPSAs 1–5× better than VPSAs	MPSAs are cheaper only for Dense Logic Blocks with 2 or 3 layers. VPSAs are up to 50% cheaper in other cases

suggests that the topology of the VPSA interconnect fabric has a significant impact on the overall performance and cost of VPSAs; it should be more thoroughly researched.

Structured ASICs also provide a cost-effective alternative to CBICs. We compared the die-cost of MPSAs to CBICs. It was observed that small dies with a core area of up to $10mm^2$ in a 2-layer 45nm MPSA can be 10× less expensive than a corresponding $2.8mm^2$ 45nm CBIC. For large designs with embedded macro blocks, the cost difference (between a 2-layer 45nm MPSA and a corresponding CBIC) is 2×. This cost advantage grows to 4× for a 4-layer MPSA.

Finally, whitespace insertion is a key component of the CAD flow that can significantly improve the performance and cost of Structured ASICs. The CAD flow currently uses uniform whitespace insertion. This inflates area and cost when there are few routing layers. It was observed that in case of two routing layers, congestion-driven whitespace insertion can potentially reduce die-cost by more than 50%. In the absence of such an algorithm, logic blocks with dense layouts (small layout area) offer little advantage.

Most of the results from this research are useful for a Structured ASIC vendor who could use them to gain an insight into various trade-offs involved in the design of Structured ASICs. However, a Structured ASIC user could also benefit from some of the observations and can make an intelligent decision about selecting a particular Structured ASIC product. For example, an MPSA would be a better choice for a design that requires high performance, whereas a VPSA could be a better option when a low-cost implementation is the primary objective.

7.2 Contributions

The key contributions of the research are as follows:

- A cost model that relates die-area and number of routing layers of Structured ASICs. The cost model takes into account mask-set cost, wafer processing costs, device volume requirements, die-yield, die-area, and number of custom masks. It estimates the cost of a single Structured ASIC die. Traditionally, for CBICs and FPGAs, area has been used as a proxy to estimate the cost of the device. However, because of the large mask-set costs, the number of custom masks in a Structured ASIC have a significant impact on its cost. The cost model allowed us to demonstrate that in most cases, contrary to popular belief, the area savings due to the availability of more routing layers does not translate into a cost savings.

- A sensitivity analysis of the structured ASIC die-cost to various parameters such as device volume requirements and mask-set costs. In terms of volume requirement, the MPSA die-cost is more sensitive to per-customer volume (V_c) than total device volume (V_{tot}); a smaller V_c makes MPSAs more cost-effective than CBICs. Similarly, increasing mask-set costs also make MPSAs more economical than CBICs.

- A CAD flow that places and routes a benchmark circuit for a variety of Structured ASIC architectures. The CAD flow makes use of open-source CBIC global placement and global routing tools. It includes a custom detailed placer to perform legalization of different types of logic blocks. It also includes a custom detailed router to perform detail routing on fixed-metal routing fabrics found in VPSAs. The CAD flow provides area, delay, and power estimates for a benchmark circuit when implemented on a particular Structured ASIC architecture.

- Cost, area, delay and power trends for MPSAs, and VPSAs as a function of the number of routing layers.

These contributions are an important step towards the development of improved Structured ASIC architectures and CAD algorithms. To the best of our knowledge, the Structured ASIC die-cost has not been directly used as a performance metric in the past. The proposed cost model will help to compare the different architectural choices in future Structured ASICs by considering their cost. The CAD flow that has been set up as part of this research, can be instrumental in the future research related to Structured ASICs. The flow provides a mechanism to analyze different architectures and different CAD algorithms. In this way, it will facilitate the development of new Structured ASIC architectures and more efficient CAD algorithms. Finally, the performance and cost trends that have been presented in this dissertation provide an insight into various trade-offs involved in Structured ASIC design. These results can also serve as a baseline result for future Structured ASIC studies.

7.3 Limitations

There are several limitations in the research presented in this dissertation. The important limitations are as follows:

- The whitespace insertion scheme used in the CAD flow distributes the whitespace uniformly across the whole die rather than inserting it only at congested locations. Thus, a large amount of whitespace may be added before a design can be routed. This can result in increased delay and power, large area, and expensive dies, especially with small block sizes and fewer routing layers.

- Buffer insertion could be necessary for improving the delay of long or high-fanout nets. It could be particularly important for VPSAs, which have a larger number of vias and higher wirelength compared to MPSAs. We did not consider the impact of buffer insertion.

- We did not perform detailed routing for MPSAs, and our detailed router for VPSAs was unable to route very large circuits in a reasonable amount of time.
- When calculating the delay and power estimates, we did not consider the delay and power dissipation of the logic blocks or precise critical paths.
- We assume that there are dedicated power and clock networks for the logic blocks and we do not consider their overhead.
- There are different possibilities for configuring the logic blocks such as lower-level vias or SRAM cells. The use of additional configuration layers can increase the die cost of Structured ASICs. However, in this dissertation we did not study this effect.
- In CBIC cost calculation, we assumed that all the masks are modified in a re-spin and did not consider the impact of ECO techniques that may reduce the number of masks that need to be changed for each spin ([59]).
- We did not perform technology mapping, but instead approximated it by clustering. As a result, we could not compare different logic block types with each other.

However, despite these limitations we believe our results are sufficiently accurate to draw important conclusions.

7.4 Future Work

Many interesting avenues for future research have been identified during this research. In the following, we highlight some of the interesting directions for future work.

One of the most important directions for future work is to develop a whitespace insertion algorithm that inserts whitespace only in congested areas. As we have shown, there is a significant performance gap that can be filled with such an algorithm. There are several possible approaches. A routability-driven placement contest has been announced as part of International Symposium on Physical Design (ISPD) 2011 conference [9]. The concepts from the open-source placers that will be released as part of the contest could be used in conjunction with the legalization techniques proposed in the RegPlace placer ([43]). This could result in a high-quality congestion-driven Structured ASIC placer. Another possible approach is to use a flow similar to Un/DoPack [53].

Exploring buffer insertion techniques for Structured ASICs is another important future direction. The buffers in Structured ASICs have to be pre-designed and pre-allocated. This makes the buffer insertion problem very challenging. In terms of architecture, the number of buffer resources and how frequently they are allocated, are interesting research questions. The drive strength of buffers is also an interesting problem; the buffers can have a fixed drive strength or their drive strength can be made via-configurable. The impact of these different architectural choices on the performance and cost of Structured ASICs should be explored. Similarly, in terms of CAD, performing the actual buffer insertion is an important problem. The buffers can be inserted into long/high-fanout nets during the routing step, or the CAD can handle this task as a post-routing step.

Detailed routing for VPSAs should be explored. The VPSA routing fabric is very flexible and there are many possibilities for making connections. We found that the increased flexibility becomes a limitation for a PathFinder-based routing algorithm when it is used to route circuits containing hundreds of thousands of nets. To tackle this problem, the use of ASIC-like track assignment techniques such as [33], should be explored. The track assignment is an intermediate step between global and detailed routing, where parts of nets are assigned to various tracks in the ASIC routing fabric. This can be useful in narrowing down the router search space. Alternatively, more restrictive routing architectures may be able to restrict the detail router search space without overly compromising routability.

The CAD flow should be enhanced to include timing-driven and low-power modes. Techniques used in FPGA CAD could be explored for this purpose. However, to identify the timing- or power-critical paths, detailed logic block models are needed (e.g., flip-flop locations). Some of the real Structured ASIC architectures that have been proposed in the literature could be used as a starting point for this purpose.

New architectures for logic blocks and routing fabrics should be investigated. Logic blocks with different numbers of inputs, outputs, and configuration points should be studied. Also, different configuration methods (i.e., through SRAM cells, lower-level vias, or upper-level vias) should be explored. Similarly, the architecture of the metal segments in VPSAs has a significant impact on its performance and cost. We have shown the trends for a few types of fabrics. However, other types of fabrics should be explored that improve the performance or cost. One possibility is to employ SingleVia-style fabrics in which the position of the custom-via layer is changed to the upper-via layers (e.g., via layer between M5 and M6). Another possibility is to explore FPGA-like interconnect fabrics that replace the programmable switches with visa. The performance improvement that can be achieved with such fabrics through CAD enhancement (e.g., intelligently selecting different types of segments during routing) should also be investigated.

Finally, in the long term, techniques to reduce the design and manufacturing complexities of Structured ASICs should be explored. The Structured ASIC design process should be as simple as the FPGA design process, where a user does not need to perform complex physical design tasks such as clock tree synthesis, scan insertion, design-rule checks etc. The devices should be reliable enough to avoid signal integrity simulations. Similarly, design of Structured ASICs using fabrication-friendly layout shapes should be investigated. The research related to Restricted Design Rules (RDRs) (such as [42]) could be useful in this regard.

References

[1] Altera HardCopy ASIC Series http://www.altera.com/products/devices/hardcopy-asics/about/hrd-index.html. → pages 29

[2] Chip-X CX6200 Structured ASIC Datasheet http://www.chipx.com/images/stories/pdf/cx6200_usbphy_ds_0210d.pdf. → pages 29

[3] eASIC Placement Contest. http://web.archive.org/web/20080723214221/http://www.easic.com/index.php?p=university. → pages 80, 98

[4] Nextreme Structured ASIC, eASIC Corp. http://www.easic.com. → pages 29

[5] Faraday Structured ASIC Technology http://www.faraday-tech.com/html/products/structuredASIC.html. → pages 29

[6] The Advent of Next Generation Lithography Technologies in Advanced Semiconductor Processing, *Frost & Sullivan Press Release*, Aug. 27, 2007. → pages 47

[7] Fujitsu AccelArray http://web.archive.org/web/20071031122818/www.fujitsu.com/global/services/microelectronics/product/asic/accelarray/index_2.html. → pages 29

[8] Actel Corporation, IGLOO FPGA Fabric User's Guide, Jul. 2010. http://www.actel.com/documents/IGLOO_UG.pdf. → pages 17

[9] Routability-driven Placement Contest, International Symposium on Physical Design, 2011.
http://www.ispd.cc. → pages 35, 135

[10] *The International Technology Roadmap for Semiconductors (ITRS)*, chapter Yield Enhancement, pages 7–10. 2007. → pages 47

[11] Lightspeed Logic
http://web.archive.org/web/20080111112237/http://www.lightspeed.com/products.html. → pages 29

