The Impact of Interconnect Architecture on Via-Programmed Structured ASICs (VPSAs)

Usman Ahmed
uahmed@ece.ubc.ca

Guy G. F. Lemieux
lemieux@ece.ubc.ca

Steven J. E. Wilton
stevew@ece.ubc.ca

Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, BC, Canada

ABSTRACT

In this paper, we evaluate the performance of an FPGA-like interconnect fabric for structured ASICs which is based upon fixed metal and programmable vias. We call this type of device a *via-programmed structured ASIC* or VPSA. We look at two different types of VPSA routing fabrics: one uses *jumper wiring* and the other uses *crossover* wiring. The performance of these fabrics is compared against an ASIC-like interconnect fabric, otherwise known as a metal-programmed structured ASIC or MPSA, which can be configured by customizing metal and via layers. We study the impact of these routing fabrics on cost, area, power and delay metrics. The results for different fabrics span a wide range, suggesting the routing architecture plays a very important role in their overall performance and it should be thoroughly researched.

Categories and Subject Descriptors

B.7.1 [**Integrated Circuits**]: Types and Design Styles—*VLSI*

General Terms

Design, Economics, Performance

Keywords

Structured ASICs, Via Programmable Fabric

1. INTRODUCTION

As process technology is scaling to finer geometries, the cost to design an integrated circuit (IC) is becoming extremely high. As a result, most designs are still being implemented in 130nm or older process technologies; advanced process technologies, such as 90nm and below, only account for 49% of TSMC's revenue [1]. Field-programmable gate arrays (FPGAs) offer one solution to this problem, but it is not possible to use them in all situations. In particular, applications from the emerging portable/hand-held device market are not suitable for implementation with current SRAM-based FPGAs [7]. Structured ASICs offer one solution to these problems.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
FPGA'10, February 21–23, 2010, Monterey, California, USA.
Copyright 2010 ACM 978-1-60558-911-4/10/02 ...$10.00.

A structured ASIC is a generic IC that is partially fabricated. It can be "programmed" to implement any digital circuit by adding one or more metal and/or via layers [17]. This partial fabrication of the device reduces the cost and turnaround time compared to a traditional cell-based IC (CBIC). In contrast to FPGAs, structured ASICs do not use active, SRAM-based programmable switches that are a major source of power and speed degradation [10]. As a consequence, they consume less power and are faster and more dense than FPGAs. For these reasons, we believe the structured ASICs are an increasingly important design methodology, especially for platform-based designs and the portable/battery operated device market. This advantage will continue to grow as we scale to fine processes such as 32nm and below.

Structured ASICs are not a new technology. They were introduced several years ago, however, they have not yet achieved the popularity that was predicted. There are several possible reasons for that such as unfamiliar technology, immature CAD, and claimed advantages that have not yet been concretely demonstrated. However, the problems that led to the emergence of structured ASICs have not been solved and have in fact become more complex. We believe that as technology continues to advance, the advantages of structured ASICs will become even more compelling. When that happens, we will need new architectures, CAD tools and design flows. In this paper, we take a step in this direction and investigate the interconnect architecture for structured ASICs that can be customized only by via layers. Since structured ASICs are so similar to FPGAs in their internal structure (based on a grid of logic blocks connected using "configurable" connectors), we believe the FPGA community is well prepared to address these issues.

The cost, performance, turnaround time and power of structured ASICs is dictated by the number of metal/via layers that can be changed to customize a product. Structured ASICs from companies, such as LSI Logic, NEC, Fujitsu, Lightspeed Logic, Faraday, Chip-X, eASIC, and Altera, vary largely in the way they are customized. As an example, eASIC's device can be configured by only a single via layer [2], whereas Altera's device can be customized by two metal and two via layers [3]. In contrast, although no longer available, the Lightspeed Logic structured ASIC offered up to six metal and six via layers of configurability. More customizable layers should lead to smaller area, lower delay and lower power, but higher mask costs. How many configurable layers are needed to achieve best quality of results? What is the sensitivity? This research attempts to address these questions.

There have also been some academic efforts to study structured ASICs, however these have focused mainly on point solutions. The works in [12] [8] [9] [13] [14], have all studied structured ASICs where the number of configurable layers was fixed. Our goal on the other hand is to investigate structured ASICs as a function of

different amounts of configurability because this is the most important factor that affects their cost and turnaround time. As part of this effort, we have looked at the trade-offs involved in structured ASICs that are configured by metal and via layers (MPSAs) [5]. In [5], we developed a cost model to estimate the manufacturing cost of MPSAs, created a CAD flow, and then studied the area, delay, power and cost trends for a range of MPSA architectures and layout assumptions. In this paper, we look at the trends for VPSAs, where all the metal layers are fixed and the customization can only be performed by one or more via layers. We investigate how the fixed metal affects the overall performance of VPSAs, and how it changes with different types of routing fabrics. Specifically, our contributions in this paper are:

- extending the cost model and the CAD framework of [5] to account for VPSAs,
- looking at two different types of via-programmable routing fabrics (similar to those in [13]), and studying their effect on cost, area, delay, and power of VPSAs as a function of number of customizable layers, and
- comparing the performance of VPSAs against MPSAs.

2. COST MODEL

In most FPGA and ASIC research, it is traditional to estimate die area as a proxy to the cost of a device. For structured ASICs, it is not possible to use area alone, as cost also depends upon how many mask layers are customized in creating a specific device. There is no agreement on how many of these custom layers are needed, so this is a new architectural parameter that needs to be explored. We wish to vary the number of these programmable layers, so we need to explicitly account for their cost.

In [5], we developed a cost model to estimate die cost of MPSAs as a function of its core area number of customizable routing layers. Here we extend the cost model to incorporate VPSAs. There are two main enhancements. First, the masks for the metal layers used for routing are now fixed and their costs are amortized across the total production volume of the generic device. These costs were previously amortized across the customer's volume. Second, we also account for the prototyping cost. The cost model with these changes is described below.

