STRUCTURED LOGIC ARRAYS AS AN ALTERNATIVE TO STANDARD CELL ASIC

by

Roozbeh Mehrabadi

B.A.Sc., the University of British Columbia, 1995

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

The Faculty of Graduate Studies

(Electrical and Computer Engineering)

The University of British Columbia

December 2005

© Roozbeh Mehrabadi 2005
ABSTRACT

In the deep submicron (DSM) era, design rules have become increasingly more stringent and have favoured the more structured architectures. The design methods using standard cell ASICs (SC-ASIC) produce randomly placed gates and interconnects. Beside reduced yield, they also suffer from high testing cost, even with the most advanced built-in self-test methods. These shortfalls motivate us to search for an alternative architecture in the structured logic arrays. First, we will explore the available structured logic arrays and their potentials as alternatives to SC-ASIC architecture. Then we will focus on programmable logic arrays to explore their potential when competing for speed and area with SC-ASIC. We have investigated the critical path delay for clock-delayed PLA and suggested equations for quick calculation of its capacitive loads and delay. We have also introduced equations to calculate their area using technology-independent parameters. This would help the front-end CAD tools in partitioning and architecture decision-making before committing to a specific technology. We found that circuits with higher than 200 product terms have slower PLA implementations than SC-ASIC. They also tend to take more than 10 times the area. Furthermore, we have introduced logical effort as a simple method for gate sizing and optimization of the PLA’s critical path delay. Finally, we have introduced methods to subdivide the slower PLAs in order to improve the overall circuit timing. We also found that by dividing a circuit to two PLAs we can cut its delay by half and keep the increase in area minimal.
# TABLE OF CONTENTS

Abstract .................................................................................................................. ii
Table of Contents ................................................................................................. iii
List of Tables ........................................................................................................... v
List of Figures ........................................................................................................ vi
Acronyms ................................................................................................................. vii
Acknowledgements ................................................................................................ viii
Chapter 1 Introduction ......................................................................................... 1  
  1.1 Research Motivation ................................................................. 2
  1.2 Research Goals ................................................................. 4
  1.3 Thesis Outline ................................................................. 5
Chapter 2 Background ....................................................................................... 6  
  2.1 Read Only Memory (ROM) ............................................. 6
  2.2 Programmable Logic Array (PLA) .................................. 7
  2.3 Field Programmable Gate Array (FPGA) ....................... 10
  2.4 Structured ASIC (SA) .................................................. 11
    2.4.1 Choices in SA basic tile .................................... 12
    2.4.2 SA configuration methods .................................. 14
    2.4.3 More on Via-Configurable SA ............................ 15
    2.4.4 SA advantages and disadvantages ....................... 17
  2.5 Why choose PLA ......................................................... 17
Chapter 3 PLA vs. ASIC Comparisons ............................................................. 19  
  3.1 PLA structure used ......................................................... 20
  3.2 PLA critical path calculations .......................................... 21
  3.3 Line delay (RC effects) .................................................. 26
  3.4 PLA area calculations .................................................. 27
  3.5 Standard Cell flow and data collection ......................... 29
  3.6 PLA vs. SC-ASIC ....................................................... 30
    3.6.1 SPICE verification of Delay calculations ............. 30
    3.6.2 Delay Comparison .............................................. 31
    3.6.3 Verification of area calculations for the PLA structure used 34
    3.6.4 Area Comparison ............................................... 36
  3.7 Computing Power Factor for PLA .................................... 37
Chapter 4 Techniques for PLA Area-Delay-Power Improvement .................... 39  
  4.1 Using Logical Effort for PLA delay improvement .............. 39
    4.1.1 Basic Logical Effort Optimization ..................... 40
    4.1.2 Modified Logical Effort Optimization .............. 42
    4.1.3 Delay estimation with Logical Effort ............... 43
    4.1.4 Considering Power with Logical Effort ............ 44
  4.2 Using Partitioning for PLA delay Improvement ................. 47
    4.2.1 Minimum Delay PLA (single output) .............. 48
    4.2.2 Optimizing with Multiple Output PLA .......... 50
Chapter 5 Conclusions and Future Directions ............................................... 53  
  5.1 Future Work .............................................................. 55
5.2 Contributions........................................................................................................56
REFERENCES .............................................................................................................58
LIST OF TABLES

Table 2.1: Product Terms used in the Sample Clocked PLA ........................................... 9
Table 3.1 Minimum size tile calculations based on $\lambda$ ............................................. 27
Table 3.2: Correlation relation between IPO and Capacitive loads ................................. 34
Table 3.3: Sample PLA layout size scaled from $\lambda=1\mu m$ to $\lambda=0.1\mu m$ .................. 35
Table 3.4: Total capacitance and calculated $\alpha$ for a 5-8-4 PLA .................................. 38
Table 4.1: Five stage gate optimization with LE ......................................................... 42
Table 4.2: Five stage gate optimization with modified LE ............................................. 43
Table 4.3: Timing comparison between fixed and optimized gate PLAs ......................... 43
Table 4.4: Table of partitioned PLAs delay/area and their %difference with their full PLA ........................................................................................................... 52
LIST OF FIGURES

Figure 1.1: Area overhead vs. Power-Delay Comparison Graph with respect to SC-ASIC 3
Figure 2.1: A 2 input 4 output ROM with transistor based fabric ............................................. 7
Figure 2.2: An example 4-7-2 (IPO) clocked CMOS PLA .......................................................... 8
Figure 2.3: General structure of an FPGA ................................................................................ 11
Figure 2.4: The generic SA structure ..................................................................................... 12
Figure 2.5: SA tiles (a) fine-grain, (b) LUT based medium-grain, (c) master tile with 4x4 base tiles ...................................................................................................................... 13
Figure 2.6: A via configurable 3-input LUT with full complementary input ......................... 15
Figure 2.7: An example of a coarse-grained tile using 3-input LUT ........................................ 16
Figure 2.8: SA interconnect routing example ........................................................................ 16
Figure 3.1: The scatter plot of the benchmark circuits’ I/O numbers ........................................ 19
Figure 3.2: The 4-7-2 (IPO) clocked CMOS PLA with critical path marked .......................... 20
Figure 3.3: Schematic of the Critical path for our Sample PLA ............................................. 22
Figure 3.4: The Critical Path’s signal wave forms .................................................................. 23
Figure 3.5: Basic NMOS pass-transistor tile dimensions ...................................................... 27
Figure 3.6: The SC-ASIC design flow and CAD tools used .................................................... 29
Figure 3.7: Plot of Spice simulation over calculated delays with respect to their no. of products ................................................................................................................................. 31
Figure 3.8: PLA circuit delay data sorted with respect to their number of product terms 32
Figure 3.9: Graph of PLA to SC-ASIC delay ratios sorted with respect to the PLA product terms ................................................................................................................................. 33
Figure 3.10: The layout view of the sample PLA ....................................................................... 35
Figure 3.11: PLA area / SC-ASIC area vs. number of product terms .................................... 36
Figure 3.12: The percentage of area ratios in 4 different ranges .......................................... 37
Figure 4.1: Pre-charge and Evaluation Critical Paths and their side loads ............................. 40
Figure 4.2: Graph of normalized PDP and Delay vs. FO factor ........................................... 45
Figure 4.3: Sizing for a load of 64 ......................................................................................... 46
Figure 4.4: Lower size gates with the original Cout ............................................................... 46
Figure 4.5: Circuit Partitioning (a) based on Input to Output path (b) based on output’s input and product term requirements .................................................................................. 48
Figure 4.6: Percentage difference of delay between full and single-output PLA implementations ................................................................................................................................. 49
Figure 4.7: The percentage of circuits in the three ranges ......................................................... 50
Figure 4.8: The Delay vs Area change with respect to the full PLA ........................................ 51
**ACRONYMS**