[12] LSI Logic Rapidchip Xtreme2
http://web.archive.org/web/20061015051026/www.lsilogic.com/files/docs/techdocs/Rapidchip/rcxtreme2_ds.pdf. → pages 29

[13] Private Conversation, Dr. John Logan, Nortel Networks, September 2007. → pages 18

[14] XPressArray-II, ON Semiconductor
http://www.onsemi.com/pub_link/Collateral/TND338-D.PDF. → pages 29

[15] Actel Corporation, ProASIC3 FPGA Fabric User's Guide, Jul. 2010.
http://www.actel.com/documents/PA3_UG.pdf. → pages 17

[16] Altera Corporation, Stratix III Device Handbook, ver 2.1, Jul. 2010.
http://www.altera.com/literature/hb/stx3/stratix3_handbook.pdf, . → pages 14

[17] Altera Corporation, Stratix IV Device Handbook, ver 4.1, Mar. 2010.
http://www.altera.com/literature/hb/stratix-iv/stx4_5v1.pdf, . → pages

[18] Altera Corporation, Stratix V Device Handbook, ver 1.0, Jul. 2010.
http://www.altera.com/literature/hb/stratix-v/stx5_5v1.pdf, . → pages 14

[19] Dylan McGrath, "FPGA startup: Process tech eases ASIC migration", EETimes, March 2010.
http://www.eetimes.com/electronics-news/4088048/FPGA-startup-Process-tech-eases-ASIC-migration. → pages 29

[20] TSMC results fall in Q1, sees rebound, Apr. 2009.
http://www.eetimes.com/showArticle.jhtml?articleID=217200925. → pages 3

[21] ViASIC ViaMask and DuoMask
http://www.viasic.com/products. → pages 29

[22] ASAP Metal Programmable Cell Libraries, Virage Logic
http://www.viragelogic.com/upload/documents/
product_broch_asap_logic_v10.pdf. → pages 29

[23] Xilinx Inc., Virtex-4 Family Overview, ver 3.1, Aug. 2010.
http://www.xilinx.com/support/documentation/data_sheets/ds112.pdf, . →
pages 14

[24] Xilinx Inc., Virtex-5 Family Overview, ver 5.0, Jun. 2009.
http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf, . →
pages

[25] Xilinx Inc., Virtex-6 Family Overview, ver 2.2, Jan. 2010.
http://www.xilinx.com/support/documentation/data_sheets/ds150.pdf, . →
pages 14

[26] S. N. Adya and I. L. Markov. Combinatorial techniques for mixed-size placement. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 10(1):58–90, 2005. → pages 32

[27] S. N. Adya, I. L. Markov, and P. G. Villarrubia. On whitespace and stability in physical synthesis. *Integration, the VLSI Journal*, 39(4): 340–362, 2006. → pages 35, 36

[28] U. Ahmed, G. Lemieux, and S. Wilton. Area, delay, power, and cost trends for Metal-Programmable Structured ASICs (MPSAs). In *IEEE International Conference on Field Programmable Technology (ICFPT)*, pages 278–284, Dec. 2009. → pages iv

[29] U. Ahmed, G. Lemieux, and S. Wilton. The impact of interconnect architecture on Via-Programmed Structured ASICs (VPSAs). In *Proceedings of ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, pages 263–272, Feb. 2010. → pages iv

[30] U. Ahmed, G. Lemieux, and S. Wilton. Performance and cost tradeoffs in Metal-Programmable Structured ASICs (MPSAs). *IEEE Transactions on VLSI Systems*, Oct. 2010. doi:10.1109/TVLSI.2010.2076841. → pages iv

[31] N. Akihiro, M. Kawaharazaki, M. Yoshikawa, and T. Fujino. Architecture of via programmable logic using exclusive-or array (VPEX) for EB direct writing. In *Proceedings of IEEE Custom Integrated Circuits Conference*, pages 261–264, Sep. 2007. → pages xiii, 26, 27

[32] T. Barnett, J. Bickford, and A. Weger. Product yield prediction system and critical area database. *IEEE Transactions on Semiconductor Manufacturing Transactions on*, 21(3):337 –341, Aug. 2008. → pages 31

[33] S. Batterywala, N. Shenoy, W. Nicholls, and H. Zhou. Track assignment: A desirable intermediate step between global routing and detailed routing. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 59–66, Nov. 2002. → pages 34, 66, 136

[34] V. Betz and J. Rose. Vpr and t-vpack: Versatile packing, placement and routing for FPGAs. In *International Workshop on Field Programmable Logic and Applications (FPL)*, pages 213–222, 1997. → pages 32, 61

[35] V. Betz, J. Rose, and A. Marquardt. *Architecture and CAD for Deep-Submicron FPGAs*. Kluwer Academic Publishers, 1999. → pages 4, 32, 61, 67, 71, 72, 82

[36] M. A. Beunder and B. Hoefflinger. New directions in semicustom arrays. *IEEE Journal of Solid-State Circuits*, 23(3):728–735, Jun. 1988. → pages 14

[37] D. Blaauw and K. Gala. Deep submicron issues in high performance design. In *Proceedings of Int'l Workshop on Power and Timing Modeling, Optimization and Simulation*, Sep. 2001. → pages 2

[38] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic. *Field Programmable Gate Arrays*. Springer, 1992. → pages 4

[39] A. Caldwell, A. Kahng, and I. Markov. Optimal partitioners and end-case placers for standard-cell layout. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(11):1304–1313, Nov. 2000. → pages 32

[40] A. E. Caldwell, A. B. Kahng, and I. L. Markov. Design and implementation of move-based heuristics for VLSI hypergraph partitioning. *J. Exp. Algorithmics*, 5, 2000. → pages 32

[41] A. E. Caldwell, A. B. Kahng, and I. L. Markov. Improved algorithms for hypergraph bipartitioning. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 661–666, 2000. → pages 32

[42] L. Capodieci, P. Gupta, A. B. Kahng, D. Sylvester, and J. Yang. Toward a methodology for manufacturability-driven design rule exploration. In *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, pages 311–316, 2004. → pages 137

[43] A. Chakraborty, A. Kumar, and D. Pan. Regplace: A high quality open-source placement framework for structured ASICs. In *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, pages 442–447, Jul. 2009. → pages 66, 135

[44] T. F. Chan, J. Cong, J. R. Shinnerl, K. Sze, and M. Xie. mpl6: enhanced multilevel mixed-size placement. In *Proceedings of the International Symposium on Physical Design (ISPD)*, pages 212–214, 2006. → pages 32

[45] C.-C. Chang and J. Cong. Pseudo pin assignment with crosstalk noise control. In *Proceedings of the International Symposium on Physical Design*, pages 41–47, 2000. → pages 34

[46] T. Chau, P. Leong, S. Ho, B. Chan, S. Yuen, K. Pun, O. Choy, and X. Wang. A comparison of via-programmable gate array logic cell circuits. In *Proceedings of ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, pages 53–62, 2009. → pages 26

[47] R. Chawala. FPGAs and structured ASICs: New solutions for changing market dynamics. *Chip Design Magazine*, Oct. 2005. → pages 18

[48] H. H. Chen and D. D. Ling. Power supply noise analysis methodology for deep-submicron VLSI chip design. In *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, pages 638–643, 1997. → pages 2

[49] J. F. Chen, T. Laidig, K. E. Wampler, and R. Caldwell. Practical method for full-chip optical proximity correction. In *Proc. SPIE*, pages 790–803, 1997. → pages 2

[50] T.-C. Chen, Z.-W. Jiang, T.-C. Hsu, H.-C. Chen, and Y.-W. Chang. A high-quality mixed-size analytical placer considering preplaced blocks and density constraints. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 187–192, 2006. → pages 32, 61

[51] S. Y. L. Chin and S. J. E. Wilton. Static and dynamic memory footprint reduction for FPGA routing algorithms. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, 1:18:1–18:20, January 2009. → pages 68

[52] D. Chinnery and K. Keutzer. *Closing the Gap Between ASIC & Custom: Tools and Techniques for High-Performance ASIC Design*. Springer, 1st edition, 2002. → pages 3

[53] D. Chiu, G. Lemieux, and S. Wilton. Congestion-driven regional re-clustering for low-cost FPGAs. In *IC-FPT*, pages 167–174, Dec. 2009. → pages 35, 36, 90, 135

[54] M. Cho and D. Pan. Boxrouter: A new global router based on box expansion and progressive ILP. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 26(12):2130–2143, Dec. 2007. → pages 34

[55] G. W. Clow. A global routing algorithm for general cells. In *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, pages 45–51, 1984. → pages 33

[56] J. Cong, J. R. Shinnerl, M. Xie, T. Kong, and X. Yuan. Large-scale circuit placement. *ACM Transactions on Design Automation of Electronic Systems (TODAES)*, 10(2):389–430, 2005. ISSN 1084-4309. → pages 32

[57] M. E. de Lima and D. J. Kinniment. Sea-of-gates architecture. *Microelectronics Journal*, 26(5):431–440, Jul. 1995. → pages 14

[58] J.-R. Gao, P.-C. Wu, and T.-C. Wang. A new global router for modern designs. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 232–237, Mar. 2008. → pages 34

[59] S. Golson. The human ECO compiler. In *Synopsys Users Group Conference (SNUG)*, 2004. → pages 54, 134

[60] J. N. Helbert. *Handbook of VLSI Microlithography*. William Andrew, 2nd edition, 2001. → pages 1

[61] J. L. Hennessy and D. A. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Inc., 2006. → pages 44

[62] A. Hetzel. A sequential detailed router for huge grid graphs. In *Proceedings of the Conference on Design Automation and Test in Europe (DATE)*, pages 332–339, 1998. → pages 34

[63] P. Heydari and M. Pedram. Capacitive coupling noise in high-speed VLSI circuits. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(3):478–488, Mar. 2005. → pages 2

[64] D. A. Hodges, H. G. Jackson, and R. A. Saleh. *Analysis and Design of Digital Integrated Circuits*. McGraw Hill, 3rd edition, 2004. → pages 2

[65] Y. I. Ismail and E. G. Friedman. *On-Chip Inductance in High Speed Integrated Circuits*. Springer, 1st edition, 2001. → pages 2

[66] R. C. Jaeger. *Introduction to Microelectronic Fabrication*. Prentice Hall, 2nd edition, 2001. → pages 1

[67] A. B. Kahng and Y. C. Pati. Subwavelength lithography and its potential impact on design and EDA. In *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, pages 799–804, 1999. → pages 2