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

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

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. We assume that C_{pkg} and C_{test} are constants, 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 total number of routing layer masks, N_{rm}, in a VPSA with N_{vl} custom via layers can be broken down into three components:

$$N_{rm} = N_{fm_m} + N_{fm_v} + N_{cm}$$

where N_{fm_m} and N_{fm_v} are number of masks needed for fixed metal and fixed via layers, and N_{cm} is the number of custom masks to program the VPSA. In a VPSA, the first two terms refer to masks that will be used across all customers of a device, while the last term is specific to an individual customer. If N_m and N_v denote the number of masks needed for a single metal and single via layers respectively, then N_{fm_m}, N_{fm_v} and N_{cm} are defined as:

$$\begin{aligned} N_{fm_m} &= (N_{vl}+1)N_m \\ N_{fm_v} &= N_v \\ N_{cm} &= N_{vl} \times N_v \end{aligned}$$

Thus, a VPSA with a single custom via layer would use two metal and two via layers for routing. The two metal layers and one via layer is fixed; the fixed via layer connects the routing fabric to the logic block pins.

The base cost can now be expressed as:

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

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 etc.), C_{sm_l} is the cost for a single lower-level mask (e.g., poly mask, metal-1 mask etc.), C_{sm_u} is the cost for a single upper-level mask (e.g., metal-4 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.

The customization cost can be expressed in a similar way as:

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

where V_c is the volume per customer.

Due to the complexity of large hardware designs, it is very difficult to get everything right the first time. Thus, 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:

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

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 multi-project wafer, where each customer uses less than a full wafer. This would reduce the wafer cost component of the prototype to nearly zero. In our previous work [5], we had implicitly set this wafer cost to zero, but the difference this has on results is very small.

By substituting the values in Equation 1 and rearranging the terms, C_{die} can be written as:

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

where K_0, K_1, K_2, and K_3 are constants that depend on the volume requirements and various foundry costs, but are fixed for a given structured ASIC product. After substituting values for N_{fm_m} and N_{fm_v} and simplifying, these constants become:

$$K_0 = C_{wpm}(N_{fm_l} + N_{fm_u} + N_m + N_v) + C_{sw}$$

$$K_1 = C_{wpm}(N_m + N_v)$$

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

$$K_3 = \frac{C_{sm_l} N_{fm_l} + C_{sm_u}(N_m + N_v + N_{fm_u})}{V_{tot}} + (C_{wpm}\,(N_{fm_l} + N_m + N_v + N_{fm_u}) + C_{sw})\left(\frac{N_s - 1}{V_c}\right) + \frac{C_{fs_1}}{V_{tot}} + N_s\left(\frac{C_{fs_2}}{V_c}\right) + C_{pkg} + C_{test}$$

Table 1 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.

We calculated the typical values for K_0, K_1, K_2, and K_3 using the parameter values of Table 1. These values are shown in Table 2 along with the corresponding values for MPSA. The key difference in the VPSA cost model is the change to K_2, which makes it cheaper to add custom layers. Previously, K_2 was one term amortized across the customer's volume alone; here, the cost of the fixed metal routing masks is separately amortized across the total device volume, which produces some savings. However, this comes at a slight increase in K_0 and K_3.

Finally, when computing cost per die, the number of good dies per wafer must be considered. This depends upon two factors, area and yield. Area determines the number of die that can be stamped out from a wafer, while yield determines the fraction of die that are functional. In this paper, we use the same yield model as our previous work [5].

Table 1: 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 total masks [11]; we assume 45nm also requires 34 masks[a] (2) Device fabricated up to metal-2, and subsequent layers require single mask
C_{sm_l}	\$107k	*Single mask cost for lower layers* (1) 45nm mask set costs \$2.5M [4] (2) Cost of lower level masks is 3x that of upper level masks
C_{sm_u}	\$36k	*Single mask cost for upper layers* [11]
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
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)*
N_m, N_v	1	*Number of masks per metal/via layer* One mask each for metal and via
$\frac{C_{fs_1}}{V_{tot}}$, $\frac{C_{fs_2}}{V_c}$	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* 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_{vl}

[a] We have also modeled a process that has 18 more fixed masks (i.e., 52 total masks) by increasing N_{fm_l} to 36. The results are shown in section 4.3.2.

Table 2: Typical Values for the Cost Model Constants

Type	K_0	K_1	K_2	K_3
VPSA	\$4840	\$440	\$0.7424	\$1.0834
MPSA	\$4400	\$440	\$1.4444	\$1.0430

3. FRAMEWORK

We have set up a framework to study the trade-offs associated with MPSAs in [5]. We have extended it to incorporate VPSAs so that we may compute various metrics such as die-area, delay, power and manufacturing cost for different VPSA architectures. In this section we describe how a logic block is modeled in our framework, our CAD flow, and the metrics that we compute.

3.1 Architecture Model

To model a particular logic block architecture, the goal is to model it without worrying about the low-level, layout-related details. 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 routing tracks. Figure 1 illustrates the modeling process for a simple 2-input, 1-output logic block.

3.2 CAD Flow

Our VPSA CAD flow is shown in Figure 2. The flow starts with reading an initially technology-mapped circuit. We then use TVPack to pack this into a multi-input, multi-output logic block. The next step is to initialize the placement step by reading in the physical size (height and width) of a logic block, the location of logic block inputs and outputs, and the location of I/O pads. 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 $\lceil\sqrt{N}\rceil \times \lceil\sqrt{N}\rceil$). Following this, we perform placement, route the placed blocks 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 following subsections describe these of the stages in more detail.

3.2.1 Technology Mapping

Generally, it is difficult to map a circuit directly into multi-output logic blocks. Hence, we use TVPack to technology map our input netlist of single-output primitives into multi-input, multi-output logic blocks. To avoid a strong dependence upon the precise in-