<table>
<thead>
<tr>
<th>ACRONYM</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASIC</td>
<td>Application Specific Integrated Circuit</td>
</tr>
<tr>
<td>ATE</td>
<td>Automated Test Equipment</td>
</tr>
<tr>
<td>BE</td>
<td>Branching Effort</td>
</tr>
<tr>
<td>BIST</td>
<td>Built-in Self-Test</td>
</tr>
<tr>
<td>CAD</td>
<td>Computer Aided Design</td>
</tr>
<tr>
<td>CMOS</td>
<td>Complementary Metal-Oxide-Silicon</td>
</tr>
<tr>
<td>Core</td>
<td>The central part of the design, usually surrounded with I/O pads</td>
</tr>
<tr>
<td>DFT</td>
<td>Design for Test</td>
</tr>
<tr>
<td>DSM</td>
<td>Deep Sub-Micron</td>
</tr>
<tr>
<td>FF</td>
<td>Flip-Flop</td>
</tr>
<tr>
<td>FO4</td>
<td>Fan-out of 4 (ratio of $C_{load}/C_{in} = 4$)</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array</td>
</tr>
<tr>
<td>IC</td>
<td>Integrated Circuit</td>
</tr>
<tr>
<td>IEEE</td>
<td>Institute of Electrical and Electronics Engineers</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output</td>
</tr>
<tr>
<td>IPO</td>
<td>Input Output Product terms, a 3-tple number (I,P,O)</td>
</tr>
<tr>
<td>ITRS</td>
<td>International Technology Roadmap of Semiconductors</td>
</tr>
<tr>
<td>LE</td>
<td>Logical Effort</td>
</tr>
<tr>
<td>LFSR</td>
<td>Linear Feedback Shift Register</td>
</tr>
<tr>
<td>MUX</td>
<td>Multiplexer</td>
</tr>
<tr>
<td>PI</td>
<td>Primary input</td>
</tr>
<tr>
<td>PLA</td>
<td>Programmable Logic Array</td>
</tr>
<tr>
<td>PO</td>
<td>Primary output</td>
</tr>
<tr>
<td>ROM</td>
<td>Read-Only Memory</td>
</tr>
<tr>
<td>RTL</td>
<td>Register Transfer Level</td>
</tr>
<tr>
<td>SA</td>
<td>Structured ASIC</td>
</tr>
<tr>
<td>SE</td>
<td>Stage Effort (with a '*' indicates the optimized SE)</td>
</tr>
<tr>
<td>SLA</td>
<td>Storage/Logic Array</td>
</tr>
<tr>
<td>SC-ASIC</td>
<td>ASIC design synthesis using Standard Cell logic library</td>
</tr>
<tr>
<td>SoC</td>
<td>System on a Chip</td>
</tr>
<tr>
<td>VHDL</td>
<td>VHSIC (Very High Speed Integrated Circuits) Hardware Description Language</td>
</tr>
<tr>
<td>VLSI</td>
<td>Very Large Scale Integration</td>
</tr>
</tbody>
</table>
ACKNOWLEDGEMENTS

It has been a great pleasure to work with Dr Resve Saleh on this thesis. He has always been ready with advice and direction and I have appreciated his willingness to help, with graciousness and careful attention to details. His dedication to research and diligent work ethics have been a great source of inspiration for me. Thank you.

I am also thankful to Dr Steve Wilton and Dr Shahriar Mirabbasi for serving on my thesis committee.

The SOC faculty have always been ready to answer questions, and been supportive teachers in the courses I have taken with them. I would also like to thank my colleagues at UBC SOC lab, Dr Roberto Rosales and Sandy Scott, for being kind and supportive.

And last, but certainly not the least, I am grateful to my family. My beloved wife Farah, who has been supporting my efforts over an extended time, and has been my source of inspiration for time management and hard work; and our boys, Arman and Zubin, who have been my cheerleaders and promised me a prize when I finish with my thesis. They have taught me what is important in life.

Finally, I would like to thank PMC-Sierra, NSERC and Canadian Microelectronics Corporation for funding this research and the supporting CAD tools.
CHAPTER 1 INTRODUCTION

Before the deep submicron era, CMOS fabrication processes could handle integrated circuit (IC) designs with complex circuits and complex geometric patterns to be produced on a chip with an acceptable yield. Today, in fabrication lines with feature sizes below 100nm, the manufacturing process is so complicated and the resolution of layout masks is so fine that design for manufacturability is a primary concern. As a result, for these technologies, foundries have imposed more stringent design rules to keep the yield at acceptable levels. These design rules limit the randomness of the designs and favour the more structured layouts for manufacturing. For example, the first IC designs which are accepted for fabrication on 65nm fabrication runs are field-programmable gate arrays (FPGAs). The highly-structured design patterns of FPGAs allow the foundry to fine-tune their process to these regular structures in order to improve yield.

By moving into deep submicron technologies, it is possible to integrate hundreds of millions of transistors onto a chip. Transistors are considered to be almost "free". Therefore, designers have more transistors than they need to implement the required functions. To make use of the extra transistors or unused silicon area, very often extra memory blocks are integrated into the chips [1]. Memories are also structured arrays, which can be produced using CAD tools known as memory compilers [2]. Their development is much easier than random logic and they generally are low-power circuits. In contrast, any increase in complexity of logic circuits leads to a sharp increase in their design and verification time, as well as increases in power and cost.

These extra transistors can also be used to address the at-speed testing problem in modern chips. According to the International Technology Roadmap for Semiconductor (ITRS)[3], the
manufacturing yield loss associated with at-speed functional test methodology is directly related to the slow improvement of automatic test equipment (ATE) performance and the ever increasing device I/O speed. By adding design-for-testability (DFT) and built-in self-test (BIST) features onto the chips, designers attempt to reduce the reliance on high-cost, full-feature testers. The benefit of this approach for random logic, however, is limited because of the low fault coverage with logic BIST due to the use of pseudo-random patterns rather than deterministic patterns [4].

1.1 Research Motivation

The issues described above suggest the use of some type of structured array for logic design in the future. For example, FPGAs are highly structured with programmable logic in the form of lookup tables and programmable interconnect which are configured after fabrication. These flexibilities, however, come at the cost of much slower speed, higher power consumption and more silicon area compared to the non-programmable instances implemented using the SC-ASIC flow [5]. A more recent approach is structured ASIC which places itself between FPGAs and ASIC design using standard cells [6]. Structured ASIC fabrics have programmable lookup tables that can be modified post fabrication and programmable interconnect that can be modified in the last step before final fabrication. Although structured ASIC’s performance is better than FPGA, its area, delay and power is more than standard cell ASICs. In Figure 1.1, the positions of FPGA, structured ASIC and custom design is shown in a graph relative to SC-ASIC in terms of area, power and delay [1,6].
In order to reduce the area and delay overhead of structured ASIC, we could view it as an improvement over FPGAs. Then, if we conceptually remove their metal layer programmability feature to reduce some of the fabric’s overhead, it would better compete with SC-ASICs. In other words, logic synthesis using standard cells would be replaced with automatic synthesis of structured logic fabrics. They can be highly-customized functional blocks with potential improvements over SC-ASIC design. To accomplish this, we must revisit a number of fixed-function structured logic arrays to investigate the possibility of finding alternatives to SC-ASIC. This possibility has already been recognized by others in the field [7].

Commonly known structured logic arrays are Read-Only Memories (ROM), Programmable Logic Arrays (PLA) and Storage-based Logic Arrays (SLA). In ROMs the output is stored in the memory locations associated with their addresses as the corresponding
input. For an n-input and m-output ROM $2^n \times m$ memory locations are needed. The core array (storing the bits) is usually formed as fabrics with horizontally and vertically crossing wires. An address decoder, based on the given input, selects the proper word lines and bit lines that lead to the outputs. Devices based on a ROM architecture tend to be area and delay intensive and PLAs were introduced to improve on the ROM efficiency by using only the needed product terms. In PLAs, the output is formed as the sum-of-product terms (NAND-NAND) or product-of-sum (NOR-NOR) from its inputs. This makes PLAs suitable for implementing combinational logic; however, to implement state machines and sequential logic, SLAs were introduced [8]. SLAs follow the PLA layout of AND and OR planes, except that they are folded together, and memory elements such as flip-flops and latches are placed in various locations on the layout. The benefit of SLA diminished with more sophisticated CAD tools and availability of larger computing powers.