[68] S. Kaptanoglu. Power and a new class of future FPGA architectures. In *Keynote Address at Int'l Conf. on Field Programmable Technologies (IC-FPT)*, Dec. 2007. → pages 17

[69] V. Kheterpal, A. J. Strojwas, and L. Pileggi. Routing architecture exploration for regular fabrics. In *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, pages 204–207, 2004. → pages 25

[70] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983. → pages 32

[71] A. Koorapaty, V. Kheterpal, P. Gopalakrishnan, M. Fu, and L. Pileggi. Exploring logic block granularity for regular fabrics. In *Proceedings of the Conference on Design Automation and Test in Europe (DATE)*, 2004. → pages 25

[72] I. Koren and A. D. Singh. Fault tolerance in VLSI circuits. *IEEE Computer*, 23:73–83, Jul. 1990. → pages 30, 31

[73] I. Kuon and J. Rose. Measuring the gap between FPGAs and ASICs. In *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA)*, pages 21–30, 2006. → pages 13, 17

[74] I. Kuon and J. Rose. Measuring the gap between FPGAs and ASICs. *IEEE Transactions on CAD*, 26(2):203–215, 2007. → pages 4, 13, 17

[75] M. Levenson, N. Viswanathan, and R. Simpson. Improving resolution in photolithography with a phase-shifting mask. *Electron Devices, IEEE Transactions on*, 29(12):1828–1836, Dec. 1982. → pages 2

[76] C. Li, M. Xie, C.-K. Koh, J. Cong, and P. H. Madden. Routability-driven placement and white space allocation. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 26(5):858–871, 2007. → pages 35, 36

[77] L. W. Liebmann. Layout impact of resolution enhancement techniques: impediment or opportunity? In *Proceedings of the International Symposium on Physical Design (ISPD)*, pages 110–117, 2003. → pages 16

[78] L. W. Liebmann, T. H. Newman, R. A. Ferguson, R. M. Martino, A. F. Molless, M. O. Neisser, and J. T. Weed. Comprehensive evaluation of major phase-shift mask technologies for isolated gate structures in logic designs. volume 2197, pages 612–623. SPIE, 1994. → pages 2

[79] L. W. Liebmann, A. F. Molless, R. A. Ferguson, A. K. K. Wong, and S. M. Mansfield. Understanding across-chip line-width variation: the first step toward optical proximity correction. volume 3051, pages 124–136. SPIE, 1997. → pages 2

[80] Y. Liu, A. Zakhor, and M. Zuniga. Computer-aided phase shift mask design with reduced complexity. *Semiconductor Manufacturing, IEEE Transactions on*, 9(2):170–181, May 1996. → pages 2

[81] T. Luo and D. Z. Pan. Dplace2.0: a stable and efficient analytical placement based on diffusion. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 346–351, 2008. → pages 32

[82] G. S. May and C. J. Spanos. *Fundamentals of Semiconductor Manufacturing and Process Control*. Wiley-IEEE Press, 1st edition, 2006. → pages 1

[83] L. McMurchie and C. Ebeling. PathFinder: a negotiation-based performance-driven router for FPGAs. In *Proceedings of ACM/SIGDA International Symposium on Field-Programmable Gate Arrays (FPGA)*, 1995. → pages 33

[84] A. Misaka, A. Goda, K. Matsuoka, H. Umimoto, and S. Odanaka. Optical proximity correction in dram cell using a new statistical methodology. volume 3051, pages 763–773. SPIE, 1997. → pages 2

[85] F. Mo and R. K. Brayton. *Regular Fabrics in Deep Sub-Micron Integrated-Circuit Design*. Springer, 1st edition, 2004. → pages 17

[86] M. D. Moffitt. Maizerouter: engineering an effective global router. In *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 226–231, 2008. → pages 34

[87] K. Morris. Un-structured ASIC. *FPGA and Structured ASIC Journal*, Feb. 2007. → pages 13, 17

[88] A. Nakamura, M. Kawarasaki, K. Ishibashi, M. Yoshikawa, and T. Fujino. Regular fabric of via programmable logic device using exclusive-or array VPEX for EB direct writing. *IEICE Transactions*, 91-C(4):509–516, 2008. → pages 26

[89] T. Okamoto, T. Kimoto, and N. Maeda. Design methodology and tools for NEC electronics' structured ASIC ISSP. In *ISPD '04*, pages 90–96, 2004. → pages 29

[90] Z. Or-Bach. Paradigm shift in ASIC technology: In-standard metal, out-standard cell. *eASIC White Paper*, September 2005. → pages 3, 15, 47, 87

[91] M. M. Ozdal and M. D. F. Wong. Archer: a history-driven global routing algorithm. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 488–495, 2007. → pages 34

[92] L. Pileggi, H. Schmit, A. J. Strojwas, P. Gopalakrishnan, V. Kheterpal, A. Koorapaty, C. Patel, V. Rovner, and K. Y. Tong. Exploring regular fabrics to optimize the performance-cost trade-off. In *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, pages 782–787, 2003. → pages 24

[93] P. Rai-Choudhury. *Handbook of Microlithography, Micromachining, and Microfabrication. Volume 1: Microlithography*. SPIE Publications, 1997. → pages 1

[94] Y. Ran and M. Marek-Sadowska. The magic of a via-configurable regular fabric. In *Proceedings of IEEE International Conference on Computer Design (ICCD)*, pages 338–343, 2004. → pages xiii, 23, 24

[95] Y. Ran and M. Marek-Sadowska. An integrated design flow for a via-configurable gate array. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 552–589, 2004. → pages

[96] Y. Ran and M. Marek-Sadowska. Via-configurable routing architectures and fast design mappability estimation for regular fabrics. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 25–32, 2005. → pages 100, 102, 103

[97] Y. Ran and M. Marek-Sadowska. Via-configurable routing architectures and fast design mappability estimation for regular fabrics. *IEEE Transactions on VLSI Systems*, 14(9):998–1009, Sept. 2006. → pages 23, 71, 100, 102, 103

[98] J. A. Roy and I. L. Markov. High-performance routing at the nanometer scale. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 496–502, 2007. → pages 34, 66

[99] J. A. Roy, D. A. Papa, S. N. Adya, H. H. Chan, A. N. Ng, J. F. Lu, and I. L. Markov. CAPO: Robust and scalable open-source min-cut floorplacer. In *ISPD*, pages 224–226, 2005. → pages 32, 61

[100] J. A. Roy, D. A. Papa, A. N. Ng, and I. L. Markov. Satisfying whitespace requirements in top-down placement. In *ISPD '06*, pages 206–208, 2006. → pages 35

[101] J. Rubinstein, P. P. Jr., and M. A. Horowitz. Signal delay in RC tree networks. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 2(3): 202–211, 1983. → pages 60

[102] F. M. Schellenberg and L. Capodieci. Impact of RET on physical layouts. In *Proceedings of the International Symposium on Physical Design (ISPD)*, pages 52–55, 2001. → pages 2

[103] F. M. Schellenberg, H. Zhang, and J. Morrow. Evaluation of OPC efficacy. volume 2726, pages 680–688. SPIE, 1996. → pages 2

[104] H. Schmit, A. Gupta, and R. Ciobanu. Placement challenges for Structured ASICs. In *ISPD'08*, pages 84–86, 2008. → pages 63

[105] N. Selvakkumaran, P. N. Parakh, and G. Karypis. Perimeter-degree: A priori metric for directly measuring and homogenizing interconnection complexity in multilevel placement. In *Proceedings of System Level Interconnect Prediction Workshop (SLIP)*, pages 53–59, 2003. → pages 35

[106] A. Sharma, C. Ebeling, and S. Hauck. Architecture-adaptive routability-driven placement for FPGAs. In *Proceedings of IEEE International Conference on Field Programmable Logic and Applications (FPL)*, pages 427–432, 2005. → pages 35, 36

[107] M. J. Smith. *Application-Specific Integrated Circuits*. Addison-Wesley Professional, 1997. → pages 3

[108] C. Snyder and M. Disman. Structured ASICs offer application adaptability. *Semiview Report*, Dec. 2003. http://web.archive.org/web/20060520112713/http://www.semiview.com/downloads/semiview_adaptability.pdf. → pages xi, 13, 15

[109] L. Stok and J. Cohn. There is life left in ASICs. In *ISPD*, pages 48–50, 2003. → pages 3

[110] C. Strapper, F. Armstrong, and K. Saji. Integrated circuit yield statistics. *Proc. of the IEEE*, 71(4):453–470, April 1983. → pages 30

[111] T. Taghavi, X. Yang, and B.-K. Choi. Dragon2005: Large-scale mixed-size placement tool. In *Proceedings of the International Symposium on Physical Design (ISPD)*, pages 245–247, 2005. → pages 32

[112] M. Tom and G. Lemieux. Logic block clustering of large designs for channel-width constrained FPGAs. In *Proceedings of the ACM/IEEE Design Automation Conference (DAC)*, pages 726–731, 2005. → pages 35, 36, 90

[113] M. Tom, D. Leong, and G. Lemieux. Un/DoPack: re-clustering of large system-on-chip designs with interconnect variation for low-cost FPGAs. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 680–687, 2006. → pages 35, 36, 90

[114] H.-P. Tseng, L. Scheffer, and C. Sechen. Timing and crosstalk driven area routing. *IEEE Transactions on Computer Aided Design*, 20:528–381, Apr. 2001. → pages 34

[115] H. Veendrick. *Deep-Submicron CMOS ICs*. Kluwer Academic Publishers, 2nd edition, 2000. → pages 2

[116] F. Veredas, M. Scheppler, and H. Pfleiderer. Automated conversion from a LUT-based FPGA to a LUT-based MPGA with fast turnaround time. In *Proceedings of the Conference on Design Automation and Test in Europe (DATE)*, pages 36–41, 2006. → pages 25

[117] F. Veredas, M. Scheppler, B. Zhai, and H. Pfleiderer. Regular routing architecture for a LUT-based MPGA. In *Proceedings of IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 257–262, 2006. → pages 25

[118] N. Viswanathan, M. Pan, and C. Chu. Fastplace 3.0: A fast multilevel quadratic placement algorithm with placement congestion control. In

Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC), pages 135–140, 2007. → pages 32

[119] A. K. Wong. Some thoughts on the ic design-manufacture interface. *IEEE Design and Test of Computers*, 22:206–213, 2005. → pages 16

[120] A. K.-K. Wong. *Resolution Enhancement Techniques in Optical Lithography*. SPIE Publications, Mar. 2001. → pages 2, 16

[121] K.-C. Wu and Y.-W. Tsai. Structured ASIC, evolution or revolution? In *Proceedings of the International Symposium on Physical Design (ISPD)*, pages 103–106, 2004. → pages 13, 18

[122] B. Zahiri. Structured ASICs: Opportunities and challenges. In *Proceedings of IEEE International Conference on Computer Design (ICCD)*, pages 404–409, Oct. 2003. → pages 4, 13, 17

Appendices

Appendix A

Characteristics of MCNC Benchmarks

In this appendix, we show the characteristics of each MCNC benchmark circuit used in the experiments. The number of primary inputs and primary outputs for each benchmark circuit are shown in Table A.1. Tables A.2 and A.3 list the number of logic blocks and number of nets in each circuit respectively, when mapped to a particular logic block type (logic blocks with a given number of input and output pins).

