Performance and Cost Tradeoffs in Metal-Programmable Structured ASICs (MPSAs)

Usman Ahmed, Guy G. F. Lemieux, *Senior Member, IEEE*, and Steven J. E. Wilton, *Senior Member, IEEE*

***Abstract*—As process technology scales, the design effort and nonrecurring engineering (NRE) costs associated with the development of integrated circuits is becoming extremely high. Structured ASICs offer one solution to these problems. However, to realize their full potential, their performance and cost advantages, architectures, and CAD must be fully understood. We believe that this can lead to wider adoption of structured ASICs. In this paper, we take a step in this direction and investigate the area, delay, power, and cost tradeoffs in metal-programmable structured ASICs (MPSAs). In particular, we quantify the impact of the number of user-defined (custom) metal mask layers on these metrics. Results indicate that for lowest cost, the number of custom layers should be minimized, especially for small die sizes (e.g., less than 100 mm^2). Delay and power, however, can be improved by a few additional custom layers. With two custom metal layers, MPSAs can be 2×–10× cheaper than cell-based ICs (CBICs).**

***Index Terms*—Structured ASICs, VLSI.**

I. Introduction

APPLICATIONS that require high volume and/or low-power consumption have traditionally been implemented with standard cells as cell-based integrated circuits (CBICs). As process technologies scale to finer geometries, new challenges affecting the design and fabrication of CBICs have emerged. One of these challenges is subwavelength lithography. Other challenges are deep submicron (DSM) effects such as variation, signal integrity, and higher leakage.

These challenges are being mitigated using several approaches. Resolution enhancement techniques (RETs) are used to cope with subwavelength lithography problems. Optical proximity correction (OPC) and the use of phase shift masks (PSM) are two of the commonly used RETs [1]. In these techniques, geometric layout shapes are transformed before fabrication in such a way that the resulting distorted shapes result in intended layout shapes. However, these techniques are very time and memory intensive and significantly increase the cost of producing and inspecting each mask. The DSM effects are mitigated by modeling the physical effects using ever-more sophisticated CAD tools and taking them into account during various design stages. This significantly complicates the CAD tools and results in a significant increase in both the tool cost and the design time/cost. As a result of these issues, access to the latest process technologies is becoming limited and many designs are still being implemented using older process technologies; advanced process technologies, including 90 nm and below, account for only 49% of TSMC's revenue [2].

Field-programmable gate arrays (FPGAs) provide one way of addressing these problems. However, there is a significant gap between the power, delay, and area performance of FPGAs compared to CBICs [3]. 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 can be achieved using CBICs. Structured ASICs are one solution to these problems.

A structured ASIC is a generic IC that is partially fabricated using standard or generic masks and can be "programmed" to implement any digital circuit by adding one or more custom metal layers and/or via layers [4]. The cost of the generic masks, in particular the more expensive lower layer masks, is amortized across a wide range of different designs. This partial fabrication of the device improves the cost and turnaround time. Power consumption is reduced (compared to an FPGA) since programmable switches are not required; in an FPGA, these switches consume significant static and dynamic power. For these reasons, we expect that structured ASICs will become an increasingly important design methodology, especially in platform-based designs and hand-held/battery powered device markets. This advantage will continue to grow at finer processes such as 32 nm and below.

Although structured ASICs were 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 continues to advance, the advantages of structured ASICs will become even more compelling, especially for low-power hand-held applications. When that happens, we will need new architectures, CAD tools, and design flows. In this paper, we take a step in this direction by investigating metal-programmable structured ASICs, or MPSAs.

The cost, turnaround time, performance, and power are the key advantages of structured ASICs. These factors depend upon the number of metal and/or via layers that are used to customize a structured ASIC. Intuitively, we would like to minimize the number of layers that can be used for customization,

Manuscript received April 15, 2010; revised July 07, 2010; accepted August 18, 2010. This work was supported in part by Altera and in part by the Natural Sciences and Engineering Research Council.

The authors are with the Department of Electical and Computer Engineering, University of British Columbia, Vancouver, BC V6T 1Z4, Canada (e-mail: uahmed@ece.ubc.ca; lemieux@ece.ubc.ca; stevew@ece.ubc.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2010.2076841

since this minimizes the cost to the designer and may shorten 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 area and possibly be slower and consume more power. This conflicting criteria suggests that there is an optimum number of layers that must be configurable, and this is the main focus of this paper. The configurability question is important because the early structured ASIC offerings ranged from a single via customization [5] up to 6-metal and 6-via customization [6], and this is the key factor which determines the performance and cost of a structured ASIC.

An early version of this work appears in [7]. This paper is a detailed and enhanced version, which includes the following:

- enhancements to the CAD framework to handle circuits with embedded macro blocks (e.g., memories, register files etc.);
- results for large industrial circuits;
- sensitivity analysis of the cost model; and
- estimation of the performance improvement that might be obtained by an improved CAD flow.

Source code for the CAD framework that also includes modeling of via-programmable structured ASICs (VPSAs) [8], is available at: http://www.ece.ubc.ca/~lemieux/downloads/.

II. Related Work

There has been only a moderate amount of academic research related to structured ASICs where specific logic blocks and routing fabrics have been proposed and evaluated. In this section, we review some of this work.

Ran and Sadowska have proposed a via-configurable structured ASIC [9]–[12]. The logic block is made up of via-configurable cells (VCCs), which are composed of vertically aligned transistor pairs and n-/p- diffusion strips [9]. 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 assumed to be 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 4$\times$ and the delay increase is 1.5$\times$, 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.

In [13], Pileggi *et al.* propose the use of regular structures and compare a via-programmable lookup-table fabric to standard cell designs. Each basic cell in the 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 [14]. The logic block is configured using only lower-layer vias.

Kheterpal *et al.* have explored different routing architectures that can be used with a via-programmable logic fabric [15]. They compared the performance of a structured 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 6-metal process 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. The performance loss for via-configurable routing was 24% and 21%, respectively, for the same two circuits.

Veredas *et al.* have proposed a mask programmable gate array (MPGA) called Zelix [16], [17]. Their goal is to reduce the large area overhead of FPGAs and not to improve the performance. Zelix is based upon 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 block 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 upon 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 structured ASIC known as VPEX, which is designed for electron-beam (EB) direct writing [18]. 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. The architecture is evaluated against a standard cell implementation for small circuits such as a full adder and a 4-b multiplier.

Finally, Chau *et al.* have proposed a via-programmable logic cell called CULG [19]. 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 four or five input functions. 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 three or 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-b multiplier, flip-flop, and a 3-input NAND.

TABLE I
COMMERCIAL STRUCTURED ASICs

Company	Product	Type	Comments
Altera	Hardcopy Series	MPSA	2 custom metal layers; aimed at FPGA-to-ASIC conversion [20]
eASIC	Nextreme Series	VPSA	Customized using a single via layer; 4-6 week turnaround time [5]
Tier Logic	-	-	Aimed at FPGA-to-ASIC Migration [21]
ChipX	CX6200	MPSA	Customized by 2 to 4 metal layers [22]
Faraday	MPCA	MPSA	Customized by 3-metal, 2-via layers; targets SoC based communication systems [23]
Fujitsu	AccelArray	MPSA	Customized using 3 to 4 metal layers [24]; discontinued
Lightspeed	-	MPSA	Different customization levels ranging from 2-metal, 2-via to 6-metal, 6-via[6]; discontinued
LSI Logic	RapidChip	MPSA	Only the transistors are fabricated, all the remaining layers can be customised [25]; discontinued
NEC	ISSP	MPSA	Customized using two metal layers; targets SoC systems [26]; discontinued
ON Semiconductor	Xpress Array-II	MPSA	150nm structured ASIC aimed at FPGA-to-ASIC migration [27]; formerly AMI Semiconductor
ViASIC	ViaMask, DuoMask	VPSA	Customized using single mask or two masks; targets SoC systems [28]
Virage Logic	ASAP	MPSA	Metal programmable cell libraries that can be customized using 3 or 4 metal layers [29]

There are a number of commercial vendors who have offered structured ASIC products. These products are customized either by both metal and via layers (MPSAs) or only through via layers [via-programmable structured ASICs (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 SoC based designs. Table I shows several of these products. Unfortunately, detailed information about most of these products is not published. Interestingly, products that had high amount of configurability have been discontinued.