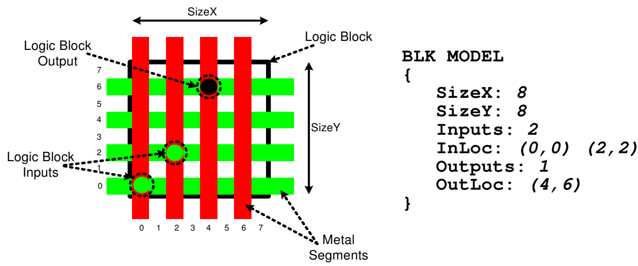


Figure 1: Modeling an Architecture

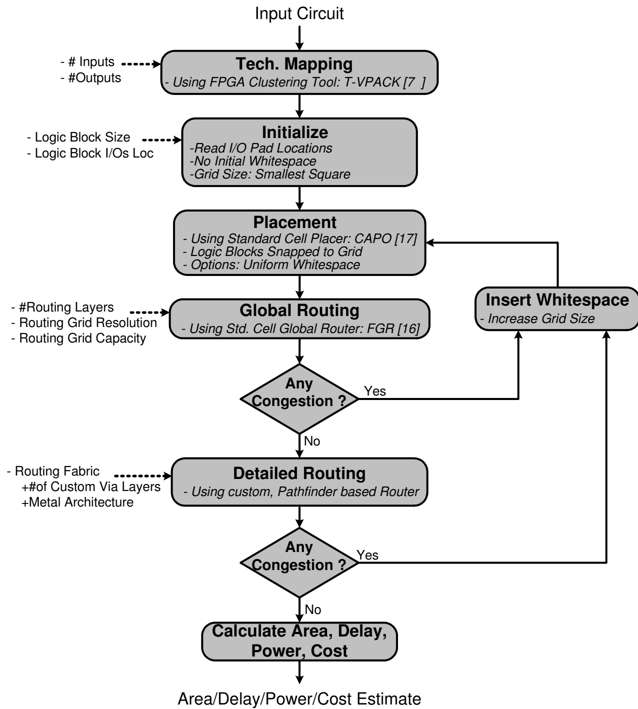


Figure 2: VPSA CAD Flow

ternal logic block architecture, we only impose input and output constraints; we do not constrain the number of gates or functions that will fit inside except for a maximum of one flip-flop per output pin. In this paper, all of our circuits start as 2-input LUTs and flip-flops, but it is possible to start with other primitives (such as 4-input LUTs). Due to the use of TVPack, our technology mapping results may not be well-packed compared to a dedicated technology mapper. It is also possible that too much logic gets packed into a logic block. However, the mapped result is always a valid netlist for which a valid logic block architecture can be designed. Such a clustered netlist also has many of the properties (such as fan-in and fan-out distributions, Rent parameter etc.) of a real technology mapped circuit.

3.2.2 Placement and Global Routing

We use an open source, standard cell placer known as CAPO to perform placement [16]. The use of an ASIC placer, instead of VPR, is preferred for two reasons. The first is placement runtime, and the second is the ability to insert whitespace. CAPO has different options for whitespace insertion; we are using the uniform whitespace distribution scheme.

To perform global routing, we use the FGR [15] global router. 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). If there is congestion after global routing, we increase the placement grid size, thus creating room for whitespace, and then re-place the circuit. The uniform whitespace allocation scheme distributes this whitespace across the die and helps to eliminate the congestion. Some circuits require a large amount of whitespace, therefore, to speed up the flow we use a binary search to find the minimum routable size.

3.2.3 Detailed Routing

To accurately measure the wirelength and delay of VPSAs, we perform detailed routing 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, 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. It is therefore infeasible to perform routing using a single routing resource graph for the whole architecture, as is done in VPR [6].

We have developed our own router to perform detailed routing. The routing algorithm is similar to the one being 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; 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. This usually 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.

3.2.4 Other CAD Limitations

The placement and routing steps are not timing-driven. Also, routing does not perform buffer insertion or use specialized heuristics for high-fanout nets, e.g. a clock router. This can increase delays observed for large fanout or long nets. Fortunately, the number of such nets in our benchmarks is typically very small; for the largest benchmark circuit, clma, 98% of nets have a fanout of 10 or less and 81% of nets have a total length that is less than or equal to 15 logic blocks.

3.3 Metrics

The metrics we use to compare different configurability choices include area, delay, power and manufacturing cost.

The area of the core is calculated by multiplying the logic block area with the size of the placement grid.

We use average net delay as our delay metric. We determine the delay of each net by taking the average of the delay values of each of its sinks. These delay values are then averaged over all nets to calculate the average net delay. We use the average net delay, as

opposed to critical-path delay, for two reasons. First, we are interested in delay variation with different configurability choices rather than actual performance of the circuit. The number of routing layers only affects 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 (e.g., presence/location of any flip-flops). Also, since our CAD flow is not timing driven, it does not add any value to calculate critical-path delay. The delay values are estimated using the Elmore delay model.

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 neglect any change to glitching activity since it is minor). 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 2 to estimate the manufacturing cost of the die.

4. RESULTS

In this section, we explain our experimental methodology and describe the different routing fabrics that we use in our experiments. This is followed by an architecture study of VPSAs. We evaluate power, delay, and area trends for three different routing fabrics and also compare these against MPSAs. Finally, we analyze the die-cost of VPSAs and examine the sensitivity of this cost on assumptions made for various parameters in Table 1.

4.1 Experimental Methodology

In our experiments, we use nineteen of the largest MCNC benchmark circuits[1] that are commonly used in FPGA research [6].

4.1.1 Logic Blocks

We do not consider any particular architecture for the logic blocks. Instead we abstract the contents of a logic block by representing its input and output pins and area. We then perform clustering to produce an interconnect netlist that approximates a real technology-mapped netlist as described in section 3.2.1.

Because we avoid real technology mapping, we must avoid comparing the results of two different logic blocks (I/O counts) directly. Hence, we do not draw any conclusions about which logic block is better in this paper.

Our experimental methodology also requires an estimate of the layout area (height and width) and pin locations of each logic block. 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 within that range, sweep through three equally spaced points. 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 use standard cell libraries to estimate the area of a “Sparse” implementation. For a two-input, one-output “Sparse” logic block, we average the areas of different basic gates such as two-input NAND and NOR, 2:1 MUX, etc. To determine the “Dense” logic block area, we assume the lower bound on area will be limited by the number of logic block i/o pins. If the block has n pins, we assume that the height (and width) of the block is such that $2n$ wires[2] can pass over it plus an additional gap of 2 wires between tiles (for jumper or crossover vias). The “Dense” area value for each of the different types (i/o counts) of logic blocks is determined using this scheme. The “Medium” and “Sparse” area values for logic blocks with more than two inputs is determined such that the ratio between the three sizes is the same as the corresponding ratio for two-input logic blocks. The different logic block *types* (i/o counts) and the corresponding layout area values (in units of half metal pitches) used in our experiments are shown in Table 3.