Table A.1: MCNC benchmarks: primary inputs and primary outputs

Circuit	Primary Inputs	Primary Outputs
alu4	14	8
apex2	38	3
apex4	9	19
bigkey	229	197
clma	62	82
des	256	245
diffeq	64	39
dsip	229	197
elliptic	131	114
ex1010	10	10
ex5p	8	63
frisc	20	116
misex3	14	14
pdc	16	40
s298	4	6
s38417	29	106
seq	41	35
spla	16	46
tseng	52	122

Table A.2: MCNC benchmarks: number of logic blocks

Circuit	Number of Logic Blocks in a Circuit for Different Logic Block Types							
	2-in,1-out	**4-in,2-out**	**6-in,3-out**	**8-in,4-out**	**10-in,5-out**	**12-in,6-out**	**14-in,7-out**	**16-in,8-out**
alu4	2732	1366	911	683	547	456	391	342
apex2	3165	1583	1055	792	633	528	453	396
apex4	2196	1098	732	549	440	366	314	275
bigkey	2979	1490	993	745	596	497	426	373
clma	14253	7127	4751	3564	2851	2376	2037	1782
des	2901	1451	967	726	581	484	415	363
diffeq	2556	1278	852	639	512	426	366	320
dsip	2531	1266	844	633	507	422	362	317
elliptic	5474	2737	1825	1369	1095	913	782	685
ex1010	8020	4010	2674	2005	1604	1337	1146	1003
ex5p	1779	890	593	445	356	297	255	223
frisc	6023	3012	2008	1506	1205	1004	861	753
misex3	2557	1279	853	640	512	427	366	320
pdc	8408	4204	2803	2102	1682	1402	1202	1051
s298	4272	2136	1424	1068	855	712	611	534
s38417	13656	6828	4552	3414	2732	2276	1951	1707
seq	2939	1470	980	735	588	490	420	368
spla	7438	3719	2480	1860	1488	1240	1063	930
tseng	1861	931	621	466	373	311	266	233

Table A.3: MCNC benchmarks: number of nets

Circuit	Number of Nets in a Circuit for Different Logic Block Types							
	2-in,1-out	**4-in,2-out**	**6-in,3-out**	**8-in,4-out**	**10-in,5-out**	**12-in,6-out**	**14-in,7-out**	**16-in,8-out**
alu4	2746	1994	1672	1480	1381	1301	1230	1173
apex2	3203	2367	2057	1871	1756	1666	1647	1587
apex4	2205	1707	1495	1352	1301	1248	1232	1191
bigkey	3208	2074	1844	1753	1738	1521	1422	1240
clma	14315	10116	8766	8012	7558	7190	6934	6672
des	3157	2384	2043	1779	1642	1621	1628	1517
diffeq	2620	2043	1647	1574	1471	1348	1304	1272
dsip	2760	2073	1470	1083	955	1170	1031	797
elliptic	5605	4391	3159	2779	2845	2622	2486	2401
ex1010	8030	6093	5157	4853	4643	4427	4333	4234
ex5p	1787	1423	1279	1193	1135	1098	1078	1067
frisc	6043	4581	3304	2989	2874	2727	2645	2566
misex3	2571	1868	1625	1496	1417	1344	1280	1251
pdc	8424	6127	5150	4735	4417	4217	4070	3894
s298	4276	2829	2294	1968	1696	1545	1452	1388
s38417	13685	9730	8869	7957	7503	7309	6991	6618
seq	2980	2211	1914	1743	1640	1530	1489	1477
spla	7454	5322	4547	4109	3837	3639	3491	3402
tseng	1913	1455	1143	1083	1004	950	941	863

Appendix B

Logic Fabrics and Pin Positions

B.1 Crossover Fabric

The trends for the Crossover fabric with different pins positions are shown in Figures B.1–B.5. The plots show trends for the most congested circuit (*pdc*) and with logic blocks having 4, 6, 8, 12, and 14 inputs.

B.2 Jumper20 Fabric

The trends for the Jumper20 fabric with different pins positions are shown in Figures B.6–B.10. The plots show trends for the most congested circuit (*pdc*) and with logic blocks having 4, 6, 8, 12, and 14 inputs.

B.3 SingleVia Fabric

The trends for the SingleVia fabric with different pins positions are shown in Figures B.11–B.15. The plots show trends for the most congested circuit (*pdc*) and with logic blocks having 4, 6, 8, 12, and 14 inputs.