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 based 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.

III. Research Problem and Approach

The research question we answer in this paper is: *how does the amount of configurability affect the efficiency of an MPSA?* More specifically, we relate the number of configurable metal and via layers to the performance, power, die-area, and dollar-cost of the MPSA. Intuitively, more configurable layers will result in less routing congestion, possibly leading to faster, smaller, and more power-efficient circuits. However, more configurable layers also result in a higher dollar-cost for each fabricated device. Understanding this tradeoff is key to creating efficient and cost-effective MPSAs.

In answering this question, we employ an experimental approach. We consider a set of potential MPSA architectures; each architecture in the set differs in the number of configurable layers available. Each MPSA is then modeled at a low level of detail, and custom CAD tools are 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 can be 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 if each potential architecture was laid out and/or manufactured.

An important part of our experimental framework is a detailed cost model which relates the die-area and number of configurable layers to the dollar-cost of an MPSA. The cost model considers the manufacturing cost of each device, the mask-set cost for a design, and device volume requirements. This model is described in Section IV. Section V then describes the CAD tools used in our experiments, and Section VI presents the experimental results.

IV. Cost Model

This section describes our detailed cost model, which relates the cost per die (C_{die}) to the number of configurable layers in a structured ASIC and the die area. The cost per die depends upon more than just the die area; a larger die with fewer layers to be customised may be less expensive than a smaller die with more customizable layers.

To estimate C_{die}, we write

$$C_{\text{die}} = C_{\text{base}} + C_{\text{custom}} + C_{\text{proto}} + C_{\text{pkg}} + C_{\text{test}} \quad (1)$$

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 prefabricated 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 paper, we assume that C_{pkg} and C_{test} are constants in our experiments, so they are not considered in our C_{die} calculations; they do depend upon the user's design, but they do not depend upon the range of SA implementations we consider (i.e., area or number of configurable layers).

The base, customization, and prototyping costs can be further subdivided into three parts: 1) a nonrecurring cost of preparing the mask sets, 2) cost of setting up the fab line, and 3) wafer processing cost. C_{base} can be expressed as

$$C_{\text{base}} = \overbrace{\left(\frac{C_{sm_l}N_{fm_l} + C_{sm_u}N_{fm_u}}{V_{\text{tot}}}\right)}^{\text{Mask costs}} + \overbrace{\left(\frac{C_{fs_1}}{V_{\text{tot}}}\right)}^{\text{Fab setup cost}} + \underbrace{\left(\frac{C_{wpm}N_{fm_l} + C_{sw}}{N_{gdpw}}\right)}_{\text{Wafer costs}}$$

where N_{fm_l} is the number of lower fixed masks, N_{fm_u} is the number of upper fixed masks (e.g., required for power grid), C_{sm_l} is the average cost for a single lower-level mask (e.g., poly 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 and C_{fs_1} is the fab setup cost of the SA device for all customers, C_{wpm} is the wafer processing cost for a single mask, C_{sw} is cost of single unprocessed wafer, and N_{gdpw} is the number of good dies per wafer.

Different lower-level masks (e.g., diffusion and poly masks) may have different costs. However, acquiring these cost values is not always possible. These would also complicate the cost model. Therefore, we have decided to use one average cost number for C_{sm_l} and a different average cost value for C_{sm_u}.

The customization cost (C_{custom}) can be calculated in a similar fashion as

$$C_{\text{custom}} = \overbrace{\left(\frac{C_{sm_u} N_{cm}}{V_c}\right)}^{\text{Mask costs}} + \overbrace{\left(\frac{C_{fs_2}}{V_c}\right)}^{\text{Fab setup cost}} + \underbrace{\left(\frac{C_{wpm}\left(N_{cm} + N_{fm_u}\right)}{N_{gdpw}}\right)}_{\text{Wafer costs}}$$

where, N_{cm} is the number of custom masks, and V_c is the volume per customer. N_{cm} can be calculated as

$$N_{cm} = N_{rl} \times N_{mprl}$$

where N_{rl} is the number of routing layers, and N_{mprl} is the number of masks needed for each layer. In an MPSA, a routing layer consists of one metal layer and one via layer.

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, the prototyping costs are calculated as

$$C_{\text{proto}} = \overbrace{(N_s - 1)\left(\frac{C_{sm_u} N_{cm}}{V_c}\right)}^{\text{Mask costs}} + \overbrace{(N_s - 1)\left(\frac{C_{fs_2}}{V_c}\right)}^{\text{Fab setup cost}} + \underbrace{\left(\frac{N_s - 1}{V_c}\right)\left(C_{wpm}\left(N_{fm_l} + N_{cm} + N_{fm_u}\right) + C_{sw}\right)}_{\text{Wafer cost}}.$$

In C_{proto}, we include the cost to manufacture one complete wafer for every prototype spin, excluding the final spin. Although minimum lot sizes offered 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 multiproject wafer, where each customer uses less than a full wafer. This could reduce the wafer cost component of the prototype to nearly zero. In our previous work [7], we had implicitly set this wafer cost to zero, but the difference this has on results is very small.

We are interested in analyzing the sensitivity of the cost function to the number of configurable routing layers (N_{rl}) and the die area (A_{die}). By substituting the values of C_{base}, C_{custom}, C_{proto} and N_{cm} in (1) and rearranging the terms, C_{die} can be written as

$$C_{\text{die}} = \frac{K_0}{N_{gdpw}} + N_{rl}\left(\frac{K_1}{N_{gdpw}} + K_2\right) + K_3 \qquad (2)$$

TABLE II
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 [34]; 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 [35] (2) Cost of lower level masks is 3x that of upper level masks
C_{sm_u}	\$36k	*Average single mask cost for upper layers* [34]
V_{tot}	2M	*Total volume*
C_{wpm}	\$220	*Wafer processing cost per mask* (1) Cost to process a 45nm wafer: \$8000 (2) 34 masks total
D_{waf}	$300mm$	*Wafer diameter*
P_w	$150\mu m$	*Pad width*
S_w	$100\mu m$	*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	*Number of fixed masks above the configurable masks (e.g., for power grid)*
Y_0	0.9	*Material and systematic yield* [32]
N_{mprl}	2	*Number of masks per routing layer* One mask each for metal and via
D_0	1395 per cm^2	*Defect rate* [32]
α	2.0	*Cluster factor* [32]
$\frac{C_{fs_1}}{V_{tot}}$, $\frac{C_{fs_2}}{V_c}$	\$0	C_{fs_1}, C_{fs_2}: *Fab line setup costs* Any fab setup costs can be ignored, esp. when these are 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* These costs are 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.

where K_0, K_1, K_2, and K_3 are constants that depend upon the volume requirements and various foundry costs, but are fixed for a given structured ASIC product. Their values are

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

Using the parameter values shown in Table II, typical values for K_0, K_1, K_2, and K_3 are \$4400, \$440, \$1.4444, and \$1.043, respectively.

A. Yield Model for N_{gdpw}

The number of good-dies-per-wafer (N_{gdpw}) depends upon number of dies per wafer (N_{dpw}) and die yield (Y_{die}), and is given as

$$N_{gdpw} = N_{dpw} \times Y_{\text{die}}$$

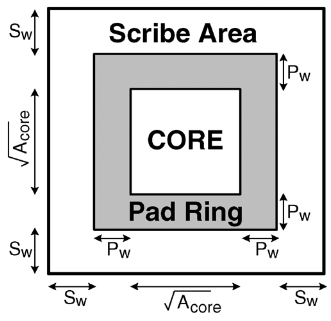


Fig. 1. Core, pad, and scribe area.

The number of dies per wafer can be approximated as [31] and [32]

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

In this equation, D_{waf} is the wafer diameter, A_{core} is the core area, A_{io} is the area for input and output pads, and A_{scribe} is the scribe area. A scribe is a ring around the die reserved for wafer testing and die cutting; it mostly influences the area of small dies. These three components are illustrated in Fig. 1. If P_w and S_w represent the pad width and scribe width, respectively, then A_{io} and A_{scribe} can be calculated as

$$A_{io} = 4P_w(P_w + \sqrt{A_{\text{core}}})$$
$$A_{\text{scribe}} = 4S_w(S_w + 2P_w + \sqrt{A_{\text{core}}}).$$

The die yield can be estimated as [31] and [32]

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

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; each additional layer may cause the yield to reduce. On the other hand, the regularity in MPSA fixed layers 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}.