Table 3: Logic Blocks used in Experiments

Type		**Block Layout Area(Width×Height)**		
IN	***OUT***	***Dense***	***Medium***	***Sparse***
2	1	16x16	20x20	24x24
4	2	28x28	34x34	42x42
6	3	40x40	50x50	62x62
8	4	52x52	64x64	80x80
10	5	64x64	80x80	100x100
12	6	76x76	94x94	116x116
14	7	88x88	110x110	136x136
16	8	100x100	124x124	154x154

4.1.2 Routing Fabrics

We use two different types of routing fabrics in our experiments. These connect between tiles using crossovers or jumpers, respectively. These routing fabrics are illustrated in Figure 3, where we show the metal architecture for two adjacent layers for each fabric. The fabrics have been specified over a tile of 2x2 logic blocks. These fabrics are similar to those described in [14], except that we use single vias (not double vias) to make connections between adjacent tiles.[3] Also, we allow long wires that can span multiple tiles in the same layer.

In our experiments, we use the routing fabric of Figure 3a and two versions of the routing fabric of Figure 3b. We define them as “Crossover”, “Jumper20”, and “Jumper40” routing fabrics, respectively. These are described in Table 4.

4.2 VPSA and MPSA Power, Delay

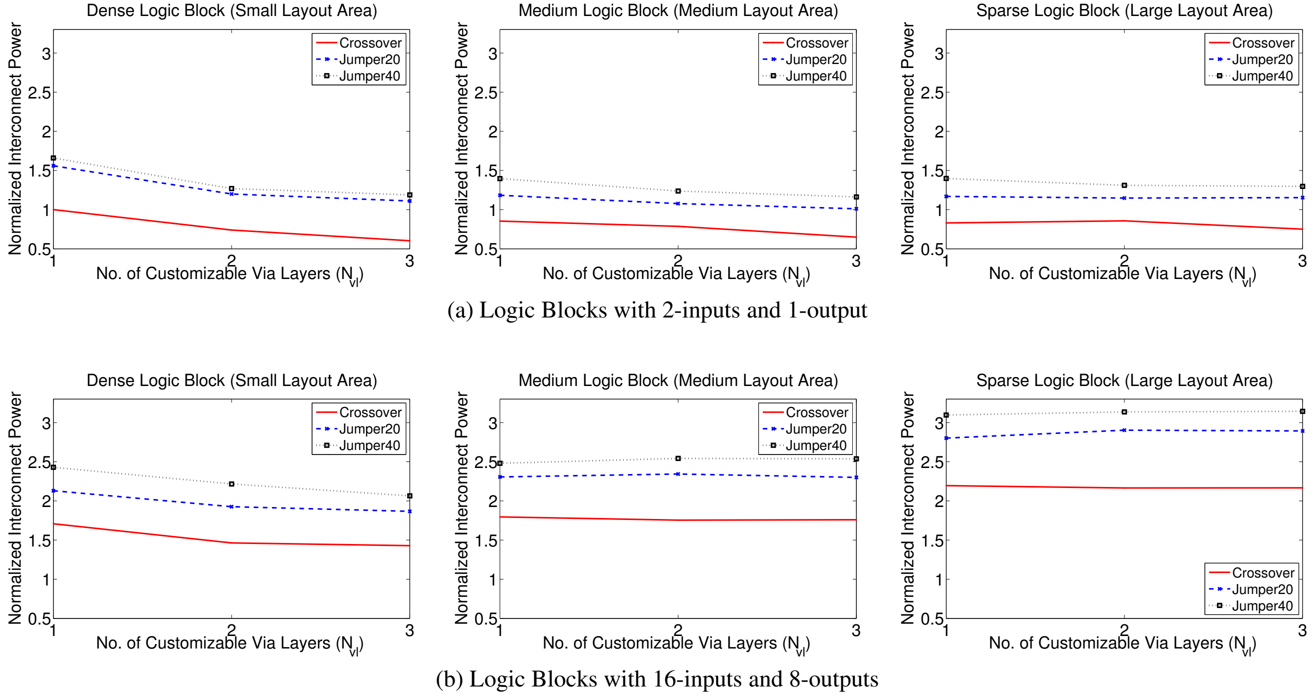
4.2.1 Power Results

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 three different routing fabrics are shown in Figure 4. The horizontal axis in each graph shows the number of via layers that can be customized to perform routing. An increase in this number represents that additional metal layers with fixed metal segments are also available for routing. For example, one customizable via layer can connect two metal layers, two customizable via layers can connect three metal layers, and so on. The vertical axis in each of the plots shows interconnect power that we estimate from total interconnect capacitance. The values in

[1] The global router (FGR) has a limitation to route only up to 1000-pin nets. One of the circuits, s38584.1, contains a net with more than 3000 pins. Instead of modifying the benchmark or the router, we chose to exclude the benchmark from our experiments. For such high-fanout nets, a different type of a router (e.g., a clock router) is likely to be used.

[2] We assume that a wire spans at least one logic block, hence there can be at most one pin in each track.

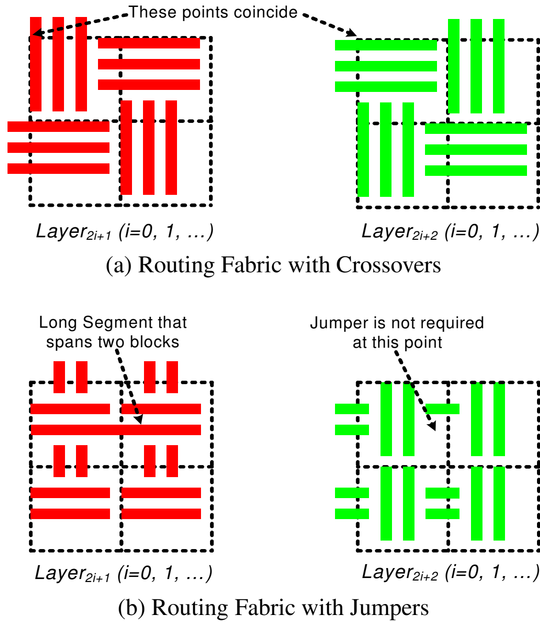
[3] Although double vias have lower resistance, they require twice the area. If used at interior points within a tile, they reduce available metal capacity (pitch) by half.