In this research, we concentrate our efforts on PLAs because they have a long history [9,10] and substantial amount of work has been done on improving their power and delay factors [11,12]. Furthermore, because of its regularity, a single PLA structure has well known timing and power characteristics and does not require the technology-mapping step of ASIC design flow. Although it is an older design style, it is a good starting point for this research direction. In fact, there has been resurgence in interest in this topic due to the inherent structured characteristics of PLAs [11, 12, 23, 36].

1.2 Research Goals

The overall goal of this study is to investigate structured logic fabrics such as PLAs as a possible alternative to ASIC design using standard cells. We will explore the area-delay tradeoffs for PLA and standard cell ASIC designs using some benchmark circuits. We will
also develop methods to partition a given circuit to a number of PLAs to minimize their delay and area in order to compete with standard cell ASIC.

Using a set of benchmark circuits, the objectives are to:

1- Produce layouts from a commercial grade standard cell library and compare their area and delay with those of their PLA implementation.

2- Explore options to improve delay in PLAs using logic effort.

3- Find parameters or heuristics to partition slow PLAs to achieve an acceptable delay (based on the delay achievable by standard cell ASIC).

4- Recommend future improvements to include PLAs in SoC design flow.

The ultimate goal is to find PLA architectures which will help to close the gap, as shown in Figure 1.1, between structured ASIC and the traditional ASIC design using standard cells.

1.3 Thesis Outline

Chapter 2 provides some background on different structured logic devices and offers reasons for choosing the PLA structure for this study. The method of implementing the circuits as PLAs, and the CAD flow used for creating cell library counterparts of the same circuits are presented in Chapter 3. Also presented are the equations used for PLA delay/area estimates as well as SPICE simulation data and comparisons with standard cell synthesis in this chapter. We also estimate PLA activity factor and power at the end of this chapter.

In Chapter 4, we will present our methods to create PLAs of moderate size that are small and fast. Further discussion on circuit partitioning and heuristics to speed up the process are presented in this chapter. Conclusions and future directions are presented in Chapter 5.
CHAPTER 2 BACKGROUND

In this chapter, we review the available structured logic arrays, and discuss their merits for use in place of SC-ASIC. First, we review ROM and PLAs, then CPLDs, and finish with FPGAs and structured ASICs. At the end of this chapter, we will discuss the reasons for targeting PLAs for our research.

2.1 Read Only Memory (ROM)

One way to represent complex logic functions is to implement lookup tables as ROMs. In these devices, the outputs are stored in the memory locations pointed to by the corresponding input. For an \( n \)-input and \( m \)-output device, we would need a ROM with a capacity of \( 2^n \times m \) bits. In this way, all possible logical combinations of the input could be stored, and made available by storing the output. Figure 2.1 shows an example of a small ROM core with an address decoder on the left and its outputs present at the bottom. In this example, all output lines (bit lines) are initially pulled up high. For each input combination, the decoder selects one horizontal line (word line) and pulls it up. The output lines associated with a logic zero are connected to the ground via a pass-transistor, whose gate is connected to the associated horizontal line. This method requires enough pull-up capacity for the decoder to charge up the capacitances of the word line and the transistor gates attached to it. The ROM input to output timing depends on the decoder's pull-up and the pass-transistors' pull-down delay. As a result the ROM device becomes rather slow as it grows in size and consumes increasingly more power. To improve the pull-up/down delay in larger ROMs, sense amplifiers are used to pull down the output line as soon as the current flow is detected [13]. As a result, smaller
pass-transistors could be used to reduce the overall size of the ROM and yet achieve higher speeds [14].

Because ROMs implement all possible combinations of the input, their use for logic devices which have “don’t-care” terms, (some of their outputs are not dependent on all of their inputs) is not efficient. They are also wasteful in terms of area and power [7]. As a result, PLAs were introduced to implement only the needed product terms for every output.

2.2 Programmable Logic Array (PLA)

The PLA designs, introduced in the last 20 years, fall into 2 main categories: static designs using nMOS technology and clocked designs using pre-charged gates in CMOS [15]. The static CMOS PLAs use a NAND-NAND or NOR-NOR structure and tend to occupy a larger area and are slower than the clocked version. Since we are targeting CMOS technology and most recent advances in PLA delay reduction have been done in clocked PLAs [11] [12], we have focused our efforts on the clocked PLAs. The static PLAs could still play a role in reconfigurable fabrics which use very small size PLAs, as noted in [16] and [17].
An example of the clocked PLA is shown in Figure 2.2. It has 4 inputs, 7 product terms and 2 outputs, i.e., IPO = 4x7x2. It produces the following two outputs:

\[ o_0 = P_1 + P_2 + P_3 + P_4 \]
\[ o_1 = P_5 + P_6 + P_7 \]

Table 2.1 shows the seven product terms in this PLA as dot products of the inputs. The trailing '~' indicates an inverted input. In this PLA, the AND-plane is located at the top and the OR-plane at the bottom separated by the inter-plane buffers. The AND-plane implements the products of the inputs and passes them, via inter-plane buffers, to the OR-plane to sum them up and pass the result to the output buffers.
Table 2.1: Product Terms used in the Sample Clocked PLA.

<table>
<thead>
<tr>
<th>P1</th>
<th>P2</th>
<th>P3</th>
<th>P4</th>
<th>P5</th>
<th>P6</th>
<th>P7</th>
</tr>
</thead>
<tbody>
<tr>
<td>~i0.i1.~i2.i3</td>
<td>~i0.~i1.~i2.~i3</td>
<td>i0.~i1.i3</td>
<td>i1.i2.~i3</td>
<td>i0.~i1.i2.i3</td>
<td>i0.i1.~i2.~i3</td>
<td>~i0.i1.i2</td>
</tr>
</tbody>
</table>

In order to compute the delay through the PLA, it is useful to understand its operation. In this PLA structure, vertical lines connected to the pull-up transistors and inter-plane buffers act as the product lines. The horizontal lines in the AND-plane are connected to the inputs or their complements. Only the input or its complement is part of a product term. When an input line is part of a product term, it is connected to the gate of a pass transistor, which could discharge the product line to the pull-down line. When the clock signal “Phi” is low, the product lines are charged up to \( V_{DD} \). This is called the pre-charge period. The evaluation period happens when Phi goes high. At this time the pre-charged product lines would stay high if they are not connected to the pull-down lines via a pass transistor (i.e., their inputs are not high). Otherwise, they will be discharged low.

The clock signal controlling the pre-charge and evaluation in the OR-plane is delayed so that the buffered product terms are ready for evaluation before the clock arrives. In the OR-plane the buffered product lines associated with each output are connected to its horizontal output line via pass-transistors. During the pre-charge period, the output lines, which are connected to the output inverters, are charged high. During the evaluation period, if one or more product lines connected to the output line are high, the output line will be discharged low and the resulting output will be high. In the next chapter, we will present the critical path for our PLA example and discuss its important characteristics.

The idea of folding PLAs and adding storage elements was first introduced by S. Patil [18], and later in more detail by Patil and Welch [19]. In this method, called storage/logic array (SLA), different memory elements are placed on the folded PLA fabric with column
and row breaks to form independent state machines and logic blocks which make up the final integrated circuit (IC). The SLA program enables the logic designer to have a physical view of the final layout without the need to interact with a layout designer. However, folding PLAs and introducing column/row breaks takes away the uniformity of interconnects and transistors present in PLAs. Any design change would require unfolding and remapping of the SLA and would lead to major changes in a layout plan which might have fixed dimensions for the device. Minor design changes in a PLA would be matter of reallocating the pass-transistor tiles. Furthermore, with access to current computing power and CAD tool support, the visual simplicity of the SLAs is not as attractive as it was 20 years ago.

2.3 Field Programmable Gate Array (FPGA)

In an attempt to give full post-manufacturing re-configurability to the user, gate array vendors introduced FPGAs. An FPGA is comprised of many uncommitted basic logic blocks surrounded by programmable interconnect fabric and input/output (I/O) blocks (see Figure 2.3). Each logic block provides combinational logic equivalent of a few gates and one or more registers (flip-flops). Many studies have been done on partitioning and technology mapping for FPGAs. In typical industrial FPGAs, a 4 to 5 input look-up table (LUT) and a flip-flop is used as the basic logic block with multiplexers to control signal gating [5].