Most of the parameters in the previously mentioned cost model 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. Table II shows the parameter values we use to estimate C_{die}. We obtained and confirmed data from various sources, including several news articles and contacts in industry. In Section VI-B, we provide a detailed sensitivity analysis of the cost model to various parameters of Table II.

For a range of values of A_{core} and N_{rl}, the output of the cost model is shown in Fig. 2. The iso-cost curves in Fig. 2 show that to maintain constant cost, one extra routing layer must save about 15 mm^2 of die area. This is because the large mask and wafer processing costs associated with each additional layer significantly increases the die-cost.

In Fig. 3, we show the die-yield and the die-cost of CBICs and MPSAs as a function of core area. In calculating the CBIC cost,

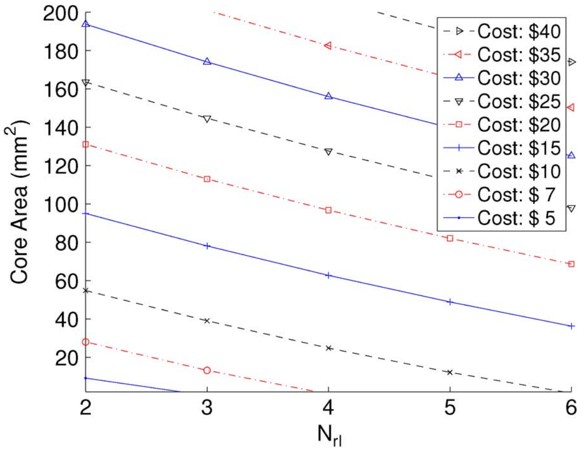


Fig. 2. MPSA cost model.

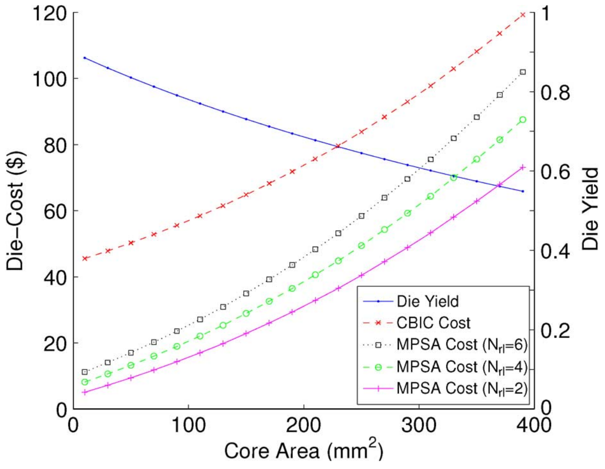


Fig. 3. Yield and die-cost ($\text{Total CBIC Volume} = V_c$).

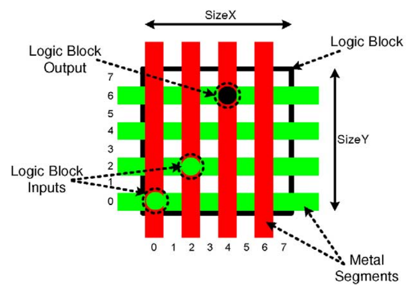
we assumed six routing layers and every mask to be custom. We also assume that during a respin, all the CBIC masks are changed whereas only $N_{rl} \times N_{mprl}$ masks are changed for MPSAs. It is possible that a CBIC respin can be completed without modifying all the masks by employing some of the engineering change order (ECO) techniques [33]. However, we do not take this into account. In Fig. 3, it is better to compare the area values of MPSAs and CBICs for a given cost, rather than comparing the cost values for a given area since the CBIC area will generally be less than the MPSA area for a given design. For example, at a fixed cost of \$45, an MPSA implementation can use nearly 200 mm^2 whereas a CBIC implementation can use only a few mm^2. However, this difference becomes smaller as the die-cost increases. Thus, for large die sizes, MPSAs must be very area efficient to compete with CBICs.

V. Framework

In this section, we describe how we model an MPSA architecture, our CAD flow, and the statistics that we collect.

A. Architecture Model

When modeling the logic block architecture, we prefer to model it without worrying about the low-level, layout related de-



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

Fig. 4. Modeling an architecture.

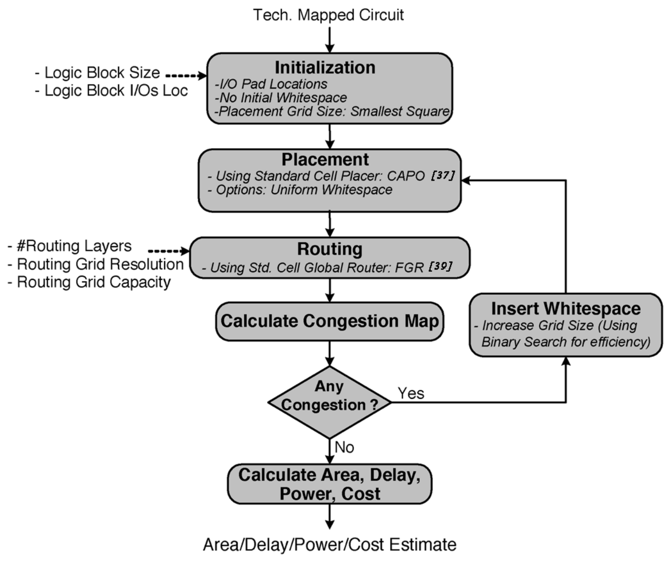

Tech. Mapped Circuit
- Logic Block Size
- Logic Block I/Os Loc
Initialization
-I/O Pad Locations
-No Initial Whitespace
-Placement Grid Size: Smallest Square
Placement
- Using Standard Cell Placer: CAPO [37]
- Options: Uniform Whitespace
- #Routing Layers
- Routing Grid Resolution
- Routing Grid Capacity
Routing
- Using Std. Cell Global Router: FGR [39]
Calculate Congestion Map
Insert Whitespace
- Increase Grid Size (Using Binary Search for efficiency)
Any Congestion ?
Yes
No
Calculate Area, Delay, Power, Cost
Area/Delay/Power/Cost Estimate

Fig. 5. CAD flow.

tails. From the perspective of the interconnect, the various logic block options differ only in their physical size and the number of inputs and outputs. Therefore, we abstract the logic block as a rectangular block with a certain number of pins on it. The logic block size (height and width) and the position of pins are specified in terms of wire half-pitches. Fig. 4 illustrates the modeling process for a 2-input logic block.

B. *CAD Flow*

Our CAD flow is shown in Fig. 5. The flow starts with 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 $\lceil\sqrt{N}\rceil \times \lceil\sqrt{N}\rceil$). We then perform placement, route the circuit for a given number of routing layers, and calculate routing congestion. If there is any congestion, we increase the placement grid size and repeat these steps. The placement and routing stages are described in the following subsections.

1) Placement: The MPSA placement problem is similar to the FPGA placement problem because all of the prefabricated logic blocks have the same size and are arranged on a grid. This implies that an FPGA placer, e.g., VPR [36], may be suitable. However, there are two problems with this approach. First, the number of blocks to be placed can be fairly large, especially in the case of fine-grained logic blocks. We have found that the simulated annealing placement algorithm of VPR is slow with such large circuits. Second, the wirelength based cost function in VPR does not allow it to insert whitespace[1] to remove congestion. Whitespace insertion is crucial, because with small logic blocks, and the routing being done on top of them, MPSAs are more likely than FPGAs to experience congestion. For these reasons, we use a standard cell ASIC placer which is faster and can insert whitespace.

We are using the CAPO [37] standard cell placer. It 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 replace 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.

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 replace the logic blocks.

If the logic fabric has dedicated flip-flops, a third pass can substantially improve wirelength. We consider these flops as hard macro blocks and lock their position after the first placement pass. Hence, the second pass only changes logic positions not flops. The third pass moves only the flip-flop blocks. 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 [38]. 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.