(a) Logic Blocks with 2-inputs and 1-output

(b) Logic Blocks with 16-inputs and 8-outputs

Figure 4: VPSA Power Trends (Normalized to Power Values of 2-input, 1-ouput Dense Logic Blocks with *Crossover* Routing Fabric)



(a) Routing Fabric with Crossovers

(b) Routing Fabric with Jumpers

Figure 3: Metal Architecture of Routing Fabrics (From [14])

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 4a and 4b show the plots corresponding to the first and last rows of Table 3; the trends for other logic block types are similar and are not shown because of space constraints.

There are two main observations. First, there is a significant range in interconnect power consumption of the different routing fabrics. For a two-input, one-output logic block (Figure 4a),

Table 4: Routing Fabrics used in Experiments

Routing Fabric	***Description***
Crossover	The routing fabric shown in Figure 3a. Each metal segment spans one logic block and the size of the logic block determines how many segments pass over it.
Jumper20	The routing fabric shown in Figure 3b. Two types of metal segments are used; one that spans a single logic block and the other that spans 4 logic blocks. The routing fabric is specified over a tile of 4x4 logic blocks. The long segments are staggered across the tile. 20% of the segments passing over a logic block are long.
Jumper40	This is similar to *Jumper20* routing fabric, except that 40% of the metal segments are long.

Jumper40 dissipates 66% more power than Crossover; Jumper40 also dissipates 6%-20% more power than Jumper20. Similarly, for the other logic block type (Figure 4b), Jumper40 dissipates 42% and 14% more power than Crossover and Jumper20, respectively. These trends do not change with the number of customizable via layers. Across different logic block layout densities (each individual plot in Figures 4a or 4b), the relative power dissipation between the three fabrics is mostly similar. It is also interesting to see that a small architectural change, varying the number of long segments from 20% to 40%, can affect the power consumption by 6% to 20%.

Second, for a particular layout density, the power dissipation improves as more routing layers are available and this improvement diminishes as the layout area of the logic block increases. This can easily be seen from the interconnect power of Crossover across the three plots in Figure 4a. For "Dense" logic block (left plot), the power consumption with three customizable via layers is 40%

lower than the power consumption with a single customizable via layer. For "Sparse" logic block (right plot), this variation is only 2%. A similar conclusion can be drawn from Figure 4b. 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.

Another interpretation of Figure 4a is that results are sensitive to whitespace allocation. In particular, the "Dense" logic block with one via layer has higher power than the "Sparse" logic block for both Jumper20 and Jumper40 routing fabrics. In other words, whitespace inserted by the placer is not as well-placed as the uniform whitespace obtained from using a sparse layout.

4.2.2 Delay Results

The delay results are shown in Figure 5. These plots are similar to Figure 4, except that the vertical axis now shows interconnect delay for the three fabrics. As in the case of power, the plots are normalized to the interconnect delay for the Crossover routing fabric with two-input, one-output "Dense" logic blocks and N_{vl} = 1. As mentioned in section 3, average net delay is used to estimate interconnect delay. There are three main observations from the delay plots. First, like power, delay has significant range across different routing fabrics. For the logic block with two-inputs and one-output (Figure 5a), Jumper20 has about 100% more delay than Crossover for "Dense" and "Medium" logic blocks. The difference for the "Sparse" logic block is 75%. Similarly, Jumper20 is 29%, 22%, and 7% slower than Jumper40 for "Dense", "Medium", and "Sparse" logic blocks respectively. For the other logic block type (Figure 5b), there is also 33% to 39% difference between Crossover and Jumper20/Jumper40. The delay of Jumper20 and Jumper40 routing fabrics is mostly similar, varying at most by only 5%. 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 delay improves as more routing layers are available. This is also because more routing layers reduce congestion, allowing the circuits to route with less whitespace. This reduces the wirelength for each net and improves delay.

A third observation is regarding the relative performance of Jumper20 and Jumper40 routing fabrics: Jumper20 has fewer long segments than Jumper40, making it better in terms of power. However, for delay the situation is reversed. This shows an interesting trade-off; having more long segments in a fabric can improve delay but can make power worse.

4.2.3 Comparison to MPSAs

In this section, we compare the performance of VPSAs and MPSAs to study the penalty of fixing all the metal layers when using only via layers for customization. To study this effect, we show the ratio of power and delay of VPSAs relative to MPSAs; the trends for area are similar and we do not show the area results. We only show the results for "Dense" and "Sparse" logic blocks with two inputs and one output.

The interconnect power comparison is shown in Figure 6. It can be seen that using only via layers for customization increases power dissipation. As the layout area of the logic blocks increase, the gap between MPSAs and VPSAs is reduced. In our experiments, for different logic block types with different routing fabrics, this increase can vary from 1.6x to 5.5x.

The comparison of VPSA and MPSA interconnect delay is shown in Figure 7. As in the case of power, the VPSA delay is worse relative to MPSA and it improves as the layout area increases. However, it is important to observe the magnitude by which the delay gets worse in VPSAs relative to MPSAs. It was surprising to see that maximum increase in the delay of VPSAs ranges from 20x, for "Dense" logic block, to about 5x, for "Sparse" logic block. To investigate this, we looked at the total wirelength and total number of vias for each of the fabrics relative to MPSA. These plots are shown in Figure 8. It can be seen that with a single customizable via layer, the increase in wirelength is only 3x to 5x, whereas the number of vias has a dramatic increase of 20x to more than 30x. This causes the delay to increase significantly. The reason for this behavior is that in VPSAs, whenever a wire has to extend in any direction, it needs to go through a via; depending on the architecture of the routing fabric, it may have to go through more than one via in order to extend (e.g., in case of Jumper20 and Jumper40 routing fabrics) by just one more segment.