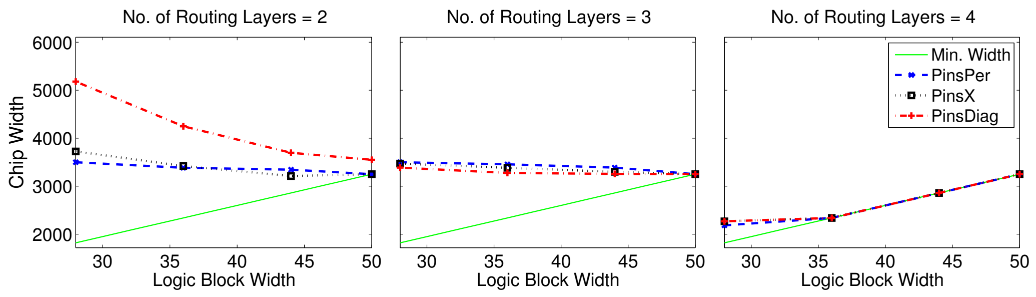


Figure B.1: Trends for 4-input, 2-output logic blocks with *Crossover* fabric

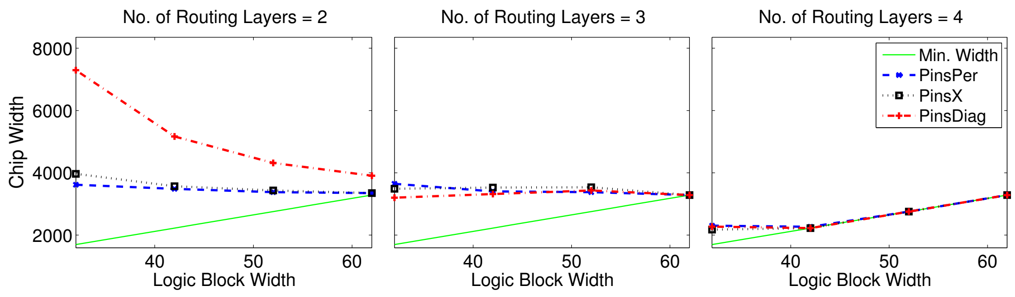


Figure B.2: Trends for 6-input, 3-output logic blocks with *Crossover* fabric

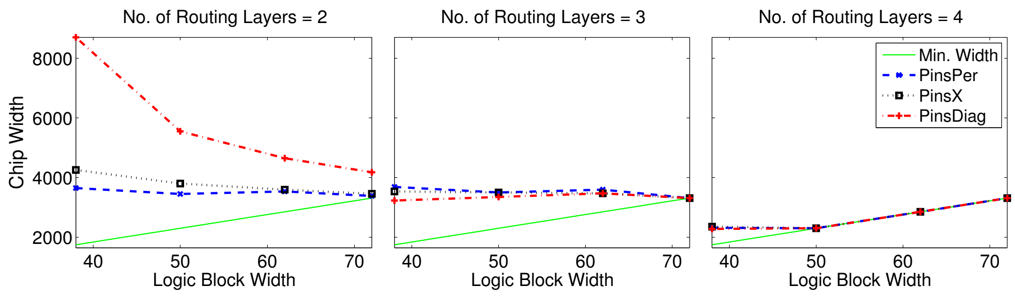


Figure B.3: Trends for 8-input, 4-output logic blocks with *Crossover* fabric

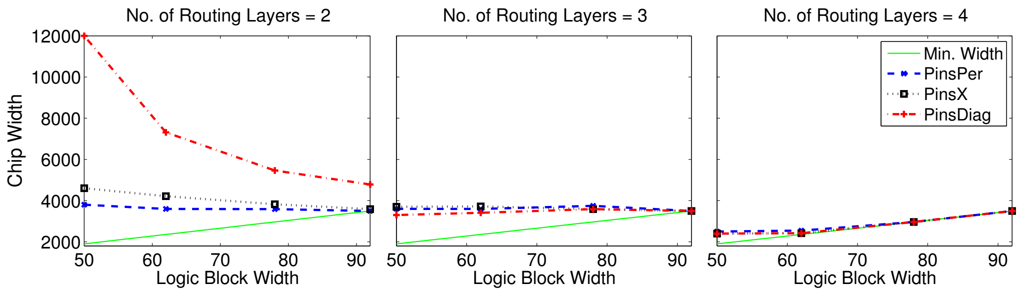


Figure B.4: Trends for 12-input, 6-output logic blocks with *Crossover* fabric

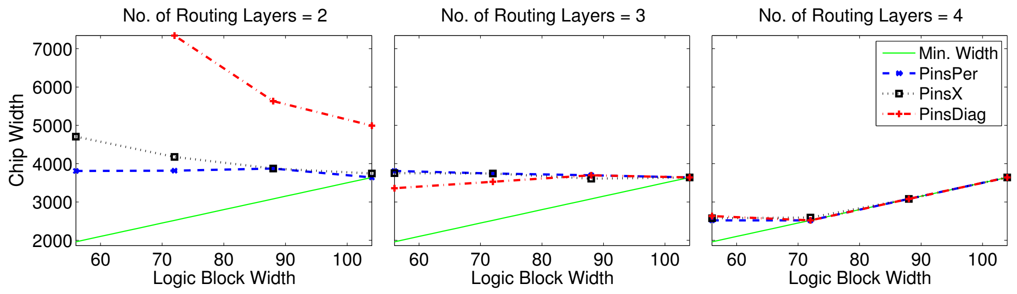


Figure B.5: Trends for 14-input, 7-output logic blocks with *Crossover* fabric

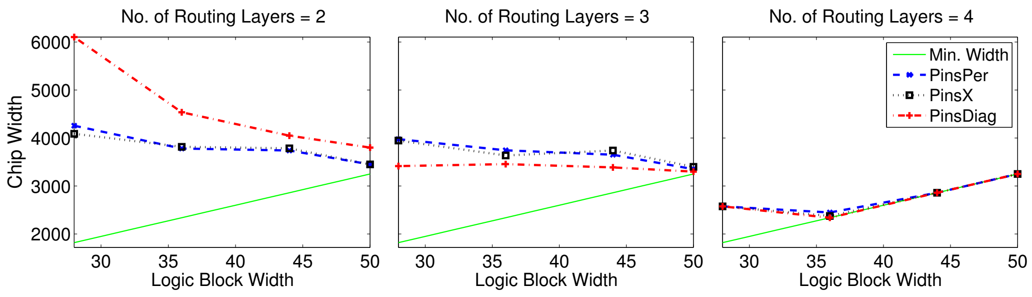


Figure B.6: Trends for 4-input, 2-output logic blocks with *Jumper20* fabric

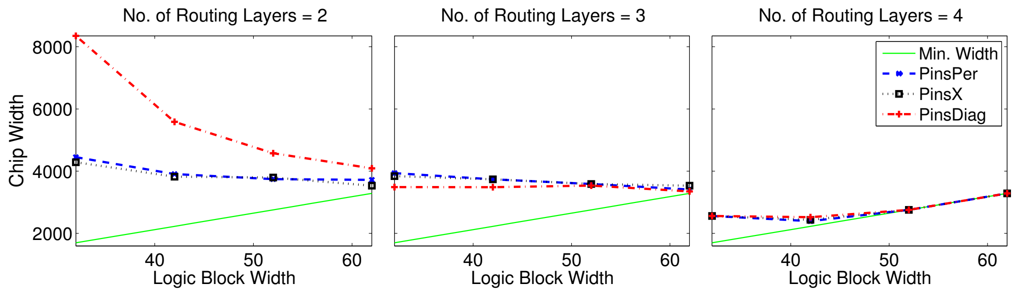


Figure B.7: Trends for 6-input, 3-output logic blocks with *Jumper20* fabric

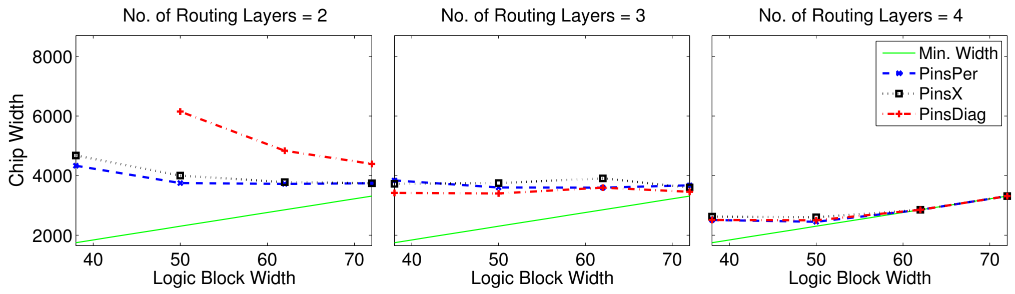


Figure B.8: Trends for 8-input, 4-output logic blocks with *Jumper20* fabric

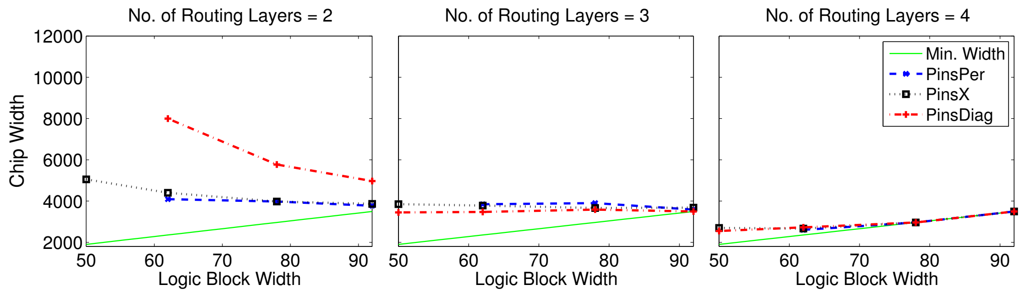


Figure B.9: Trends for 12-input, 6-output logic blocks with *Jumper20* fabric

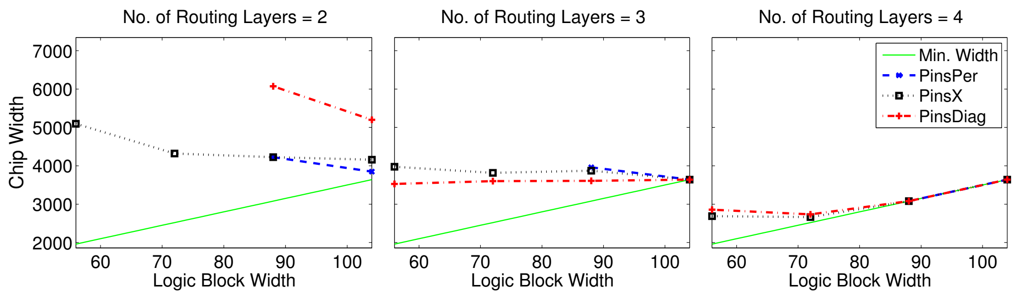


Figure B.10: Trends for 14-input, 7-output logic blocks with *Jumper20* fabric

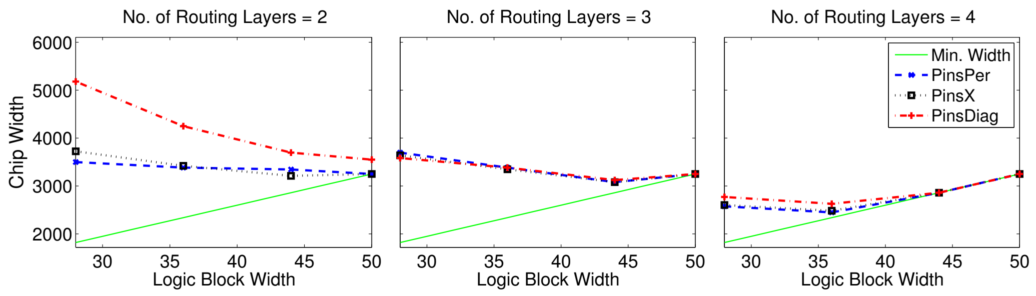


Figure B.11: Trends for 4-input, 2-output logic blocks with *SingleVia* fabric

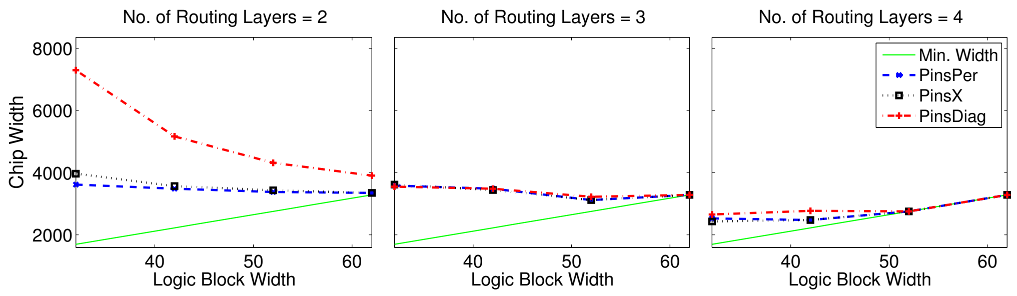


Figure B.12: Trends for 6-input, 3-output logic blocks with *SingleVia* fabric

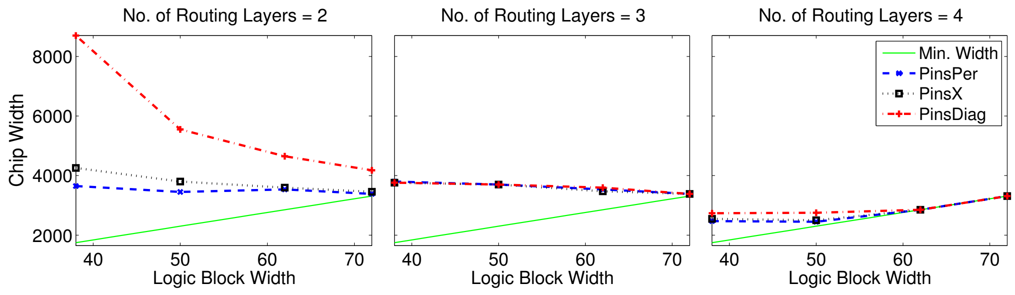


Figure B.13: Trends for 8-input, 4-output logic blocks with *SingleVia* fabric

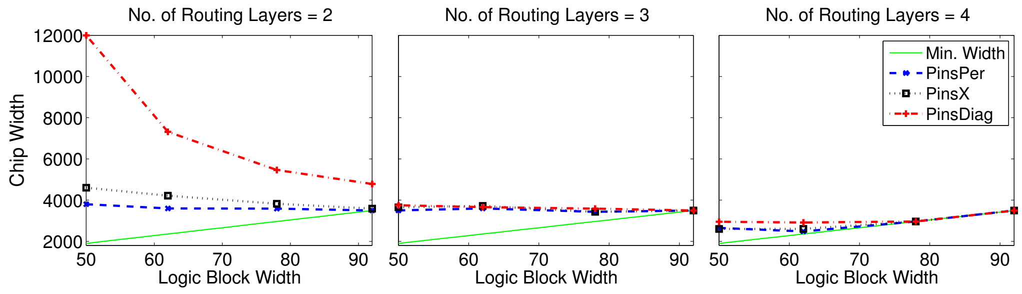


Figure B.14: Trends for 12-input, 6-output logic blocks with *SingleVia* fabric

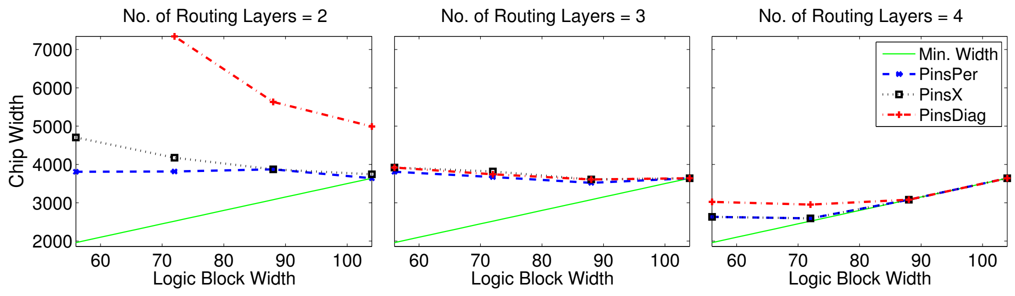


Figure B.15: Trends for 14-input, 7-output logic blocks with *SingleVia* fabric

Appendix C

Logic Block Dimensions

Minimum and maximum logic block sizes calculated for each MCNC circuit for different IO pin count are shown in Tables C.1 and C.2 respectively. The minimum block size is the smallest block size that is routable with 2 routing layers. The maximum block size is the smalled block size that can be routed without any whitespace. The block sizes were determined using *Crossover fabric with* PinsPer pin position scheme. The largest value for each logic block type (IO pins) was used to determine the logic block dimensions.