2) Routing: After placement, the next step is to route all the nets to estimate wirelength. In our flow, we use the FGR global router [39]. In addition to the list of nets to route, the inputs to the router include the number of available metal and via 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).

The MPSA routing problem is very similar to the ASIC routing problem. Detailed routing in ASICs confines the connections to the given global routing and deals mainly with meeting the design rules [40]; in general, the quality of the

[1]By whitespace, we mean an entire empty logic block.

routing results is dictated entirely by the global route. Therefore, to simplify the flow we do not perform detailed routing. We constrain the global router so that it can only use up to 85% of 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.

C. Metrics

The metrics we use to compare different configurability choices include core area, delay, power and manufacturing cost. The core area is calculated by multiplying the logic block area with the size of the placement grid.

We use the Elmore delay model to estimate the delay [41]. 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 critical-path driven.

For the power metric, we are concerned with the dynamic power dissipated in the interconnect since this is the only component of power that would change significantly as we vary the number of routing layers. We use the total interconnect (metal and via) capacitance as a first-order estimate for power.

Finally, we use the cost model described in Section IV to estimate the manufacturing cost of the die.

VI. Results

In this section, we show the impact of the number of programmable layers on the cost, area, speed, and power of the MPSA device. The experimental results are presented for two different suites of benchmarks: homogeneous circuits that consist of only one type of logic cell, and heterogeneous circuits that contain up to one million logic cells along with different IP blocks (block RAM, register files, etc.). We also study the sensitivity of the cost model to various parameters of Table II. Finally, we estimate the effect of an using an improved white-space insertion algorithm on MPSA die-cost.

A. Power, Delay, Area, and Cost Trends

1) Homogeneous Circuits: For homogeneous circuits, we used the 19 largest MCNC benchmark circuits[2] that have commonly been used in the research on FPGAs [36] and structured ASICs [12].

The flow described in the previous section assumes a technology-mapped circuit, however, the technology mapping depends upon the internal structure of each physical cell in the MPSA. In order to focus our attention on the interconnect architecture, we abstract the contents of the cell by representing only its input and output pins and cell 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 written 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 ([36]), an FPGA clustering algorithm, for this purpose.

[2]One of the circuits, s38584.1, contains a net with more than 3000 pins which was too large for the router. We chose to exclude the benchmark rather than modify it.

TABLE III
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	8x8	12x12	15x15	19x19	22x22
4	2	12x12	17x17	22x22	27x27	32x32
6	3	12x12	19x19	26x26	33x33	39x39
8	4	16x16	23x23	30x30	37x37	44x44
10	5	16x16	25x25	33x33	42x42	50x50
12	6	20x20	29x29	37x37	46x46	54x54
14	7	20x20	30x30	40x40	50x50	59x59
16	8	20x20	31x31	42x42	53x53	63x63
7	2	*Described in section VI-A2*				

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 (I/O counts) for each layout density.

Our experimental methodology also requires the pin locations and an estimate of the layout area (height and width) for each physical cell. Pin locations are randomly generated within each cell. 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 would not be able 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 III shows the different logic block *types* (I/O counts) and the corresponding layout area values used in our experiments.

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 Fig. 6. The plots show averaged (geometric mean)

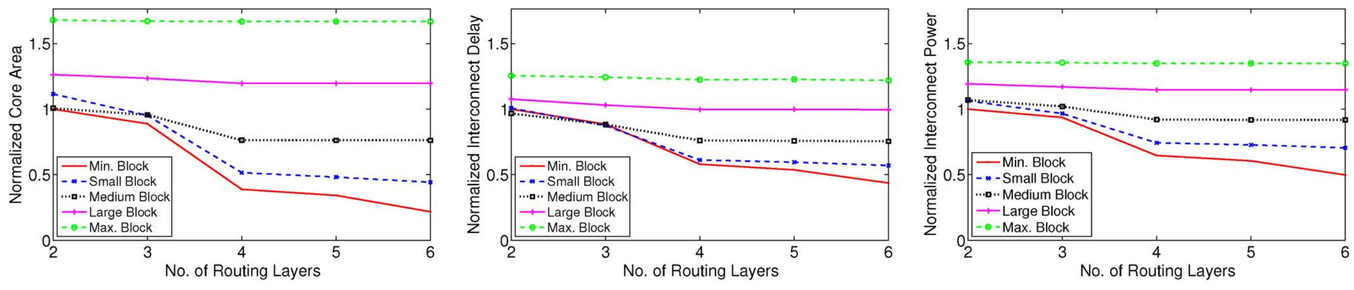


Fig. 6. MCNC circuits: area, delay and power trends (nominal core area at $45\ \mathrm{nm} = 0.008\ \mathrm{mm}^2$). (a) Area. (b) Delay. (c) Power.

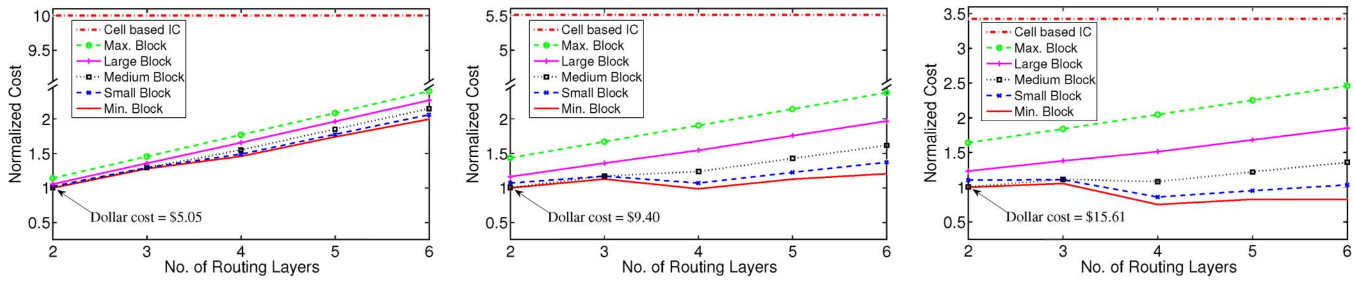


Fig. 7. MCNC circuits: trends for die cost at 45 nm. (a) $\mathrm{Nominal\ Core\ Area} = 10\ \mathrm{mm}^2$. (b) $\mathrm{Nominal\ Core\ Area} = 50\ \mathrm{mm}^2$. (c) $\mathrm{Nominal\ Core\ Area} = 100\ \mathrm{mm}^2$.

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 core area of the minimum block layout area with two routing layers. The nominal area in these plots is 0.008 mm^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 Fig. 6(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 Fig. 6 is examined for individual logic block types (I/O counts), but these data are not shown due to space constraints.

Next, we estimated the dollar cost by applying the cost model described in Section IV. However, the homogeneous circuits we used are quite small. This is impractical, and artificially increases (N_{gdpw}) significantly, reducing (2) to $N_{rl}K_2 + K_3$. Because of this, we scaled the core area to a realistic value before applying the cost model.[3] For scaling, we multiplied the core areas such that the nominal core area gets the values of 10 mm^2, 50 mm^2, and 100 mm^2. The resulting cost plots are shown in Fig. 7. It can also be seen from Fig. 7 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 Fig. 7, 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.

Finally, we compare the cost of implementing a design in an MPSA and a CBIC. We consider two different process technologies—90 nm and 45 nm. The area of the 90 nm implementation is 4$\times$ the area of 45 nm implementation. For MPSA costs we assumed a "medium" block layout area whereas for CBIC we assumed "min" block layout area. With these assumptions, the CBIC implementation of a design takes 3.5$\times$ less area than the MPSA implementation in the same process technology. The ratio of CBIC costs to MPSA costs are then shown in Fig. 8. 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 90 nm CBIC implementation versus a 45 nm MPSA implementation. Again, MPSAs are much cheaper than a CBIC implementation, especially for

[3]In Section VI-A-2, we show results that did not involve any scaling.