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

4.3 VPSA and MPSA Area, Cost

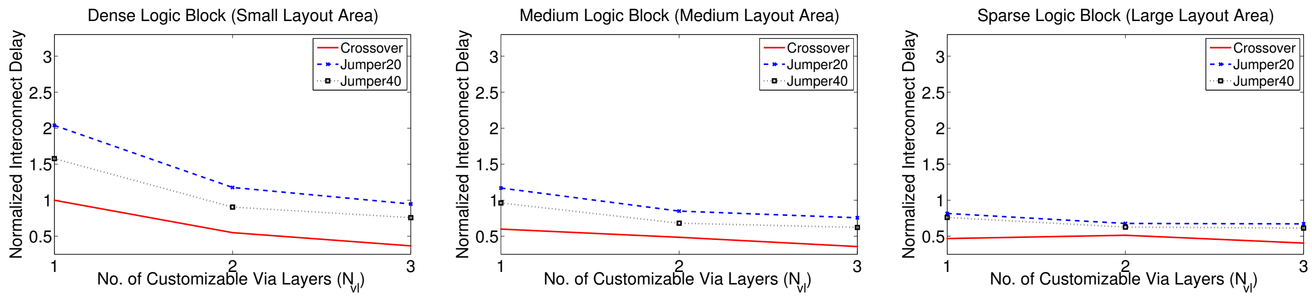
4.3.1 Area Results

The plots for core area are shown in Figure 9. These plots are similar to power and delay plots and the observations are similar too. It can be seen from Figure 9 that, like power and delay, there is a significant range of core area between the different routing fabrics. The core areas with Crossover are considerably lower than that with Jumper20 or Jumper40 fabrics. However, there is not much difference between the core areas of Jumper20 and Jumper40. Also, area decreases as more routing layers are available. Generally, the area with $N_{vl} = 1$ is larger than the area with $N_{vl} = 2, 3$, especially for the "Dense" logic block, because usually there is a lot of congestion when few routing layers are available. 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.

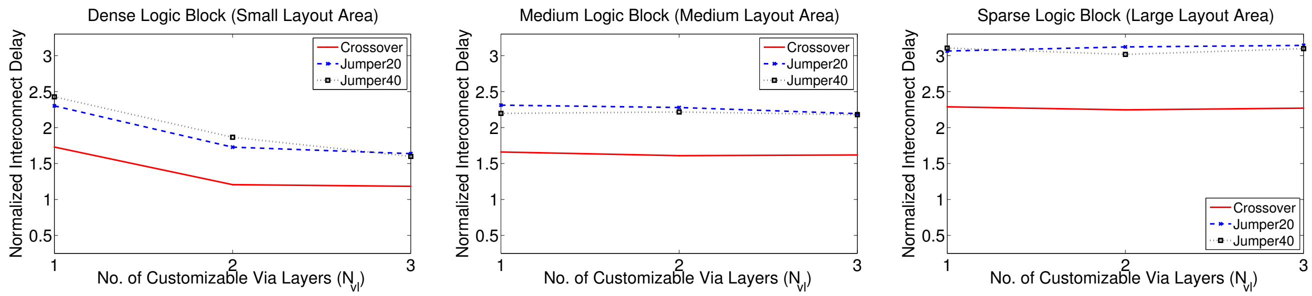
4.3.2 Cost Results

Finally, we apply the cost model described in section 2 on the area values to estimate the die-cost. However, because the MCNC benchmarks are very small, we need to scale the core area to a reasonable value. For 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 custom metal layers (this corresponds to a VPSA with a single customizable via layer) is $10mm^2$. The resulting cost plots are shown in Figure 10. Instead of showing relative values, we show absolute die-cost values and also include a cost curve for MPSAs. The important observations are described below.

The routing fabric architecture affects die-cost. Different VPSA routing fabrics have different die-costs relative to each other and also relative to MPSA. In some cases, the MPSA cost is better than VPSA cost whereas in other cases a VPSA is cheaper than MPSA, despite the fact that VPSAs are always larger than MPSAs. Generally, for densely laid out logic blocks, MPSA is cheaper when fewer routing layers are available. With more routing layers, VPSAs become cheaper. This is easily seen from Figures 10a and 10b for "Dense" logic block. As the layout area of logic blocks increases

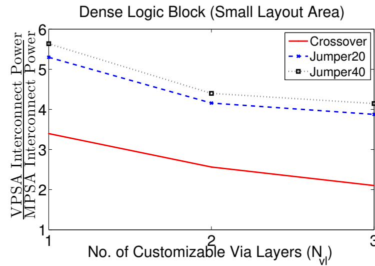


(a) Logic Blocks with 2-inputs and 1-output

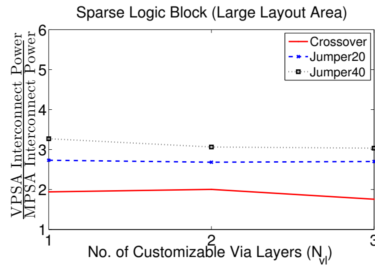


(b) Logic Blocks with 16-inputs and 8-outputs

Figure 5: VPSA Delay Trends (Normalized to Delay Values of 2-input, 1-ouput Dense Logic Blocks with *Crossover* Routing Fabric)

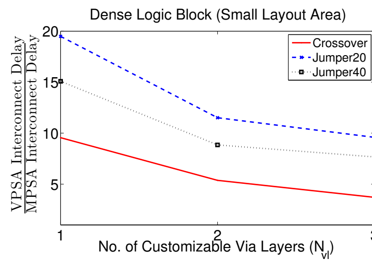


(a)

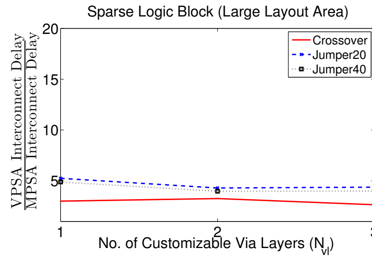


(b)

Figure 6: VPSA and MPSA Power Comparison (2-input, 1-output Logic Block)

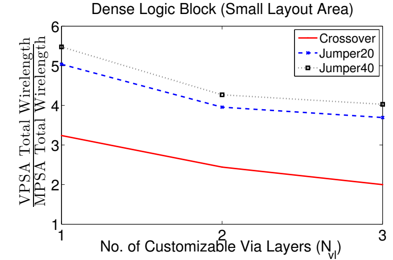


(a)

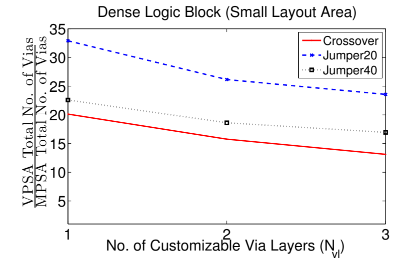


(b)

Figure 7: VPSA and MPSA Delay Comparison (2-input, 1-output Logic Block)