At this time, FPGAs provide the highest capacity general-purpose programmable logic chips. They could contain 1 to 6 million equivalent gates and include RAM, micro-controllers and other pre-designed modules [6]. Their programmability makes them suitable for most communication and multi-media applications where the technology and protocols are constantly changing or some hardware changes need to be applied while in use. In some products, such as network switches and routers, FPGAs are the only choice to reconfigure the
hardware while in service to adapt to network changes. While FPGA chips are expensive, their non-recurring engineering (NRE) cost is close to zero since they are ready made chips and no more fabrication process is needed. Finally, their highly-structured design makes them suitable for current and future DSM fabrication technologies.

FPGA programmability, however, comes at a very high cost of area and performance penalty when compared to standard cell (SC) ASIC design (see Figure 1.1). They also consume much more power and have higher unit costs [6]. This makes them suitable primarily for prototyping or low volume production runs.

2.4 **Structured ASIC (SA)**

To bridge the gap between FPGAs and ASIC design using standard cells, gate array vendors [20] have introduced structured ASIC. Similar to gate arrays, a structured ASIC contains an array of highly-structured and optimized elements (tiles) to form a logic fabric. Each tile contains a small amount of generic logic that might be implemented with a

![General structure of an FPGA](image)

*Figure 2.3: General structure of an FPGA*
combination of gates, LUTs and/or multiplexers. The tiles may also contain one or more registers and small amount of local RAM. The prefabricated SA contains an array of tiles and may also contain configurable I/O, RAM blocks and IP cores. In Figure 2.4, a general SA structure is shown with embedded RAM and I/O or IP cores around it.

In the description so far, the SA device resembles a modern gate array structure. Their main difference is that, in SA structures, all non-metal layers and some metallization layers have already been fabricated and the wafers are stored for later completion. The user is only left with a few (may be one or two) metallization layers to specify. Therefore, the turn-around time from final specification to a completed chip is only 1 to 2 weeks.

### 2.4.1 Choices in SA basic tile

Most SA vendors offer a variety of IP cores and I/O blocks that to their customers. The main difference between their devices is in the architecture of their basic tiles. They range from very fine-grained, to medium and course-grained tiles.
The fine-grained tile usually contains a number of unconnected basic devices such as transistors and resistors (see Figure 2.5a). These tiles are pre-connected in various configurations. The logic engineer can use the available metal layers to make the necessary connections and join the tiles to the local and global routing structure.

Other vendors have chosen medium-grain tiles which contain generic logic in the form of LUTs or multiplexers and one or more registers (Figure 2.5b). In each of these tiles, the global connections and polarity of the register clocks (positive or negative edge-triggered) can be set by the remaining metal layers. The denser tiles are formed hierarchically by combining some base tiles which contain generic logic, multiplexers and memory elements. Figure 2.5c shows a sample master tile containing a 4x4 array of base tiles. The master tiles could have 8x8 arrays of base tiles or larger. Then a sea of master tiles is prefabricated on the chip.

The choice of fine vs. medium vs. coarse grained tiles sets the density of interconnects going into and out of the tiles. The fine-grained tiles require a lot of connections compared to the amount of functions they provide. The more coarse-grained tiles tend to require fewer connections. On the other hand, the fine-grained tiles provide more design flexibility as compared to the more coarse-grained tiles. Evaluation of tile sizes and architectures is a
critical subject for research and development and needs to mature like the FPGA architectures [6].

2.4.2 SA configuration methods

All architectures using the above tiles require one or two metal layers and possibly a via layer for final routing and configurations. At least one vendor has introduced a device which only requires the configuration of one via layer. They all require CAD tools to place the design onto the SA devices and route the final interconnects.

In all tile based architectures, there is trade-off of functionality vs. area efficiency. For example, the use of extra registers in the tiles may improve their functionality for some applications, while if not used, they tend to waste area. In [21], they have introduced a number of via-configurable functional blocks and a via-decomposable flip-flop. By decomposing the pre-fabricated flip-flops, they can reuse their parts for other functions and save the usually wasted space. They show that their via-configured blocks could cover all logic functions and claim that it has a comparable delay to other architectures, including PLAs. Furthermore, they report the highest transistor utilization as compared to other fabrics.
2.4.3 More on Via-Configurable SA

One type of reconfigurable fabric uses via programmable lookup table (LUT), as shown in Figure 2.6. This example shows a 3-input LUT using three pairs of pass-gates as 2-to-1 multiplexers. It is very similar to a FPGA LUT, except that it does not need the SRAM memory cells to store the programmable bits. Hence it has $1/8^{th}$ of the transistor density of a FPGA LUT. In this example, the three signals A, B, and C and their complements are applied to the LUT, and empty via locations are filled to make the necessary connections. To implement the given function, out, first we need the signal A. The two via connections are made from the VDD line to the left-most multiplexer. Then the third via is placed on the GND line to the first connection of the second multiplexer to pass the complement of signal B. The fourth via connects signal C to the second multiplexer’s second connection to generate the AND function between $\overline{B}$ and C.

![Figure 2.6: A via configurable 3-input LUT with full complementary input.](image-url)
This structure is used in commercial reconfigurable fabric as a medium to coarse-grained tile, an instance of which is shown in Figure 2.7 [22]. This example also contains a scanable flip-flop and some memory elements to increase its functionality. The via-configurable fabrics typically have the first two metal layers routed and even vias at these two levels placed. This is to ensure maximum uniformity in printed masks for the lower metal layers which are usually the most difficult ones to print. The configurable interconnect and via layers are usually at higher layers. An example of interconnect routing and via placement at
higher metal layers is shown in Figure 2.8. Notice the uniform routing of metal 4 and 5 and allowance for non-uniform routing of metal 6 layer.

2.4.4 SA advantages and disadvantages

Because in SA all devices have already been placed and most routing except for a few upper metals and via layers have been done, there is much less work in final mask generation. As a result, the non-recurring engineering cost (NRE) for the customer is dramatically reduced, and the turn-around time for bug fixes and re-runs are much lower as compared to the SC-ASIC flow.

However, the SA devices are still around 3 times slower than SC-ASIC designs and take as much as 3 times more area. They also consume more power [6, 21]. Furthermore, due to their recent introduction in the field, there is a lack of optimized EDA tools for their technology mapping and evaluation [6]. Currently, the vendors provide the traditional ASIC tools with their devices, which are not well-suited to a cell-based design flow. The FPGA CAD tools are generally “cell-aware” and are more suited to tile placement and block-based routing. Nevertheless, a long-term investigation of SA architectures is required to bring them up to the level of maturity that FPGA structures have achieved.

2.5 Why choose PLA

Considering the gap that exists between SA and SC-ASIC and existence of room for improvement with custom design (see Figure 1.1), we investigate PLAs as an alternative to SC-ASIC to determine its potential as a future logic fabric. PLAs are well-understood and have been subject of some very recent investigations [5,11,12,16,17, 23,25]. They have also been mentioned by some experts as poised for a come back [7]. Furthermore, a recent
investigation [23] concludes that dynamic PLAs use less power and should be considered for further investigation. Before moving onto the other structures, it is important to investigate PLA characteristics and decide if it is a useful structure to replace SC-ASIC.
CHAPTER 3  PLA VS. ASIC COMPARISONS

In this chapter, we present the PLA structure we used for evaluation and comparison with standard cell ASIC (SC-ASIC). Our goal is to set determine the advantages and disadvantages of PLA structures with respect to SC-ASIC. Moreover, we will gain insight as to how to improve the PLA architectures. Another reason for this analysis is that we will need CAD tools to partition the logic functions and make architectural decisions. The tools require models and simple expressions for estimating area, delay and power at a reasonable speed. We have developed these models for clock-delayed PLAs, and in the next chapter, we will investigate improving PLA structures.