Table C.1: Minimum block sizes

Circuit	Block Sizes for Logic Blocks with different I/O pin count							
	2-in,1-out	**4-in,2-out**	**6-in,3-out**	**8-in,4-out**	**10-in,5-out**	**12-in,6-out**	**14-in,7-out**	**16-in,8-out**
alu4	10 × 10	16 × 16	20 × 20	22 × 22	24 × 24	28 × 28	30 × 30	30 × 30
apex2	10 × 10	16 × 16	20 × 20	22 × 22	24 × 24	30 × 30	32 × 32	34 × 34
apex4	12 × 12	16 × 16	20 × 20	22 × 22	26 × 26	28 × 28	34 × 34	34 × 34
bigkey	10 × 10	16 × 16	18 × 18	20 × 20	22 × 22	24 × 24	28 × 28	28 × 28
clma	10 × 10	16 × 16	22 × 22	24 × 24	30 × 30	32 × 32	36 × 36	36 × 36
des	12 × 12	16 × 16	20 × 20	20 × 20	24 × 24	28 × 28	30 × 30	30 × 30
diffeq	10 × 10	14 × 14	16 × 16	18 × 18	22 × 22	24 × 24	30 × 30	28 × 28
dsip	10 × 10	16 × 16	18 × 18	20 × 20	22 × 22	26 × 26	28 × 28	28 × 28
elliptic	10 × 10	16 × 16	20 × 20	22 × 22	26 × 26	28 × 28	34 × 34	34 × 34
ex1010	10 × 10	16 × 16	20 × 20	22 × 22	28 × 28	30 × 30	36 × 36	38 × 38
ex5p	12 × 12	18 × 18	22 × 22	24 × 24	28 × 28	32 × 32	36 × 36	38 × 38
frisc	10 × 10	16 × 16	20 × 20	22 × 22	26 × 26	30 × 30	34 × 34	36 × 36
misex3	10 × 10	16 × 16	18 × 18	22 × 22	24 × 24	28 × 28	32 × 32	32 × 32
pdc	12 × 12	20 × 20	22 × 22	26 × 26	30 × 30	32 × 32	38 × 38	40 × 40
s298	8 × 8	14 × 14	16 × 16	16 × 16	20 × 20	24 × 24	26 × 26	26 × 26
s38417	10 × 10	16 × 16	20 × 20	22 × 22	24 × 24	30 × 30	34 × 34	34 × 34
seq	10 × 10	16 × 16	20 × 20	22 × 22	26 × 26	28 × 28	32 × 32	34 × 34
spla	12 × 12	18 × 18	22 × 22	24 × 24	28 × 28	32 × 32	36 × 36	38 × 38
tseng	10 × 10	14 × 14	18 × 18	18 × 18	22 × 22	24 × 24	30 × 30	28 × 28
MAX	12 × 12	20 × 20	22 × 22	26 × 26	30 × 30	32 × 32	38 × 38	40 × 40

Table C.2: Maximum block sizes

Circuit	Block Sizes for Logic Blocks with different I/O pin count							
	2-in,1-out	**4-in,2-out**	**6-in,3-out**	**8-in,4-out**	**10-in,5-out**	**12-in,6-out**	**14-in,7-out**	**16-in,8-out**
alu4	24 × 24	40 × 40	46 × 46	54 × 54	60 × 60	70 × 70	76 × 76	80 × 80
apex2	24 × 24	38 × 38	50 × 50	56 × 56	66 × 66	76 × 76	82 × 82	88 × 88
apex4	28 × 28	42 × 42	52 × 52	60 × 60	74 × 74	76 × 76	86 × 86	94 × 94
bigkey	22 × 22	34 × 34	42 × 42	48 × 48	48 × 48	52 × 52	62 × 62	62 × 62
clma	24 × 24	40 × 40	56 × 56	68 × 68	76 × 76	84 × 84	94 × 94	100 × 100
des	26 × 26	38 × 38	46 × 46	54 × 54	58 × 58	68 × 68	74 × 74	76 × 76
diffeq	20 × 20	30 × 30	38 × 38	42 × 42	50 × 50	56 × 56	60 × 60	66 × 66
dsip	22 × 22	34 × 34	42 × 42	40 × 40	52 × 52	62 × 62	64 × 64	60 × 60
elliptic	24 × 24	40 × 40	46 × 46	54 × 54	64 × 64	70 × 70	86 × 86	88 × 88
ex1010	26 × 26	40 × 40	52 × 52	64 × 64	70 × 70	80 × 80	88 × 88	94 × 94
ex5p	28 × 28	44 × 44	56 × 56	64 × 64	74 × 74	80 × 80	90 × 90	100 × 100
frisc	24 × 24	40 × 40	50 × 50	60 × 60	68 × 68	78 × 78	84 × 84	92 × 92
misex3	24 × 24	40 × 40	48 × 48	56 × 56	66 × 66	70 × 70	76 × 76	86 × 86
pdc	28 × 28	50 × 50	62 × 62	72 × 72	82 × 82	92 × 92	104 × 104	106 × 106
s298	16 × 16	26 × 26	32 × 32	36 × 36	40 × 40	44 × 44	50 × 50	52 × 52
s38417	20 × 20	32 × 32	46 × 46	54 × 54	62 × 62	72 × 72	76 × 76	82 × 82
seq	26 × 26	40 × 40	52 × 52	60 × 60	68 × 68	78 × 78	82 × 82	92 × 92
spla	26 × 26	46 × 46	60 × 60	68 × 68	80 × 80	88 × 88	98 × 98	104 × 104
tseng	20 × 20	30 × 30	36 × 36	42 × 42	48 × 48	54 × 54	58 × 58	60 × 60
MAX	28 × 28	50 × 50	62 × 62	72 × 72	82 × 82	92 × 92	104 × 104	106 × 106

Appendix D

VPSA Performance and Cost Trends

In this appendix, we show the power, delay, area, and die-cost for logic blocks with 4, 6, 8, 12, and 14 inputs.

Figures D.1–D.5 show the trends for power. The delay trends are shown in Figures D.6–D.10. The area trends are shown in Figures D.11–D.15. Finally, Figures D.16–D.20 show the die-cost trends.