(a) Total Wirelength



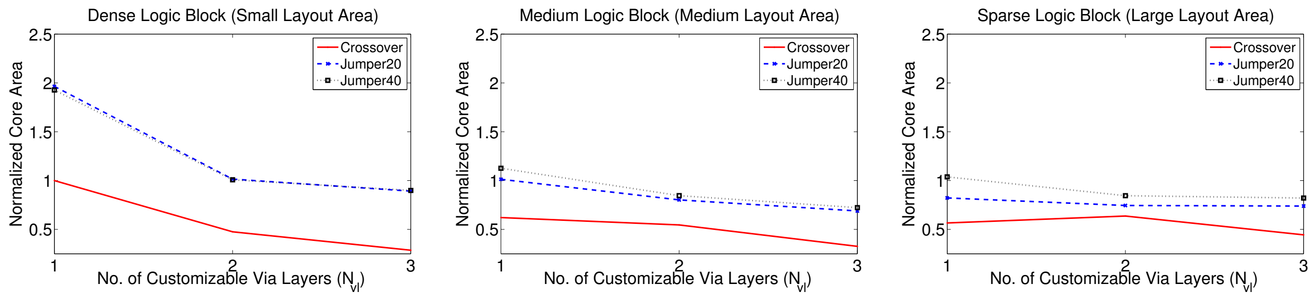
(b) Total Number of Vias

Figure 8: VPSA Delay: Effect of wirelength and number of vias

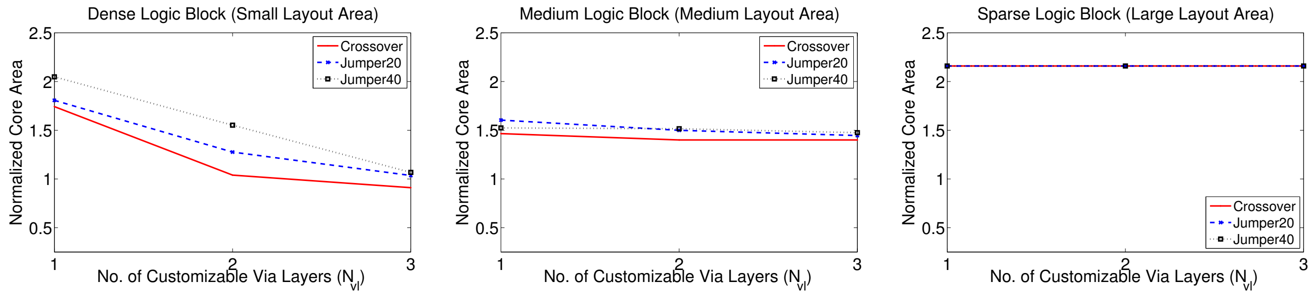
(layouts become sparse), VPSAs become more cost effective even with fewer routing layers. It can also be seen that sparsely laid out logic blocks may lead to a more cost effective die than densely laid out logic blocks; this may be caused by inferior whitespace allocation performed during placement.

The plots in Figure 10 also show that, with additional routing layers, the die cost for VPSAs can reduce. In contrast, the cost of MPSAs always increases with more routing layers [5]. The reason is that, with new process technologies, the cost of the maskset is the dominant factor that determines die cost. In MPSAs, as the number of routing layers increases, the number of masks that need to be customised also increases, nullifying the effect of any area savings.

Finally, we investigated the sensitivity of the cost model to various parameters of Table 1. We looked at a range of values for different parameters and studied the die costs. The cost plots for some of the interesting results are shown in Figure 11. Because of space

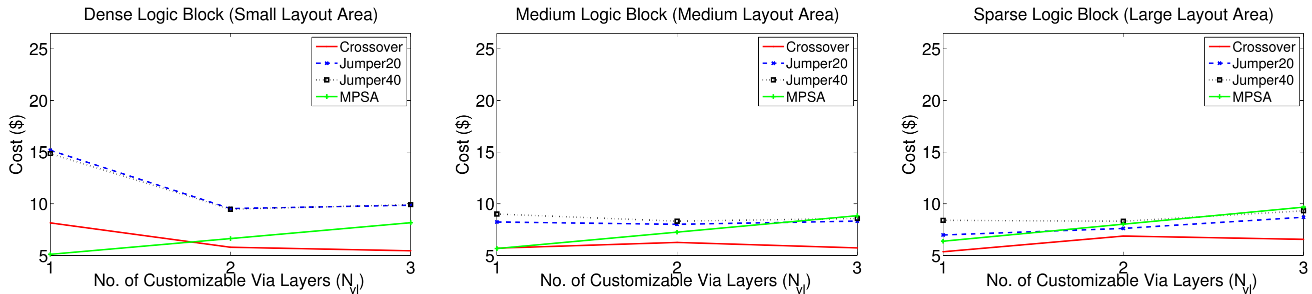


(a) Logic Blocks with 2-inputs and 1-output

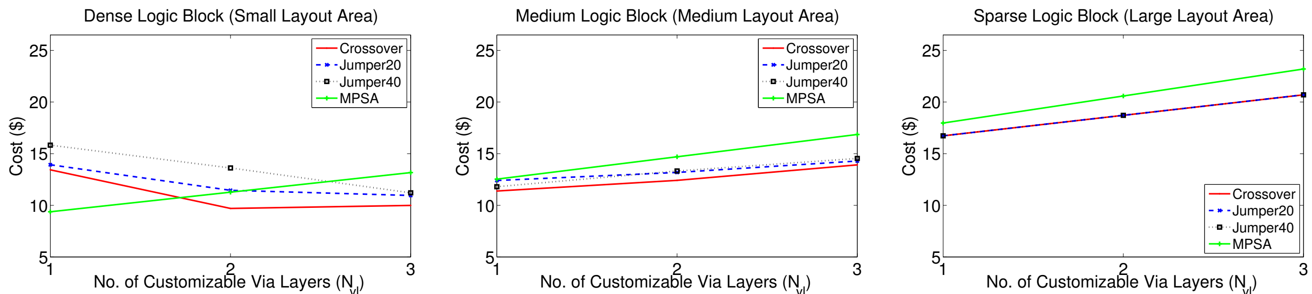


(b) Logic Blocks with 16-inputs and 8-outputs

Figure 9: VPSA Area Trends (Normalized to Area Values of 2-input, 1-ouput Dense Logic Blocks with *Crossover* Routing Fabric)



(a) Logic Blocks with 2-inputs and 1-output



(b) Logic Blocks with 16-inputs and 8-outputs

Figure 10: VPSA Cost Trends

constraints, we only show the plots for "Dense" logic block size with 2-inputs and 1-output. Figure 11a shows that, with more fixed masks, the increase in the cost of VPSAs is more rapid than MPSAs. This is because of the large area of VPSAs relative to MPSAs; the increase in the cost to process a wafer is divided among fewer dies, resulting in a larger increase in VPSA cost. Figures 11b and 11c show that MPSAs increase in cost more quickly when mask costs go up or volumes go down. Finally, the Crossover routing fabric is consistently better than Jumper20 and Jumper40 fabrics under any changes to the cost model parameters.