We used about 150 combinational benchmark circuits in the Berkeley “.pla” format to compare their implementation using SC-ASIC as well as PLA layout. They were obtained from “Berkeley PLA Test Set” which is included with Espresso as part of the SIS package [24]. They are combinational circuits implementing various arithmetic and industrial control functions such as: ALU, control, adders and multipliers. The scatter plot of their I/O numbers

![Figure 3.1: The scatter plot of the benchmark circuits' I/O numbers](image-url)
is shown in Figure 3.1. Over 70% of the benchmark circuits have I/O numbers less than or equal to 20. We will discuss the critical path for the PLA structure used and our method of data collection for the PLA and SC-ASIC implementations of the benchmarks in the following sections.

3.1 PLA structure used

We have used the generic clock-delayed PLA structure (as shown again in Figure 3.2) as the base PLA in our study. The clock-delayed PLAs have been the subject of many recent studies for power and delay improvement, which makes them good candidates for replacing SC-ASIC [25,11,12].
There may be room for further area and/or power-delay improvement; however, this was not the goal of our comparison. We focused on area and delay measurement of PLAs as compared to SC-ASIC, the development of equations for fast comparison, and development of methods for partitioning circuits to achieve area, delay and power efficiency in their PLA implementation. It is important for partitioning tools to quickly estimate area, delay and power for a PLA implementation of a given circuit.

For delay, we calculate the total capacitive loads and the equivalent resistance \( R_{eq} \) of the gates on the critical path. For area, we will extract the basic tile sizes (pass-transistor and pull-up/down transistors) and sum them up with the buffer areas based on the IPO number of the PLA\(^1\). For power, we take the total capacitance and drive a value for the activity factor, \( \alpha \), since the average power is calculated as: 
\[
P = \alpha CV_{DD}^2 f.
\]

The delay, area and power calculation results are presented in the next sections.

### 3.2 PLA critical path calculations

In the PLA example shown in Figure 3.2, the critical-path starts from the clock signal \( \Phi \), follows the \( P2 \) vertical line and ends with output \( o0 \). It is marked with a dotted line. An extracted schematic of the PLA’s critical-path is shown in Figure 3.3. To compute the worst-case delay using this circuit, only one of the transistors attached to the product line and one attached to the output line are assumed to be on at any given time. Next, we will discuss the equations that we developed for quick calculation of the PLA capacitive loads.

\(^1\) Here, IPO stands for input-product term-output. These three factors are used as multipliers in the area calculations.
The delay of a path containing logic gates and capacitive loads is computed with the equation: \( D = \sum R_i C_i \) where \( R_i \) is the effective resistance (\( R_{eff} \)) of the \( i^{th} \) gate, and \( C_i \) is the capacitive load the gate has to drive [29,30]. To calculate the critical path of a clocked PLA, we need to sum up the total delays that set the minimum clock period of the PLA.

The PLA clock cycle has two parts: pre-charge and evaluation time. During the pre-charge time the clock signal is low and the vertical product lines or the horizontal OR lines are charging up. At evaluation time, the clock signal goes high and the pull-down line of the product lines or the OR lines are pulled low. At this time, the lines attached to the activated pass-transistors will be evaluated low. The pre-charge timing depends on the size of clock buffers (\( R_{1/2} \)), pull-up/down transistors (\( R_{ud} \)), inter plane buffers (\( R_{3/4} \)) as well as the loads they need to service: \( C_{CLK}, C_{AND}, C_{INTR}, C_{CLK\_delay} \) and \( C_{OR} \) (see Figure 3.3).

To find the maximum operating frequency, we need to compute the pre-charge time \( t_P \) and evaluation time \( t_E \). The delay between clock signal Phi going low until the node \( n5 \) (final stage of the product line) has gone high is shown as \( t_p \). The \( t_E \) is the delay between Phi going high until the node \( o0 \) has switched from low to high. The signal waveforms associated with
each PLA internal node and their relative occurrences are shown in Figure 3.4. The minimum $t_p$ is from \( \Phi \) high-to-low edge until node \( n5 \) low-to-high edge. The minimum $t_e$ is the delay between \( \Phi \) going high until output \( o0 \) is switching. Also, the clock signal applied to the OR-plane cannot be the same as the clock applied to the AND-plane. The delay between \( \Phi l \) and \( \Phi l \_delay \) signals should allow the evaluation of the product terms in the AND-plane to happen before the evaluation in the OR-plane. In other words, the pre-charged period of node \( n5 \) must not coincide with the evaluation time at node \( n6 \). Otherwise, a race condition will occur and the voltage level at node \( n6 \) will be pulled down in error (see Figure 3.4).

Since the OR-plane pre-charge time ends right after node \( n5 \) has been stabilized, the pre-charge time only needs a small increase over $t_p$. Notice that input delay (nodes \( i0 \) and \( n2 \)) does not affect the timing directly. However, for the PLA to function properly, it is required
that the input arrives at node \( n2 \) before evaluation time starts at Phil. It should also remain constant for the duration of the evaluation time (see Figure 3.4).

The sum of pre-charge and evaluation times plus a safety factor of 10% gives us the minimum clock period for our PLA to function properly. In order to compute the PLA timing we need to obtain the capacitive loads and calculate buffer and pull-up/down transistor timings. To do this we need the gate’s intrinsic delay \( \tau_{int} \) and effective resistance \( R_{eff} \). The intrinsic delay of a device is measured with a self-capacitive load, while \( R_{eff} \) gives the device delay time per unit capacitance of the load. For our calculations and simulations, the Artisan cell libraries for TSMC 0.18um technology, at worst-case environment conditions, were used as the reference. The generic gate delay, \( \tau_{delay} \), (based on logical effort [30]) is calculated with equation 3.1:

\[
\tau_{delay} = R_{eff} (C_{self} + C_{load}) = \tau_{int} + R_{eff} C_{load} \tag{3.1}
\]

In our PLA critical path we have a chain of gates whose total delay is:

\[
PLA_{clock\_period} = t_p + \text{margin} + t_E \tag{3.2}
\]

Where \text{margin} is the margin-of-error (10% in our case) added for safety to take into account for timing changes due to process and environment variations. Assuming uniform inverter and \( M_{up/down} \) sizes, we can use the following equations to compute \( t_p \) and \( t_E \):

\[
t_p = 4\tau_{int} + R_1 C_{inv} + R_2 C_{CLK} + \tau_{up/dwn} + R_{up/dwn} C_{AND} + R_3 C_{inv} + R_4 C_{IN\bar{T}R} \tag{3.3}
\]

\[
t_E = 5\tau_{int} + R_1 C_{inv} + R_2 C_{CLK} + R_3 C_{inv} + R_6 C_{CLK\_delay} + \tau_{up/dwn} + R_{up/dwn} C_{OR} + R_7 C_{Out} \tag{3.4}
\]

The capacitive loads are directly related to the IPO numbers of the PLA. The capacitance of the clock lines (\( C_{CLK} \) in the AND-plane and \( C_{CLK\_delay} \) in the OR-plane) are due to
interconnect and $M_{\text{Pull-up}}/M_{\text{Pull-down}}$ gate capacitances. Each product line has one pull-up and one pull-down transistor, so to calculate clock line capacitances for the AND and OR planes we have developed the following 2 equations:

$$C_{\text{CLK}} = numProd \times (C_{\text{inUpDwn}} + Cap_{\text{PerLine}} \times CellWidth) \quad (3.5)$$

$$C_{\text{CLK} \_\text{delay}} = 2 \times Input \times Cap_{\text{PerLine}} \times Cellheight + Outputs \times (C_{\text{inUpDwn}} + Cellheight \times Cap_{\text{PerLine}}) \quad (3.6)$$

We have assumed that the clock line spans all product columns and the clock line for the OR-plane has to cross the AND-plane height. The number of the horizontal rows in the AND-plane is twice the number of inputs, to include the inverted and non-inverted forms. The $C_{\text{AND}}$ is the line capacitance of a product line plus the junction capacitance of the pass-transistors present on its column; it should also include the input capacitance of the inter-plane buffer ($C_{\text{inv}}$). It is computed with the equation (3.7):

$$C_{\text{AND}} = numInputs \times 2 \times Cap_{\text{PerLength}} \times Cellheight + attached \_\text{transistors} \times C_{\text{ds}} + C_{\text{INV}} \quad (3.7)$$

$C_{\text{INTR}}$ is the capacitance on the rest of the product line in the OR-plane plus the gate capacitances of the gates connecting the product line to the outputs. It is summed up in equation (3.8):

$$C_{\text{INTR}} = Outputs \times C_{\text{PerLength}} \times Cellheight + attached \_\text{gates} \times C_{\text{gate}} \quad (3.8)$$

$C_{\text{OR}}$ is the line capacitance of the output line plus the junction capacitance of the pass gates attached to it. It is calculated using the equation (3.9):

$$C_{\text{OR}} = Products \times Cell \_\text{width} \times C_{\text{PerLength}} + C_{\text{INV}} \quad (3.9)$$

\footnote{In Artisan documents, it is referred to as K-load for their pull-up/down power of their gates with unit of: delay time per load capacitance (nano seconds/Pico Farad).}
In the next section we will discuss the effect of line resistance in our calculations. The PLA timing data using SPICE simulations as well as fast manual calculations are presented in the following sections.