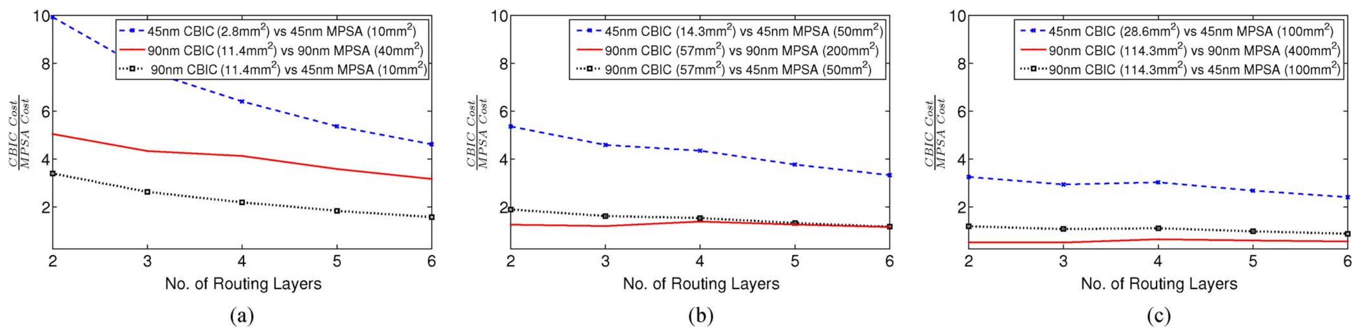


Fig. 8. Cost advantage of MPSAs over CBIC at 90 nm and 45 nm (higher value means MPSA is lower cost). (a) $45\ \mathrm{nm}\ \mathrm{MPSA\ Core\ Area} = 10\ \mathrm{mm}^2$, $90\ \mathrm{nm}\ \mathrm{MPSA\ Core\ Area} = 40\ \mathrm{mm}^2$. (b) $45\ \mathrm{nm}\ \mathrm{MPSA\ Core\ Area} = 50\ \mathrm{mm}^2$, $90\ \mathrm{nm}\ \mathrm{MPSA\ Core\ Area} = 200\ \mathrm{mm}^2$. (c) $45\ \mathrm{nm}\ \mathrm{MPSA\ Core\ Area} = 100\ \mathrm{mm}^2$, $90\ \mathrm{nm}\ \mathrm{MPSA\ Core\ Area} = 400\ \mathrm{mm}^2$.

TABLE IV
eASIC BENCHMARKS: CHARACTERISTICS

Circuit	**Original Circuits**						**Packed Circuits**				
	Nets	**Logic Blocks**	**Flip Flops**	**Block RAMs**	**Reg. Files**	**Avg. Used Pins** *(Total Pins=9)*	**Nets**	**Logic Blocks**	**Blocks Clustered** Max.	**Blocks Clustered** Avg.	**Avg. Used Pins**
easic1	930,116	832,824	87,052	110	172	3.18	723,455	318,448	14	2.64	5.99
easic2	873,483	812,200	45,478	175	686	3.08	658,846	304,520	14	2.67	5.87
easic3	1,016,364	961,063	52,780	192	0	2.99	533,161	261,366	19	3.68	6.41
easic4	126,224	102,038	23,330	0	44	3.06	82,822	31,746	13	3.22	6.27
easic5	1,010,422	913,853	84,505	145	262	3.14	738,587	332,341	20	2.77	5.99

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 90 nm; this may partly explain the slower than anticipated adoption rate of structured ASICs to date.

2) Heterogeneous Circuits: For heterogeneous circuits, we used circuits that were released by eASIC as part of a placement contest [42]. These circuits are modified versions of large industrial designs and contain up to approximately one million logic blocks. The logic fabric consists of four different types of elements: ecells (logic block), flip-flops, block RAMs, and register files. The circuits have been technology mapped to these blocks but the internal architecture of these blocks has not been disclosed. There are multiple clock domains in these circuits, however, for our experiments we only assume a single clock domain.

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 it can have four different clocks. The chip is made up of array of groups and can have 32 different clocks.

The original technology mapping of the eASIC circuits is very sparse. This can be seen from the last column under “Original Circuits” in Table IV. The logic block has nine pins (seven input pins and two output pins), but the circuits, on the average, are only using three 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 (Fig. 6). Therefore, we modified the circuits by clustering the logic blocks to make the mapping more dense using the T-VPack algorithm [36]. The characteristics of the original and the packed circuits are shown in Table IV.

To conduct the experiments, we need an estimate of the layout area for different circuit elements. 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 (36 kb dual-port memory), and used that to determine the layout area of 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 area values of these blocks, in terms of wire half-pitches, are 50×50, and 96×96, respectively.

We pass four of the circuits through the CAD flow described in Section V. 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 number of customizable layers and use the cost model described in Section IV to calculate the die-cost. The plots for the average (geometric) area, delay, and power trends are shown in Fig. 9(a)-(c). 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 customizable routing layers. The bulk of the improvement occurs in going from two to four 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 four to six layers is only 12%, 11%, and 13%, respectively. The trends for other block sizes are similar.

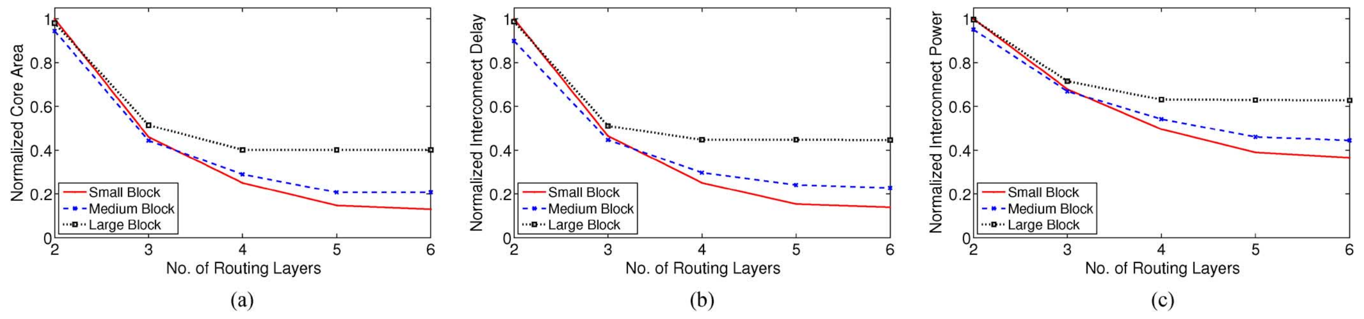


Fig. 9. Packed eASIC circuits: area, delay and power trends ($\text{Core area with small block and two routing layers} = 162\ \text{mm}^2$). (a) Area. (b) Delay. (c) Power.

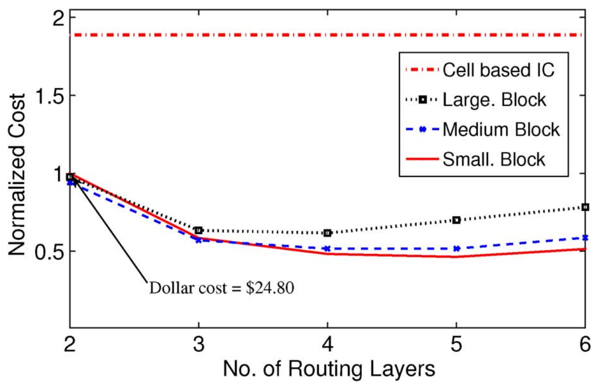


Fig. 10. Packed eASIC circuits: die-cost trend (normalized to cost values for “small block”).

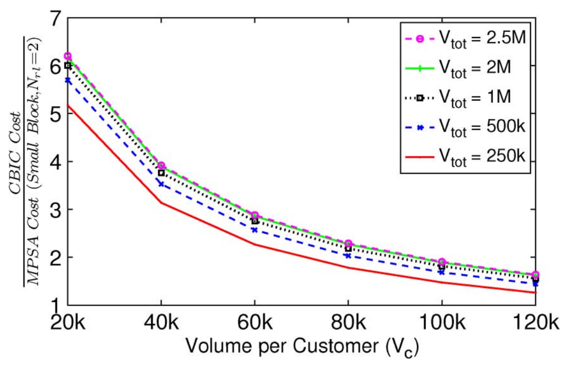


Fig. 11. Die-cost sensitivity to volume requirements.

Second, with fewer custom routing layers, the difference between the different block sizes is small but it grows with more custom layers. With two custom 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 four layers is 15%, 20%, and 13% respectively; with six layers, the difference grows to 27%, 31%, and 26%, respectively.