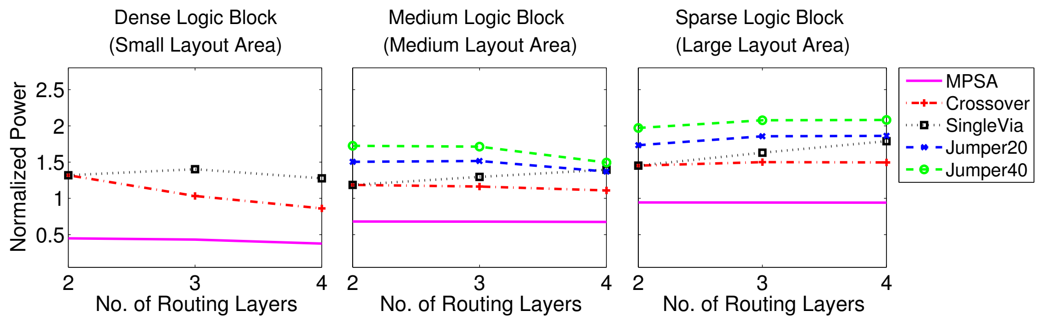


Figure D.1: Power trends for 4-input, 2-output logic blocks

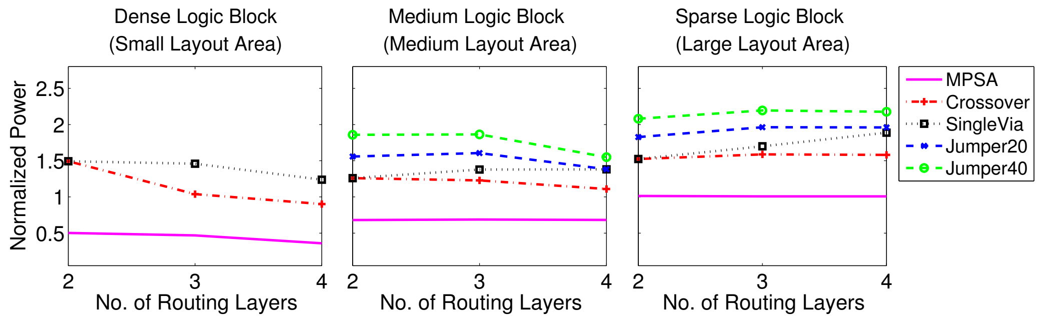


Figure D.2: Power trends for 6-input, 3-output logic blocks

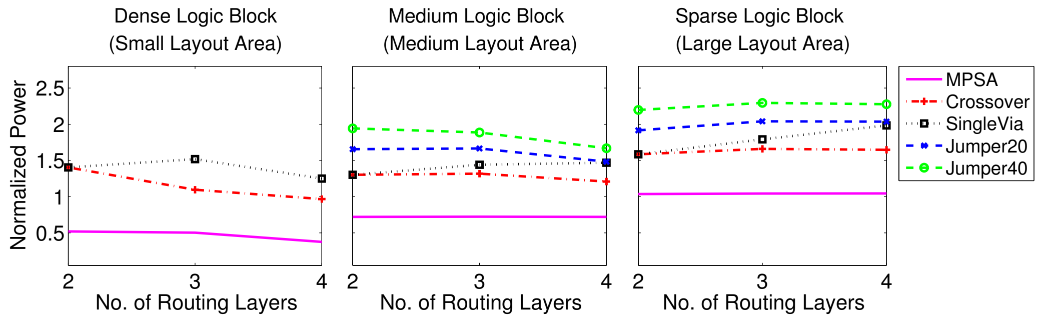


Figure D.3: Power trends for 8-input, 4-output logic blocks

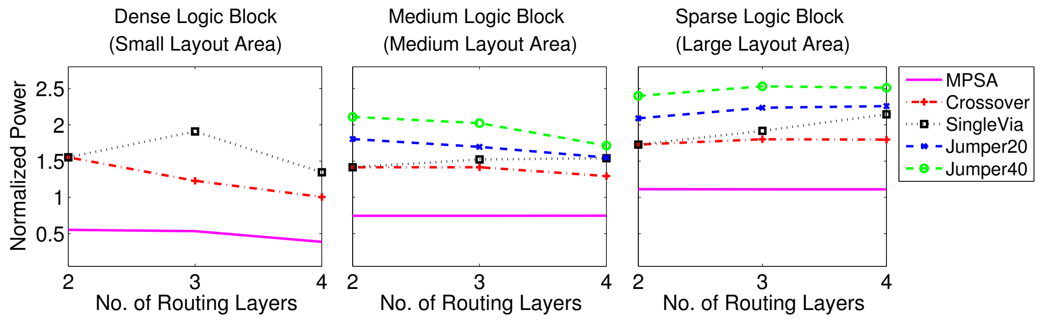


Figure D.4: Power trends for 12-input, 6-output logic blocks

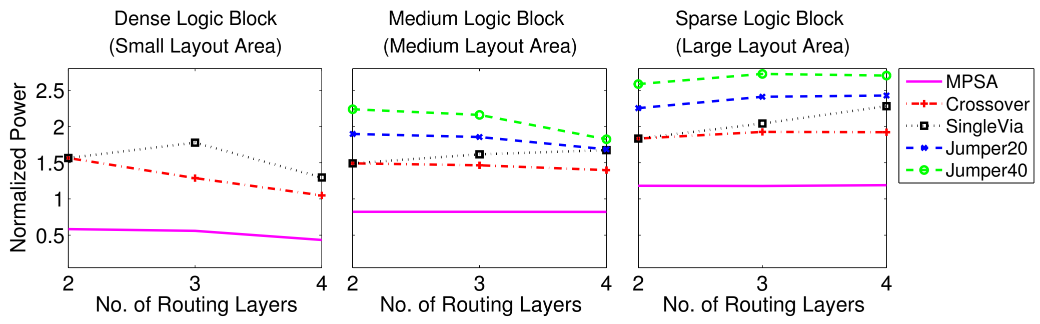


Figure D.5: Power trends for 14-input, 7-output logic blocks

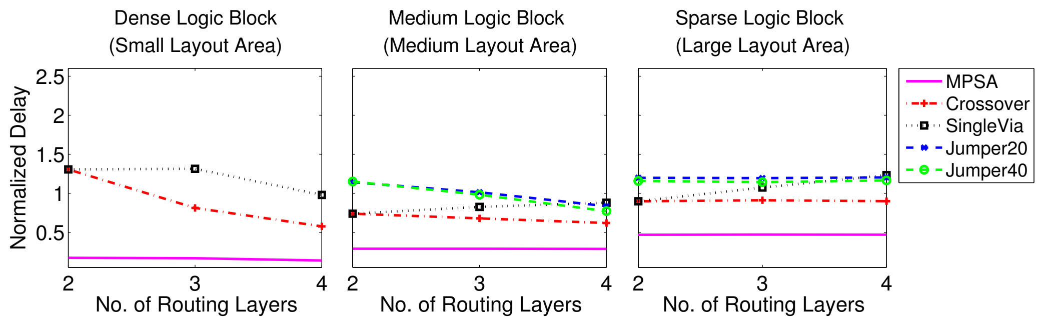


Figure D.6: Delay trends for 4-input, 2-output logic blocks

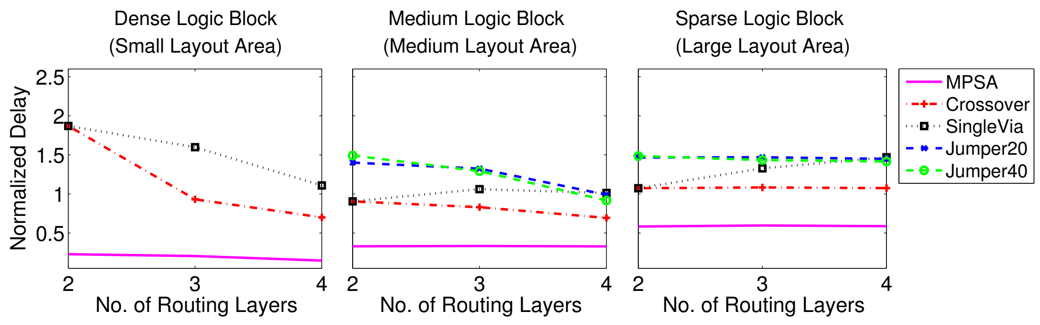


Figure D.7: Delay trends for 6-input, 3-output logic blocks

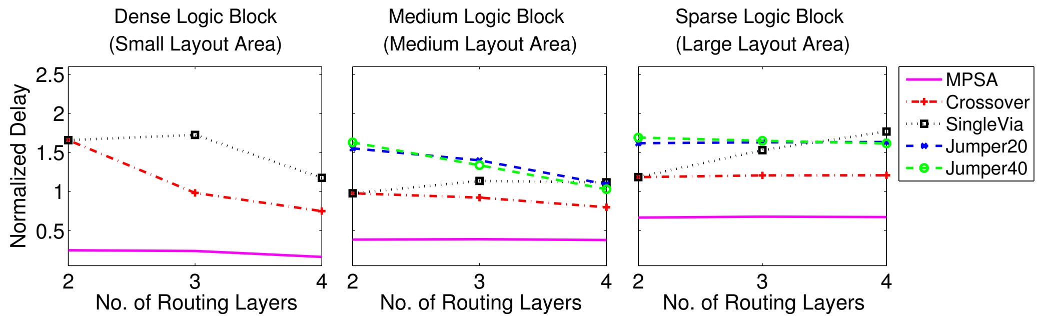


Figure D.8: Delay trends for 8-input, 4-output logic blocks

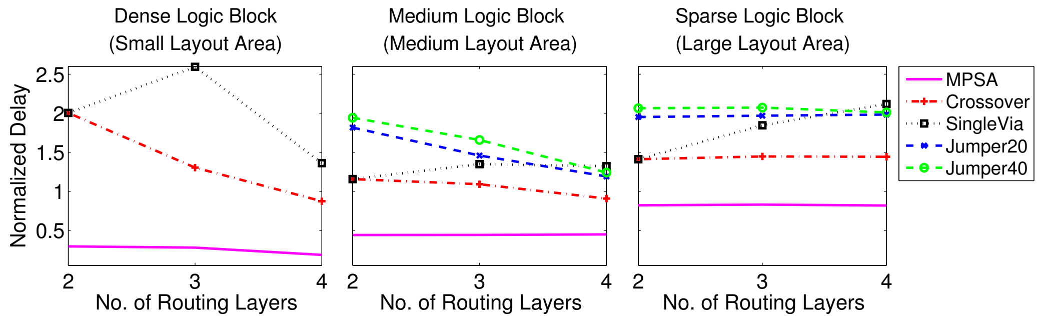


Figure D.9: Delay trends for 12-input, 6-output logic blocks

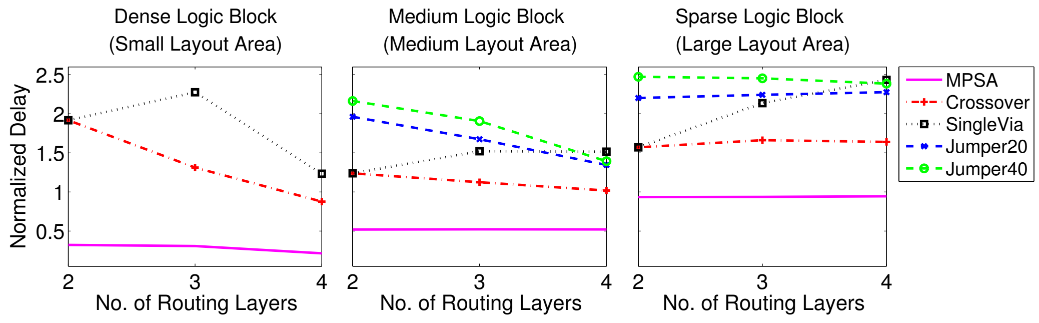


Figure D.10: Delay trends for 14-input, 7-output logic blocks