### 3.3 Line delay (RC effects)

The RC characteristics of polysilicon and metal lines were computed with the information obtained from TSMC’s documents [26]. We found that the line resistance of the poly lines on average is 24 ohm/micron (24 to 250 ohm for range of 1 to 10 microns) at the nominal width (0.23 microns). The capacitive contribution of these poly lines was in the range of 0.14 to 1.4 fF for the same range of length. The dominant time constant \( \tau \) of a signal going through such a line is shown in this equation: 

\[
\tau = \frac{RC}{2} 
\]

Where \( R \) and \( C \) are resistance and capacitance per unit length and \( L \) is the line length.

For the range of 1 to 10 microns the time constant ranges from 1.7e-27s to 1.8e-23s. Even at 100micron length the time constant is in the order of 5e-20s, which is much less than our period of operation (1ns at 1 GHz clock cycle). The resistance of metal lines were found to be 2 orders of magnitude less than those of the poly line. Hence, as long as the poly lines are kept short and metal lines are used for PLA interconnects we may ignore their resistive effects. This conclusion was also confirmed with simulation of a simple circuit consisting of a pull-up/down transistor cell connected to the equivalent RC load of a 1000um metal line. The signal delay difference between including and not including the resistive effect was less than 5%. Hence, we have excluded the line resistance in our calculations. The line resistance effect becomes significant in large or high-speed PLA designs where placement of product lines with a larger number of input connections near the input lines would improve timing.
3.4 PLA area calculations

The PLA area calculation is based on its IPO numbers and the size of its buffers and pass-transistor tiles which cover the AND and OR planes. In order to make the initial calculation for our PLA sizing independent of technology, we need to use a technology independent parameter such as lambda (\( \lambda \)). It has been used to express the feature sizes in a given technology without specifying exact measurement. Then, by assigning a specific size to \( \lambda \) it could be used to get the exact size of the device.

![Basic NMOS pass-transistor tile dimensions](image)

**Figure 3.5: Basic NMOS pass-transistor tile dimensions**

A typical transistor tile is shown in Figure 3.5 with minimum dimensions based on the design rules which are shown with capital letters A, B, C and E. These values are dictated by fabrication lines’ design rules for each technology. They are listed in Table 3.1 along with their corresponding \( \lambda \) values in the 4th and 5th columns. The minimum width and height of the tile is calculated in the bottom 2 rows and the values from rounded up figures only increase the minimum size by 17% for the 0.35um and 5% for 0.18um

<table>
<thead>
<tr>
<th>Technology Dimension</th>
<th>0.35um</th>
<th>0.18um</th>
<th>0.35 (( \lambda ))</th>
<th>0.18 (( \lambda ))</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.4</td>
<td>0.22</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>B (contact space)</td>
<td>0.4</td>
<td>0.25</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>0.3</td>
<td>0.16</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>E</td>
<td>0.15</td>
<td>0.1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Min-width (um)</td>
<td>2.05</td>
<td>1.14</td>
<td>2.4</td>
<td>1.2</td>
</tr>
<tr>
<td>Min-height (um)</td>
<td>1.05</td>
<td>0.6</td>
<td>1.2</td>
<td>0.6</td>
</tr>
</tbody>
</table>

**Table 3.1 Minimum size tile calculations based on \( \lambda \).**
technology. This is acceptable since designing with minimum size gates would run into many difficulties and is less scaleable. The minimum size tiles could also be used, once the initial design calculations and technology mapping is done with the chosen $\lambda$ values. In this case the minimum widths for 0.35 and 0.18 micron technology correspond to $12\lambda$. We use the following equations to calculate the minimum tile width and height:

$$\text{Tile-Width} = 2 \times A + 2 \times C + 2 \times E + \text{PO}$$
$$\text{Tile-height} = A + 2 \times E$$

(3.10)

The $\text{PO}$ is the minimum width of the polysilicon line, which sets the length of the nMOS device. The minimum height should be set to be the same as the width to have square tiles. This would allow them to be used in both planes and also simplify the buffer placement along the 2 axis. Furthermore, this would allow slight flexibility in the pass-transistor gate size selection without the need to adjust the tile size. For our experiments, the cell dimension of 1.6x1.6 micron was used to also match the cell width of the inverters used from the cell library.

By selecting the buffer and pull-up/down cells to be of the same or some multiple of the tile size, we ensure the efficient placement of the PLA parts. The cell sizes can also be expressed as some multiple of $\lambda$ to keep the initial design calculations simple. This would simplify the task of the CAD tool which makes the initial technology mapping to decide between selecting a PLA or synthesized SC library solution based on their area, delay, and power cost functions. The total area of a given PLA based on its IPO numbers is computed with the following equation:

$$\text{PLA_Area} = \text{area}_{\text{P}_{\text{up}, \text{down}}} \times (\text{products} + \text{Outputs}) + (2 \times \text{Inputs} + \text{Outputs} + 2 \times \text{products}) \times \text{Inv\_area} + (2 \times \text{Inputs} + \text{Outputs}) \times \text{products} \times \text{Tile\_area} + \text{Over\_head}$$

(3.11)
The first part of the equation calculates the area of pull-up/down cells, the second part sums up the buffer area and the third part adds the pass-transistor tiles used in AND/OR planes. The final term is the over-head associated with the placement of the three parts together. In the next section we will discuss the procedure used for generating the SC-ASIC for the benchmark circuits, to compare with their PLA implementations.

### 3.5 Standard Cell flow and data collection

The design flow used in generating the SC-ASIC solution for the benchmark circuits is outlined in Figure 3.6. First the benchmark circuits are optimized with Espresso [27] to use minimum terms and ensure they have correct syntax. This step is common to both PLA and SC-ASIC flow, to ensure both start from a common RTL description.

Each benchmark circuit was then compiled by Synopsys dc_shell™ using the information from Artisan Cell library for TSMC 0.18 technology (single poly 6 metal layers) [28]. A timing constraint of 2ns was placed on all input to output paths and timing evaluations were done based on the worst-case timing information. The timing constraint was
placed to ensure the tool does not use minimum size gates and yet it stays within the timing capacity of this library. After the gate synthesis, a Verilog gate-level netlist was produced and was passed to Cadence SOC Encounter™ for placement and routing. Because the circuits are all combinational, routing congestion was not expected and the placement was setup as a sea-of-gates (i.e., no extra space for routing channels was left between the cell rows) with all 6 metal layers available for routing. This was done to ensure that the SC-ASIC flow could reach its maximum density potential.

Following placement and routing, interconnect parasitic resistances and capacitances (RC) were extracted. The gate-level netlist and the RC parasitic information were passed to Synopsys Prime-Time™ for static timing analysis to find the critical path among all input to output paths. All synthesized benchmark circuits met or exceeded the 2ns timing constraint. We did not consider any overhead due to power stripes and/or I/O planning and as a result the area reported for the SC-ASIC layouts are the same as their basic gate area. The area/delay information for the SC-ASIC and PLA solutions is discussed in the next section.