Third, area is most sensitive to the number of customizable routing layers, whereas power is least sensitive. The reduction in area and power in going from two to six layers with small blocks is 80% and 60%, respectively.

Finally, we see that with two custom routing layers, the area, delay and power with small blocks are more than the medium block. We also see that in going from two to three layers, there is a significant performance improvement. The reason for both these observations is 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 exactly the case with fewer routing layers and/or smaller block sizes. In Section VI-C, we provide insight into the improvement that can be obtained from the use of an intelligent whitespace insertion algorithm.

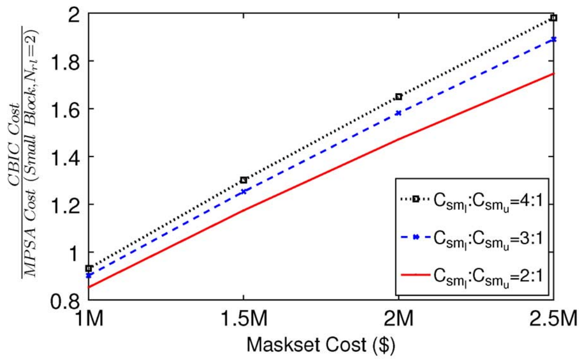
Next, we estimate the die-cost using the area values of Fig. 9(a). The resulting plot is shown in Fig. 10. The plot shows that the decrease in core area with more custom routing layers does not reduce the die-cost by the same proportion. It can be seen that the reduction in die-cost obtained by having more than three custom layers is very small and there is almost no cost advantage of having more than four custom routing layers. The reason for this behavior is the large cost associated with the maskset; 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 six routing layers) and consider all masks as custom. The resulting cost is also shown in Fig. 10. It can be seen that MPSAs with two custom routing layers have a 2× cost advantage over CBICs, and with four custom layers it grows to about 4×.

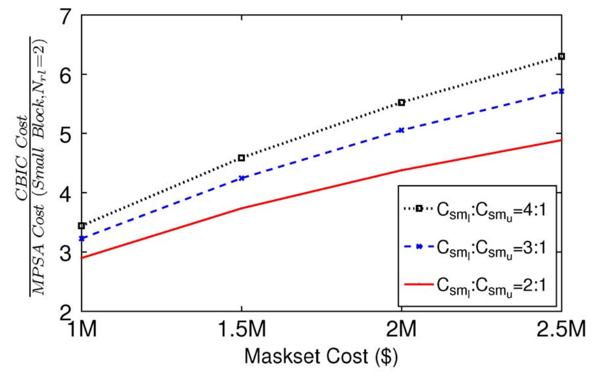
B. Cost Sensitivity

In this section, we study the sensitivity of the die-cost to some of the parameters of Table II. In particular, we look at the effect of different volume requirements (V_c and V_{tot}), maskset prices (C_{sm_l} and C_{sm_u}), and number of fixed masks (N_{fm}). We have noticed that the trends for different MPSAs (different logic block sizes and different number of custom layers) are largely insensitive to these parameters. However, the cost of MPSAs relative to CBICs does change. Therefore, we only compare the die-cost of 45 nm CBICs against the 45 nm MPSA with small block and two custom layers and show the results for heterogeneous circuits.

The sensitivity of die-cost to volume requirements is shown in Fig. 11. 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



(a)



(b)

Fig. 12. Die-cost sensitivity to maskset cost. (a) $V_c = 100\ \text{k}$, $V_{\text{tot}} = 2\ \text{M}$. (b) $V_c = 20\ \text{k}$, $V_{\text{tot}} = 500\ \text{k}$.

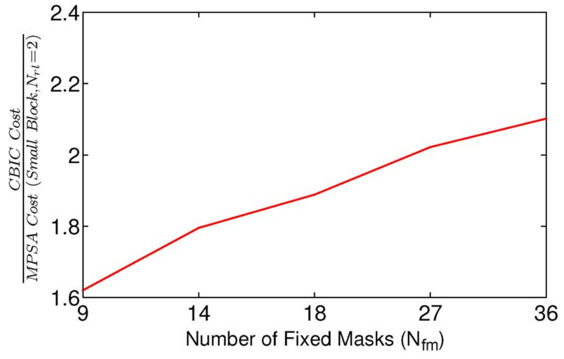


Fig. 13. Die-cost sensitivity to number of fixed masks (N_{fm}).

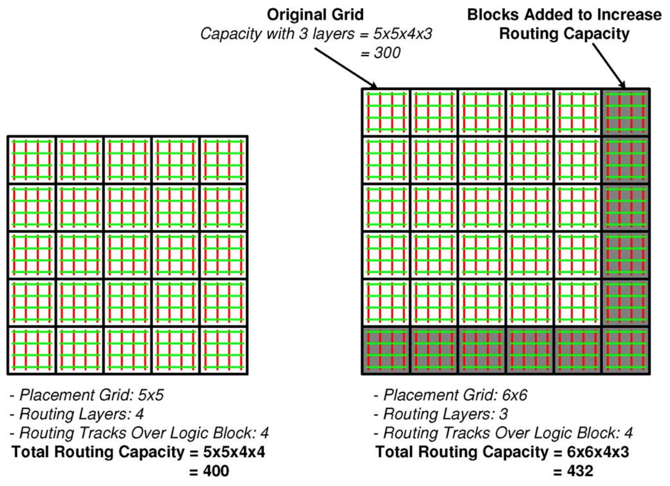

Original Grid
Capacity with 3 layers = 5x5x4x3 = 300
Blocks Added to Increase Routing Capacity
- Placement Grid: 5x5
- Routing Layers: 4
- Routing Tracks Over Logic Block: 4
Total Routing Capacity = 5x5x4x4 = 400
- Placement Grid: 6x6
- Routing Layers: 3
- Routing Tracks Over Logic Block: 4
Total Routing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

Fig. 14. Estimating area with use of an intelligent whitespace insertion algorithm.

CBIC maskset cost is amortized over the customer volume only. For small volumes, MPSAs are, therefore, very cost effective.

Next, we look at the impact of maskset cost. There are two factors in the mask set cost. First, the total maskset cost, and second, the ratio of the cost of 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 Fig. 12. It can be seen that the higher maskset costs favor MPSAs, especially for smaller volumes. Also, an increasing ratio between C_{sm_l} and C_{sm_u}, which matches current trends, favors MPSAs.

Finally, we also modeled different processes in which the number of masks needed to manufacture the fixed portion of the device (N_{fm}) may differ. The results, shown in Fig. 13, illustrate that a larger number would favor MPSAs over CBICs. This is because, with large N_{fm}, a larger portion of the cost of the maskset is amortized over total device volume (V_{tot}). This lowers the per-die cost of MPSAs.

C. Effect of an Improved Whitespace Insertion Algorithm

We are using uniform whitespace insertion in our CAD flow. As described in Section VI-A-2, 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.

Congestion-aware whitespace allocation problem has been studied before, both for CBICs [43]–[45] and FPGAs [46]–[49]. However, it still remains an active area of research. In FPGAs, the whitespace insertion problem is particularly hard because whitespace can only be inserted at fixed grid locations and in units of LUTs or CLBs. The nature of the whitespace insertion problem in MPSAs is similar to that of FPGAs. In FPGAs, the use of empty CLBs as whitespace has not been very successful. Instead, most of the techniques rely on depopulating CLBs (using fewer than available LUTs) [47]–[49]. 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 paper. 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 custom routing layers for which a given circuit can be routed without any whitespace insertion is L. For architectures with fewer than L custom 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 custom layers, since each empty logic

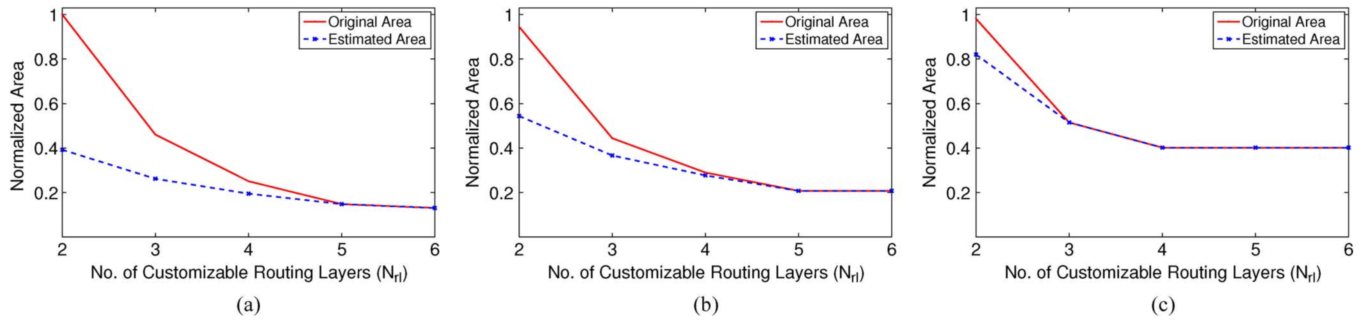


Fig. 15. Estimated area for packed eASIC benchmarks. (a) Small block. (b) Medium block. (c) Large block.