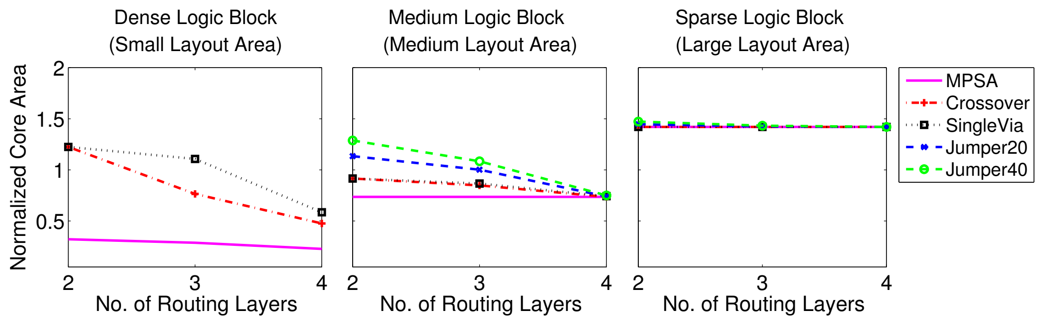


Figure D.11: Area trends for 4-input, 2-output logic blocks

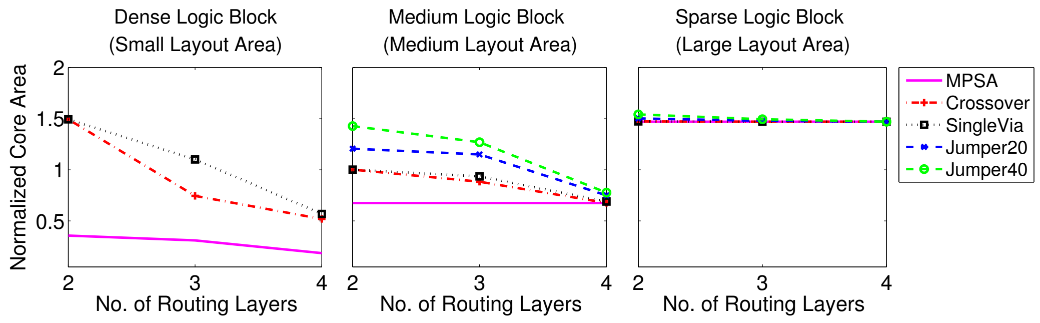


Figure D.12: Area trends for 6-input, 3-output logic blocks

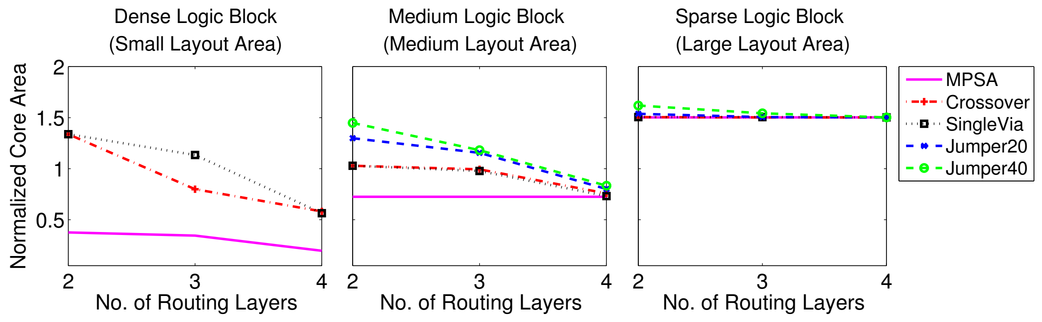


Figure D.13: Area trends for 8-input, 4-output logic blocks

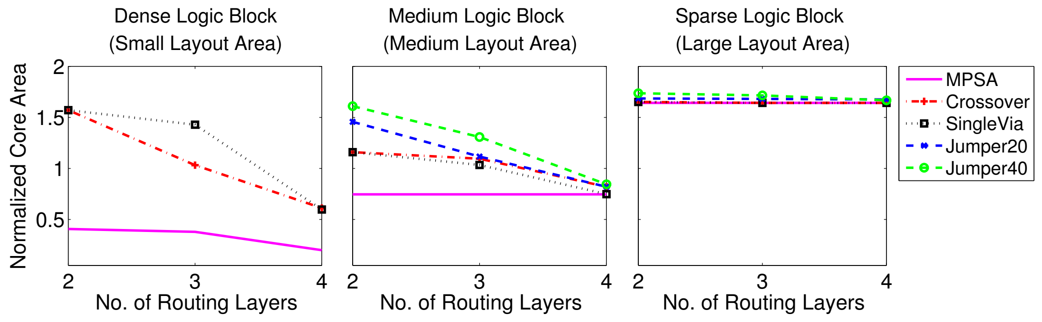


Figure D.14: Area trends for 12-input, 6-output logic blocks

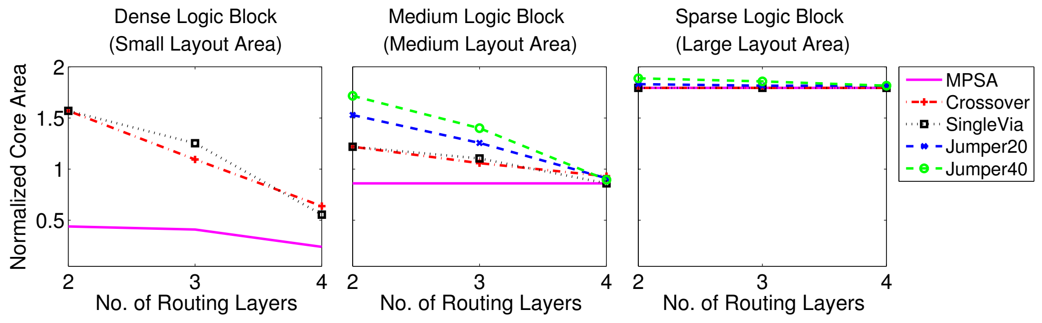


Figure D.15: Area trends for 14-input, 7-output logic blocks

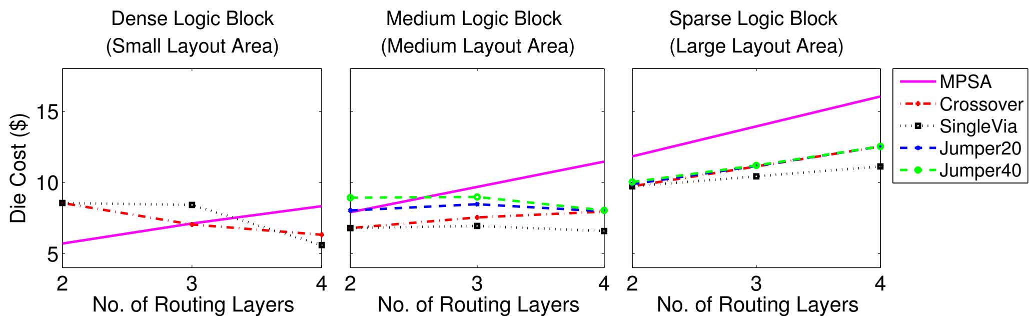


Figure D.16: Die-cost trends for 4-input, 2-output logic blocks

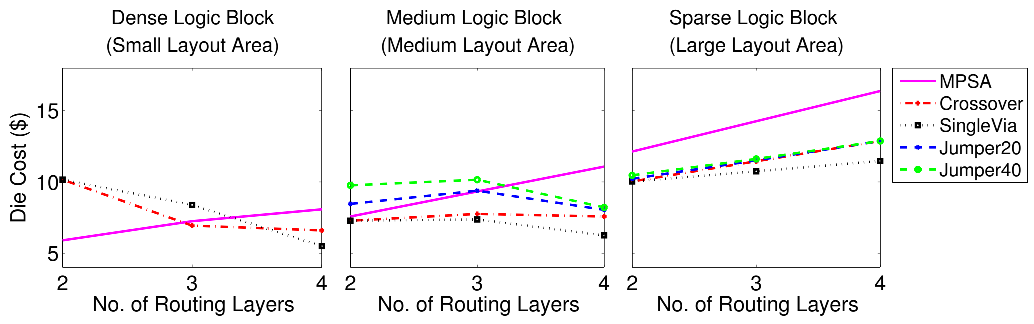


Figure D.17: Die-cost trends for 6-input, 3-output logic blocks

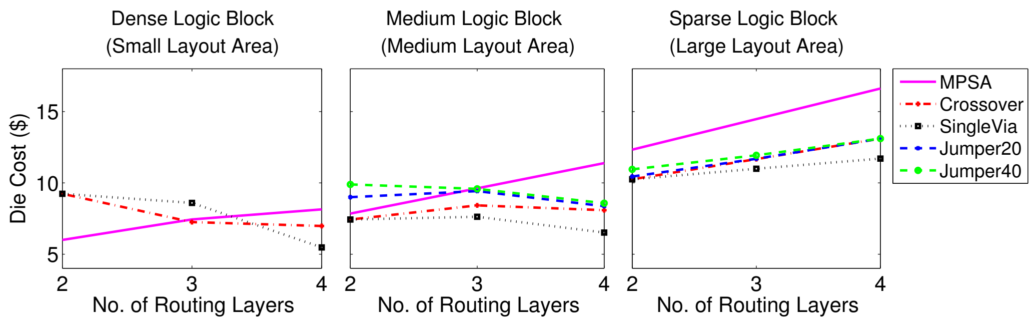


Figure D.18: Die-cost trends for 8-input, 4-output logic blocks

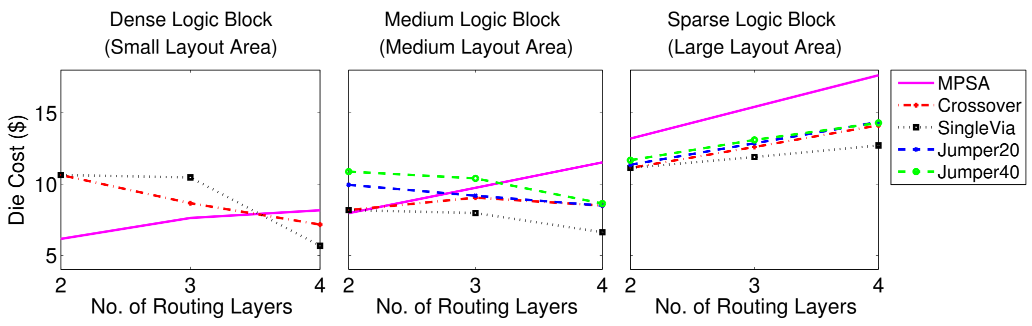


Figure D.19: Die-cost trends for 12-input, 6-output logic blocks

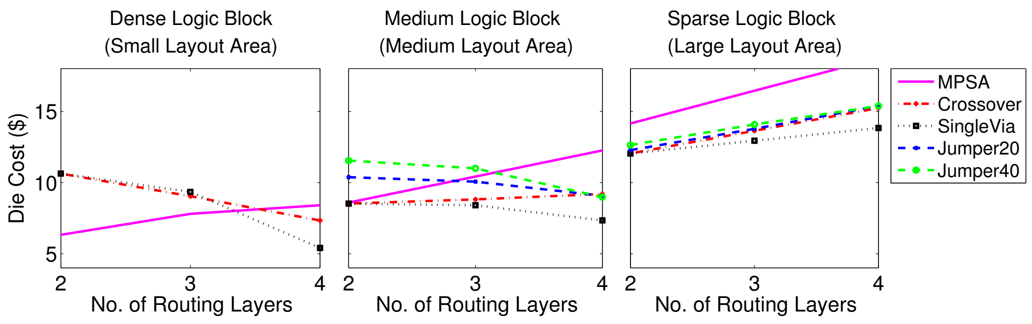


Figure D.20: Die-cost trends for 14-input, 7-output logic blocks