3.6 PLA vs. SC-ASIC

For the PLA implementations, inverter intrinsic delay and \( R_{\text{eff}} \) were taken from the cell library reference document [28], and computed for worst case environment conditions (Temperature = 125°C and Vdd = 1.62V). This is to ensure equivalent environment conditions to the delay measurements done in the SC-ASIC flow.

3.6.1 SPICE verification of Delay calculations

To verify the accuracy of the calculation method used for estimating PLA delays, the capacitive loads were generated for all benchmarks and SPICE simulation results were
tabulated for pre-charge and evaluation times for each PLA. The total delay of the PLA was considered the sum of the two values. To compare the results, the ratio of delays measured by SPICE simulation over the quick calculation method is shown in Figure 3.7. The data is sorted for the increasing number of the PLAs' product terms.

For all circuits with less than 200 product terms, our calculation method underestimates the delay by about 10% or less. For circuits with more than 200 product terms, our method overestimates the delay by less than 10%. This is due to the nonlinear nature of inverter buffers whose delay we are trying to estimate with linear functions such as RC delay calculation. Since we are interested in matching or exceeding SC-ASIC delays, we will be limiting the number of PLA product terms to less than 300. Furthermore, increasing our calculated delays by 10% will give us the upper bound necessary to estimate our PLA timing without the need to run a full SPICE simulation.

3.6.2 Delay Comparison

The PLA critical path delay was calculated by summing up the delays contributing to pre-
charge and evaluation times as was shown in Figure 3.4 waveforms. A Perl script was used to read in the netlist in "pla" format, calculate the capacitive loads due to PLA's IPO numbers and add the gate or junction capacitances for the number of connections necessary for each product line or output line. Finally, the RC delays for each input to output path were calculated and reported. The path with longest delay \((Total\_delay = t_p + margin + t_E)\) was chosen as the critical path.

Figure 3.8 shows the graph of PLA delays vs. their number of product terms\(^3\). It is a good indicator of how closely the PLA delay follows the number of product terms. A correlation factor of 95% was found between PLA critical path delay and its number of product terms. In other words, optimizing a netlist for PLA timing should reduce its total number of product terms. Optimizing for a SC-ASIC involves reducing the number of slow

\[\text{Figure 3.8: PLA circuit delay data sorted with respect to their number of product terms.}\]

\(^3\) Note that every 5-6\(^{th}\) circuit name is printed.
gates in its critical path that could involve logic conversions on faster gates.

The relation between PLA delay and its number of product terms is also confirmed in the graph of ratios between PLA delays and SC-ASIC delays shown in Figure 3.9. As the graph indicates, for about 30% of the circuits, their PLA implementation was as fast or faster than their SC-ASIC one (i.e., the ratio was equal to or less than 1.0). Furthermore, the circuits with significantly longer PLA delays tend to have more than 100 product terms. A few of the circuits with small product terms did not follow this trend since their SC-ASIC implementations were much faster (2 to 3 times). We could conclude that by keeping the number of product terms less than 100, we could at least match the delay time of the SC-ASIC circuits with a conventional clock-delayed PLA structure. If we were to use faster PLA structures, as reported by recent papers [12], we could pack up to 200 or more product terms in a PLA and still match or exceed the SC-ASIC timing.

Table 3.2 reports the correlation coefficients between the PLA critical path’s capacitive delay ratio and the number of products.
loads and the PLA’s IPO numbers. As the number of product terms increase, the length of the input lines and possible number of gates attached to them increase, which gives rise to $C_{in2}$ which is seen by the input inverter. The number of product terms also sets the length of the clock line and the number of pull-up/down gates attached to it, which contribute to $C_{clk}$. It also sets the length of the output line in the OR plane which is the main contributor to $C_{OR}$. The number of inputs sets the height of the AND-plane which is also the length of the product line. $C_{AND}$ includes this line capacitance and the junction capacitance of the pass-transistors attached to it. The number of outputs sets the number of pull-up/down gates in the OR-plane, which are the main contributors to $C_{clk\_delay}$. It also sets the height of the OR-plane, which sets the length of the product line in the OR-plane. This line and the gate capacitances attached to it are the main contributors to $C_{intr}$.

<table>
<thead>
<tr>
<th></th>
<th>$C_{in2}$</th>
<th>$C_{AND}$</th>
<th>$C_{intr}$</th>
<th>$C_{OR}$</th>
<th>$C_{clk}$</th>
<th>$C_{clk_delay}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>98%</td>
<td>14%</td>
<td>15%</td>
<td>93%</td>
<td>100%</td>
<td>17%</td>
</tr>
<tr>
<td>Input</td>
<td>9%</td>
<td>96%</td>
<td>15%</td>
<td>4%</td>
<td>9%</td>
<td>39%</td>
</tr>
<tr>
<td>Output</td>
<td>19%</td>
<td>47%</td>
<td>81%</td>
<td>14%</td>
<td>17%</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 3.2: Correlation relation between IPO and Capacitive loads

For most circuits, the number of product terms is dominant (much larger than their I/O numbers). As a result the clock capacitance is the largest and contributes the most of the critical path delay.

3.6.3 Verification of area calculations for the PLA structure used

In order to verify our PLA area estimates, we used Berkley’s MPLA which produces a PLA layout from a given netlist in “.pla” format. The tool uses MOSIS scalable SCMOS technology with $\lambda=1\mu m$ and produces static PLA structures with NMOS pass gates. By scaling it for $\lambda=0.1\mu m$ we could obtain approximate layout sizes for 0.18um technology. The
Table 3.3: Sample PLA layout size scaled from $\lambda=1\mu m$ to $\lambda=0.1\mu m$

area of the PLA fabric produced by MPLA for a 5-8-4 PLA, excluding the I/O buffers, is shown in Table 3.3. MPLA uses tile sizes of about $16.5\lambda \times 10\lambda$. This is close to $1.6\mu m \times 1\mu m$ in $0.18\mu m$ technology. It also uses the diffusion layer for ground line, which reduces the dimensions of the AND/OR-planes. As shown in Table 3.3, our PLA fabric with minimum size would be close to the MPLA's fabric. Furthermore, with minimum size tiles, we would be able to reduce our AND/OR-plane sizes by a factor of 3 (i.e. going from $205\mu m^2$ to $70\mu m^2$). The sample layout used for this measurement is shown in Figure 3.10.
3.6.4 Area Comparison

The PLA area was estimated by using equation (3.10), with the same tile and buffer sizes used for the delay computation. The PLA area was found to have a strong correlation of 94% with the number of product terms. However, it is not trivial to predict the size of SC-ASIC circuit from the netlist or the IPO number. The ratio between PLA area and SC-ASIC layouts was found to range from 1.5 to more than 70, with 74% lying below 10 times (see Figure 3.11). The relative spread of PLA over ASIC area ratios is shown in Figure 3.12. About 80% of the circuits with less than 100 product terms have PLA/SC-ASIC area ratio of less than 5. Clearly, the PLA implementation of a circuit consumes more silicon area and it must be considered during design partitioning. However, our tiles were not minimum size and PLA area improvement was not the goal of this project. If we use minimum size tiles (see Table 3.1), the size of the main PLA fabric could be cut-down by a factor of 3. As a result, the percentage of PLAs with area ratio of less than 4 will rise to 74% (see Figure 3.12).

Figure 3.11: PLA area / SC-ASIC area vs. number of product terms
3.7 Computing Power Factor for PLA

The dynamic power of a circuit is calculated with the equation: \( P = \alpha \Sigma C V_{DD}^2 f \) [35]. If we consider the supply voltage \( V_{DD} \) and frequency \( f \) constant, then the dynamic power only depends on the sum of the circuit’s total capacitances. In most circuits, however, not all parts of the circuit are activated at each clock cycle. In other words, not all its capacitive loads are charged and discharged. To account for this, a unit-less factor \( \alpha \) (activity or power factor) is added to the equation. In order to compute \( \alpha \) we need to measure the average power for a large random set of input to the circuit over a long period of time. Then we divide the power by the sum of the total capacitive loads, square of power supply voltage and the frequency to obtain \( \alpha \).