5. LIMITATIONS

The work presented in this paper is early work and there are some limitations in it. First, in our delay and power estimates, we did not consider delay and power dissipation of the logic blocks or precise

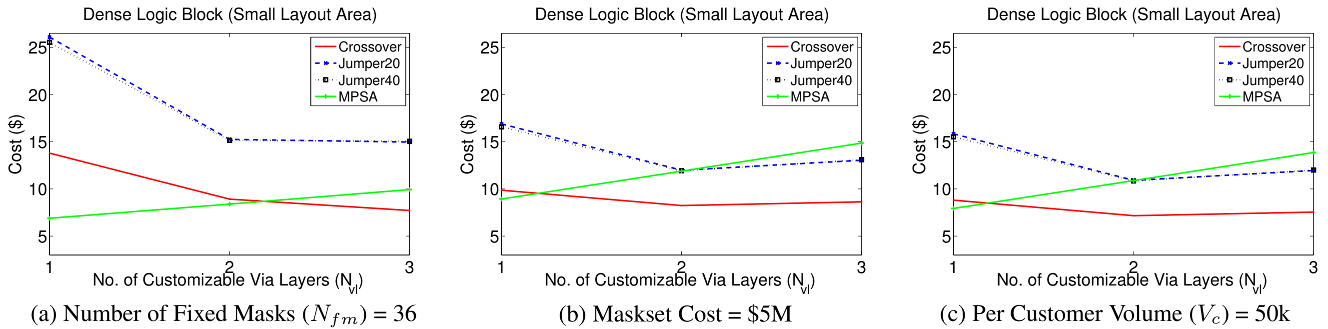


Figure 11: Sensitivity of Cost Model to Various Parameters

critical paths. Second, 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. This can cause the core area to escalate, especially with small block sizes and fewer routing layers. Third, we assume that there are dedicated power and clock networks for the logic blocks and we do not consider their overhead. Fourth, the logic blocks could be configured in different ways such as vias or SRAM cells. The use of vias in the lower layers can increase the die cost of VPSAs. However, in this paper we did not study this effect. Finally, due to the increased via count and wirelength of VPSAs over MPSAs, a buffer-insertion strategy may need to be followed by the CAD. However, despite these limitations we believe our results present an important first look at trends in structured ASIC architecture.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have looked at the impact of the VPSA routing fabric on power, delay, area, and die cost. We investigated two different routing fabrics, one that uses crossover wiring and the other that uses jumper wiring. It was observed that the performance of VPSAs gets better with more customizable via layers, however, most of the improvement occurs in going from one to two layers. The trends across different fabrics are similar, but there is a significant difference between the performance of these fabrics. This suggests that the interconnect architecture plays a very important role in the overall quality of results in VPSAs and it should be thoroughly researched.

We have also compared the performance of VPSAs against MPSAs. In terms of delay and power, MPSAs perform better than VPSAs. One of the reasons for this is the large number of series-connected vias in VPSAs. This affects both delay and power. This also shows an important aspect of the interconnect architecture: for better performance, a routing fabric should use fewer series-connected vias to make connections between the fixed metal segments. In terms of die-cost, VPSAs can be more cost effective than MPSAs. However, the cost effectiveness of VPSAs depend on various factors such as logic block layout area, number of customizable via layers, and device volume requirements.

In the future, we plan to extend this work by looking at other possible interconnect structures. This includes studying fabrics that have many fixed routing layers, but customization can only be performed using a single via layer. We also plan to explore routing fabrics in which the metal segments at different layers have different lengths. Also, we plan to study the effect of whitespace insertion on the overall results. Finally, we also plan to perform a detailed sensitivity analysis of the VPSA die-cost to volume requirements.

7. REFERENCES

[1] `http://www.eetimes.com/showArticle.jhtml?articleID=217200925`.

[2] Nextreme Structured ASIC, eASIC Corp. `http://www.easic.com/pdf/asic/nextreme_asic_structured_asic.pdf`.

[3] Hardcopy II Datasheet, Altera Corp. `http://www.altera.com/literature/hb/hrd/hc_h5v1_05.pdf`.

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

[5] U. Ahmed, G. Lemieux, and S. Wilton. Area, delay, power, and cost trends for metal-programmable structured asics (MPSAs). In *IC-FPT*, Dec. 2009.

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

[7] 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.

[8] V. Kheterpal, A. J. Strojwas, and L. Pileggi. Routing architecture exploration for regular fabrics. In *DAC*, 2004.

[9] A. Koorapaty, V. Kheterpal, P. Gopalakrishnan, M. Fu, and L. Pileggi. Exploring logic block granularity for regular fabrics. In *DATE*, 2004.

[10] I. Kuon and J. Rose. Measuring the gap between FPGAs and ASICs. *IEEE Trans. on CAD*, 26(2):203–215, 2007.

[11] Z. Or-Bach. Paradigm shift in ASIC technology: In-standard metal, out-standard cell. *eASIC White Paper*, September 2005.

[12] 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 *DAC*, 2003.

[13] Y. Ran and M. Marek-Sadowska. Via-configurable routing architectures and fast design mappability estimation for regular fabrics. In *ICCAD*, pages 25–32, 2005.

[14] Y. Ran and M. Marek-Sadowska. Via-configurable routing architectures and fast design mappability estimation for regular fabrics. *IEEE Trans. on VLSI*, 14(9):998–1009, Sept. 2006.

[15] J. A. Roy and I. L. Markov. High-performance routing at the nanometer scale. In *ICCAD*, pages 496–502, 2007.

[16] 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.

[17] B. Zahiri. Structured ASICs: Opportunities and challenges. In *ICCD*, page 404, Oct. 2003.