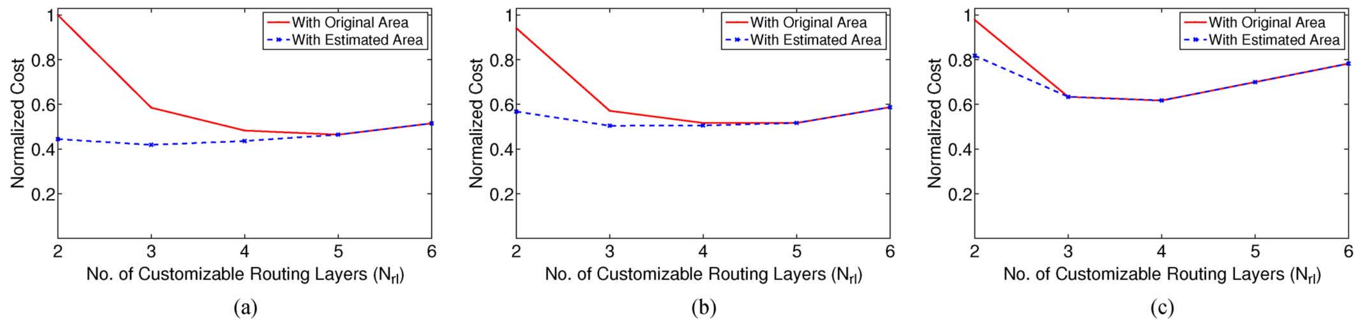


Fig. 16. Estimated die-cost with use of an intelligent whitespace insertion algorithm. (a) Small block. (b) Medium block. (c) Large block.

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' custom 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' custom layers is same as the total number of available routing tracks in the architecture containing L custom layers. The die area and dollar cost of an architecture with N additional logic blocks but only L' custom 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 custom 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 Fig. 14 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 actual benefit will likely be less.

The die-area values obtained by using the previous technique for heterogeneous circuits are shown in Fig. 15 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 custom layers available for routing. The most potential for area savings is with an architecture containing a small block where the estimated die-area for two custom layers is 60% less than what was obtained using uniform whitespace allocation. This difference reduces to 7% if four custom layers are available.

The die-cost values corresponding to the estimated area values are shown in Fig. 16. As the graph shows, the 60% area saving (due to improved whitespace insertion) for the small block with two layers, translates to a 55% cost reduction. However, the area reduction in going from two to four 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. 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 four layers to three layers.

From Fig. 9, 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 cell, especially with fewer routing layers. In the future, however, as better CAD techniques evolve, densely laid-out logic blocks will become advantageous.

VII. Conclusion

This paper has presented area, delay, power, and cost trends for MPSAs. Area is the most sensitive, whereas the power is the least sensitive to the number of customizable 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 benchmarks that consisted of homogeneous and heterogeneous circuits. The results show that, to achieve lowest cost in most cases, the number of customizable layers should be as small as possible; the area savings that could be obtained if more customizable layers are available does not usually translate into a cost savings. A few additional custom layers, however, can provide delay and power improvements.

The cost advantage provided by MPSAs can make modern technologies much more accessible, giving products access to further benefits (higher clock speed, lower power) of a modern process that are not available in older-technology CBICs.

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

One of the limitations in our CAD flow is the lack of an intelligent whitespace insertion algorithm. This inflates area and cost when there are too few routing layers. Developing an effective whitespace insertion algorithm is important, as it can potentially lower the cost of large designs on 2-layer MPSAs by 2$\times$, usually matching the low cost of 4-layer MPSAs.

There are some additional limitations in this work. In our delay and power estimates, we did not consider delay and power dissipation of the logic blocks or precise critical paths. In estimating CBIC cost, we assumed that all the masks are modified in a respin and did not consider the impact of ECO techniques ([33]). We also did not perform detailed routing or considered the impact of buffer insertion. Finally, we assume that there are dedicated power and clock networks for the logic blocks and we do not consider their area overhead. However, despite these limitations we believe our results are sufficiently accurate to draw important conclusions.

One direction for future work is to develop a whitespace insertion algorithm that inserts whitespace only in congested areas. One possible approach is to use a flow similar to Un/DoPack [49]. As we have shown, there is a significant performance gap that can be filled with such an algorithm. We also plan to investigate via-programmable structured ASICs (VPSAs). We have done some preliminary work in this regard [8], but we plan to extend it by looking into other possible architectures.

Acknowledgment

The authors would like to thank Dr. H. Schmit from eASIC Corporation and Mr. S. Khushal from QSi Consulting Group for their valuable comments.

References

[1] A. K.-K. Wong, *Resolution Enhancement Techniques in Optical Lithography*. Bellingham, WA: SPIE, Mar. 2001.
[2] “TSMC results fall in Q1, sees rebound” Apr. 2009 [Online]. Available: http://www.eetimes.com/showArticle.jhtml?articleID=217200925
[3] I. Kuon and J. Rose, “Measuring the gap between FPGAs and ASICs,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 26, no. 2, pp. 203–215, Feb. 2007.
[4] B. Zahiri, “Structured ASICs: Opportunities and challenges,” in *Proc. IEEE Int. Conf. Comput. Des.*, Oct. 2003, p. 404.
[5] eASIC Corp., Nextreme Structured ASIC [Online]. Available: http://www.easic.com
[6] Lightspeed Logic, ASIC offerings [Online]. Available: http://web.archive.org/web/20080111112237/http://www.lightspeed.com/products.html
[7] U. Ahmed, G. Lemieux, and S. Wilton, “Area, delay, power, and cost trends for metal-programmable structured ASICs (MPSAs),” in *Proc. Int. Conf. Field Program. Technol.*, Dec. 2009, pp. 278–284.
[8] U. Ahmed, G. Lemieux, and S. Wilton, “The impact of interconnect architecture on via-programmed structured ASICs (VPSAs),” in *Proc. Int. Conf. Field Program. Technol.*, Feb. 2010, pp. 263–272.
[9] Y. Ran and M. Marek-Sadowska, “The magic of a via-configurable regular fabric,” in *Proc. IEEE Int. Conf. Comput. Design*, 2004, pp. 338–343.
[10] Y. Ran and M. Marek-Sadowska, “An integrated design flow for a via-configurable gate array,” in *Proc. Int. Conf. Comput. Aided Design*, 2004, pp. 552–589.
[11] Y. Ran and M. Marek-Sadowska, “Via-configurable routing architectures and fast design mappability estimation for regular fabrics,” in *Proc. Int. Conf. Comput. Aided Design*, 2005, pp. 25–32.
[12] Y. Ran and M. Marek-Sadowska, “Via-configurable routing architectures and fast design mappability estimation for regular fabrics,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 9, pp. 998–1009, Sep. 2006.
[13] 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 *Proc. IEEE Design Autom. Conf.*, 2003, pp. 782–787.
[14] A. Koorapaty, V. Kheterpal, P. Gopalakrishnan, M. Fu, and L. Pileggi, “Exploring logic block granularity for regular fabrics,” in *Proc. IEEE Design Autom. Test Eur. Conf.*, 2004, vol. 1, pp. 468–473.
[15] V. Kheterpal, A. J. Strojwas, and L. Pileggi, “Routing architecture exploration for regular fabrics,” in *Proc. IEEE Design Autom. Conf.*, 2004, pp. 204–207.
[16] F. Veredas, M. Scheppler, B. Zhai, and H. Pfleiderer, “Regular routing architecture for a LUT-based MPGA,” in *Proc. Int. Symp. Very Large Scale Integr. (VLSI) Syst.*, 2006, pp. 257–262.
[17] F. Veredas, M. Scheppler, and H. Pfleiderer, “Automated conversion from a LUT-based FPGA to a LUT-based MPGA with fast turnaround time,” in *Proc. IEEE Design Autom. Test Eur. Conf.*, 2006, pp. 36–41.
[18] 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 Trans.*, vol. 91-C, no. 4, pp. 509–516, 2008.
[19] 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 *Proc. FPGA*, 2009, pp. 53–62.
[20] Altera HardCopy ASIC Series [Online]. Available: http://www.altera.com/products/devices/hardcopy-asics/about/hrd-index.html
[21] Tier Logic [Online]. Available: http://www.tierlogic.com
[22] Chip-X CX6200 Structured ASIC Datasheet [Online]. Available: http://www.chipx.com/images/stories/pdf/cx6200_usbphy_ds_0210d.pdf
[23] Faraday Structured ASIC Technology [Online]. Available: http://www.faraday-tech.com/html/products/structuredASIC.html
[24] Fujitsu AccelArray [Online]. Available: http://web.archive.org/web/20071031122818/www.fujitsu.com/global/services/microelectronics/product/asic/accelarray/index_2.html

[25] LSI Logic Rapidchip Xtreme2 [Online]. Available: http://web.archive.org/web/20061015051026/www.lsilogic.com/files/docs/techdocs/Rapidchip/rcxtreme2_ds.pdf
[26] T. Okamoto, T. Kimoto, and N. Maeda, "Design methodology and tools for NEC electronics' structured ASIC ISSP," in *Proc. ISPD*, 2004, pp. 90–96.
[27] ON Semiconductor [Online]. Available: http://www.onsemi.com/pub_link/Collateral/TND338-D.PDF
[28] ViASIC ViaMask and DuoMask [Online]. Available: http://www.viasic.com/products
[29] Virage Logic, ASAP Metal Programmable Cell Libraries [Online]. Available: http://www.viragelogic.com/upload/documents/product_broch_asap_logic_v10.pdf
[30] J. L. Hennessy and D. A. Patterson*, Computer Architecture: A Quantitative Approach*. San Mateo, CA: Morgan Kaufmann, 2006.
[31] C. Strapper, F. Armstrong, and K. Saji, "Integrated circuit yield statistics," *Proc. IEEE*, vol. 71, no. 4, pp. 453–470, Apr. 1983.
[32] The International Technology Roadmap for Semiconductors (ITRS), Yield Enhancement 2007, pp. 7–10.
[33] S. Golson, "The human ECO compiler," presented at the Proc. Synopsys Users Group Conf., 2004.
[34] Z. Or-Bach, Paradigm shift in ASIC technology: In standard metal, out standard cell eASIC White Paper, Sep. 2005.
[35] Frost & Sullivan Press Release, "The advent of next generation lithography technologies in advanced semiconductor processing," Aug. 27, 2007.
[36] V. Betz, J. Rose, and A. Marquardt*, Architecture and CAD for Deep-Submicron FPGAs*. Norwell, MA: Kluwer, 1999.
[37] 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 *Proc. ISPD*, 2005, pp. 224–226.
[38] A. Chakraborty, A. Kumar, and D. Pan, "Regplace: A high quality open-source placement framework for structured ASICs," in *Proc. IEEE Design Autom. Conf.*, Jul. 2009, pp. 442–447.
[39] J. A. Roy and I. L. Markov, "High-performance routing at the nanometer scale," in *Proc. IEEE Int. Conf. Comput. Aided Design*, 2007, pp. 496–502.
[40] S. Batterywala, N. Shenoy, W. Nicholls, and H. Zhou, "Track assignment: A desirable intermediate step between global routing and detailed routing," in *Proc. IEEE Int. Conf. Comput. Aided Design*, Nov. 2002, pp. 59–66.
[41] J. Rubinstein, P. P. , Jr., and M. A. Horowitz, "Signal delay in rc tree networks," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. CAD–2, no. 3, pp. 202–211, Jul. 1983.
[42] eASIC Placement Contest [Online]. Available: http://web.archive.org/web/20080723214221/http://www.easic.com/index.php?p=university
[43] J. A. Roy, D. A. Papa, A. N. Ng, and I. L. Markov, "Satisfying whitespace requirements in top-down placement," in *Proc. ISPD*, 2006, pp. 206–208.
[44] S. N. Adya, I. L. Markov, and P. G. Villarrubia, "On whitespace and stability in physical synthesis," *VLSI J. Intergr.*, vol. 39, no. 4, pp. 340–362, 2006.
[45] C. Li, M. Xie, C.-K. Koh, J. Cong, and P. H. Madden, "Routability-driven placement and white space allocation," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 26, no. 5, pp. 858–871, May 2007.
[46] A. Sharma, C. Ebeling, and S. Hauck, "Architecture-adaptive routability-driven placement for FPGAs," in *Proc. FPL*, 2005, pp. 427–432.
[47] M. Tom and G. Lemieux, "Logic block clustering of large designs for channel-width constrained FPGAs," in *Proc. IEEE Design Autom. Conf.*, 2005, pp. 726–731.
[48] 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 *Proc. IEEE Int. Conf. Comput. Aided Design*, 2006, pp. 680–687.
[49] D. Chiu, G. Lemieux, and S. Wilton, "Congestion-driven regional re-clustering for low-cost FPGAs," in *Proc. Int. Conf. Field Program. Technol.*, Dec. 2009, pp. 167–174.

Usman Ahmed received the B.Eng. degree in computer systems engineering from the National University of Sciences and Technology, Rawalpindi, Pakistan, in 2001, the M.A.Sc. degree in electrical and computer engineering from Ryerson University, Toronto, ON, Canada, in 2005, and is currently pursuing the Ph.D. degree in electrical and computer engineering at the University of British Columbia, Vancouver, BC, Canada.

In 2000, he joined a California-based startup company, Avaz Networks, where he worked until 2003. During his stay at Avaz, he worked on the design and verification of different entities for a carrier class VoIP processor. His current research interests include structured ASIC and FPGA architectures, CAD algorithms, VLSI, and SoC design.

Guy G. F. Lemieux (S'91–M'04–SM'08) received the B.A.Sc., M.A.Sc., and Ph.D. degrees from the University of Toronto, Toronto, ON, Canada.

In 2003, he joined the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC, Canada, where he is now an Associate Professor. He is co-author of the book *Design of Interconnection Networks for Programmable Logic* (Kluwer, 2004). His research interests include FPGA architectures, computer-aided design algorithms, VLSI and SoC circuit design, and parallel computing.

Dr. Lemieux was a recipient of the Best Paper Award at the 2004 IEEE International Conference on Field-Programmable Technology.

Steven J. E. Wilton (S'86–M'97–SM'03) received the M.A.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Toronto, Toronto, ON, Canada, in 1992 and 1997, respectively.

In 1997, he joined the Department of Electrical and Computer Engineering, University of British Columbia, Vancouver, BC, Canada, where he is now a Professor. From 2003 to 2004, he was a Visiting Professor in the Department of Computing, Imperial College, London, U.K, and at the Interuniversity MicroElectronics Center (IMEC), Leuven, Belgium. He is a cofounder of Veridae Systems, which supplies post-silicon validation architectures and tools. His research focuses on the architecture of FPGAs, and the CAD tools that target these devices.

Dr. Wilton was the Program Chair for the ACM International Symposium on Field-Programmable Gate Arrays in 2005 and the program co-chair for the International Conference on Field Programmable Logic and Applications. He received best paper awards at the International Conference on Field-Programmable Technology in 2003, 2005, and 2007, respectively, and at the International Conference on Field-Programmable Logic and Applications in 2001, 2004, 2007, and 2008, respectively. In 1998, he won the Douglas Colton Medal for Research Excellence for his research into FPGA memory architectures. He is currently an Associate Editor of the *ACM Transactions on Reconfigurable Technology and Systems*.