We generated a layout for a PLA example with 5-8-4 IPO number (see Figure 3.10), and extracted all its capacitive loads. The resulting netlist was simulated with spice for a set of 350 randomly generated input vectors. Its capacitive loads and average power consumption at 400Mhz clock frequency are tabulated in Table 3.4, and an \( \alpha \) factor of 86% was
calculated. The result for larger PLAs will be lower, since their product lines could be randomly populated by pass-transistor tiles. As a result, the portion of capacitive loads which are switching will be less.

<table>
<thead>
<tr>
<th></th>
<th>Σcaps (fF)</th>
<th>Avg Power (mW)</th>
<th>V\text{DD} (V)</th>
<th>Frequency (MHz)</th>
<th>α</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>1009</td>
<td>1.12</td>
<td>1.8</td>
<td>400</td>
<td>86%</td>
</tr>
<tr>
<td>Non-Clock</td>
<td>645</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>80%</td>
</tr>
<tr>
<td>Clock</td>
<td>363</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>100%</td>
</tr>
</tbody>
</table>

Table 3.4: Total capacitance and calculated α for a 5-8-4 PLA.

The activity factor along the clock lines is one (i.e., they are always switching). For the rest of the circuit, the activity factor is close to 0.8. Close to 50% of capacitive loads are due to buffers which are mostly off the clock lines. We found that only 36% of loads on the fabric are connected to the clock lines. Based on the variation of IPO numbers in our benchmark circuits, a range of 80% to 90% is expected for α. This result also matches other reports of activity factor for clock delayed PLAs [12].

Although, the power measurement will differ for other PLA architecture and/or fabrication technology, the same method could be applied to calculate the α factor. Hence, the PLA dynamic power could be estimated without the need for a full spice simulation.
CHAPTER 4 TECHNIQUES FOR PLA AREA-DELAY-POWER IMPROVEMENT

In this chapter, we will present a few techniques to improve PLA area, delay and power. First, we will apply the concept of logical effort (LE) to our PLA structure to improve its delay. Then we show a heuristic approach to LE gate sizing to optimize the power-delay product (PDP). Next, we will investigate partitioning a circuit to improve its PLA delay, and its effect on area and power.

4.1 Using Logical Effort for PLA delay improvement

So far in working with our PLA structure, we have considered our tile and gate sizes fixed. In order to improve the PLA critical path, we still need to consider gate sizing where the timing bottle-necks occur. Normally, this would require many iterations of gate sizing and simulating with SPICE to arrive at an optimum size/delay combination. The PLA critical paths, which contribute to the pre-charge and evaluation phase timing, are shown in Figure 4.1. It is essentially a chain of inverters with the main capacitive loads as their side loads. We applied the concept of logical effort (LE) [29,30] to the PLA’s critical path to explore a fast and yet accurate way of gate sizing for this structure. Since the method of LE is independent of technology, it has been used for optimization of complex CMOS circuits [31,32,33]. However, it has not been applied to PLAs as yet. This method will determine the fastest possible speed of the PLA. In using this method, there is an assumption that a library of near continuous gate sizes exist. This is achievable since already there are numerous gate sizes in current cell libraries and for flexible PLA design we will need access to a range of inverter and pull-up/down gate sizes that match our PLA tile sizes.
4.1.1 Basic Logical Effort Optimization

The two main critical paths are shown in Figure 4.1. The pre-charge path has 7 stages, 2 of which are the pull-up/down gates in the AND and OR-planes. During the pre-charge stage, only the pull-up transistor is on and we are only concerned with its rise time. During the evaluation, we can assume that the input signal has already arrived (input timing requirement) and we are only concerned with the pull-down transistors' fall time. As a result, for our timing evaluations, the pull-up/down gates behave like inverters. Furthermore, the pre-charge path is interrupted after the node $n5$. The rest of this path's timing depends on the arrival of the OR-plane's delayed clock ($\Phi1\_delay$). The timing of the delayed clock is set along the evaluation path. Notice that during the evaluation period if the input is low, nodes $n3$ and $n5$ will remain high and will not delay the path. Hence, we can concentrate on the 5 stages in the pre-charge path and the 4 stages in the evaluation path. The logical effort path optimization starts with the following equations:

$$Path\_Effort = \prod (LE \times BE \times FO) = \prod (LE \times BE) \times \frac{C_{load}}{C_{in}}$$ (4.1)

$$SE^* = (Path\_Effort)^{1/N}$$ (4.2)

$$C_{in} = LE \times \frac{C_{out}}{SE^*}$$ (4.3)
A gate's LE is its ability to drive current as compared to an inverter with the same input capacitance as the gate. Hence, the LE of a balanced inverter is 1. The branching effort (BE) is the path effort of side branches connected to the main path we are analyzing. In our case, BE is also equal to 1. The electrical effort or fan-out (FO) is the ratio of the load capacitance to the input capacitance ($C_{out}/C_{in}$). Equations (4.1) to (4.3), help to find the optimum gate size for each stage. The path effort is computed by the ratio of the output capacitance of the last stage ($C_{out}$) over the input capacitance of the initial stage ($C_{in}$). The optimized stage effort $SE^*$ is computed as the Nth root of the path effort (equation 4.2), from which the $C_{in}$ of the current stage is calculated from equation (4.3).

The following experiment was carried out to evaluate the effect of LE optimization as compared to using the fixed size gates along the PLA critical paths. First, the capacitive loads for the PLA critical path, as shown in Figure 3.3, of a few benchmark circuits with X4 inverters and fixed tiles were calculated. They were used as parameters to a SPICE netlist, and simulated to get the pre-charge and evaluation time for each circuit. Then, the same line loads were used as side loads to compute the optimized gate sizes using LE.

A sample spreadsheet result for the pre-charge path gate sizes is shown in Table 4.1. The $C_{out}$ at stage 5 (node n5) is $C_{intr}$ plus the gate capacitance of the pass-transistor in the OR-plane (see Figure 4.1). The optimized stage effort $SE^*$ is calculated in the 3rd column and $C_{in}$ for each stage is calculated as $C_{out}/SE^*$ and shown in the 4th column. In other words, at every stage, after calculating the new $C_{in}$, the load to the previous stage becomes the sum of previous stage's side-load plus the $C_{in}$ of the current stage. The device size at stage 3 (pull-up/down gates) also sets the $C_{clk}$ side-load. It is calculated in the 2nd row of the 6th column.
based on the circuit's number of product terms and the input capacitance of the stage 3 gate. The \( C_{\text{clk}} \) side load is then used as the \( C_{\text{out}} \) for the remaining 2 stage inverters.

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Input</th>
<th>Output</th>
<th>Product</th>
<th>( C_{\text{clk}} ) (fF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z5xp1</td>
<td>7</td>
<td>10</td>
<td>65</td>
<td>1326.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stage</th>
<th>( \text{SE}^* ) (FO)</th>
<th>( C_{\text{in}} ) (fF)</th>
<th>( C_{\text{out}} ) (fF)</th>
<th>( W_{p} ) (um)</th>
<th>( W_{n} ) (um)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1.07</td>
<td>14.25</td>
<td>15.2</td>
<td>5.03</td>
<td>2.10</td>
</tr>
<tr>
<td>4</td>
<td>1.07</td>
<td>13.36</td>
<td>14.25</td>
<td>4.71</td>
<td>1.96</td>
</tr>
<tr>
<td>3</td>
<td>1.35</td>
<td>20.09</td>
<td>27.16</td>
<td>7.09</td>
<td>2.95</td>
</tr>
<tr>
<td>2</td>
<td>10.98</td>
<td>120.81</td>
<td>1326.82</td>
<td>42.64</td>
<td>17.77</td>
</tr>
<tr>
<td>1</td>
<td>10.98</td>
<td>11.00</td>
<td>120.81</td>
<td>3.88</td>
<td>1.62</td>
</tr>
</tbody>
</table>

Table 4.1: Five stage gate optimization with LE

Based on the gate's input capacitance, its P/N device widths, \( W_{p} \) and \( W_{n} \), are calculated with the ratio of 2.4 to 1 for balanced rise and fall times. The input capacitance to device width conversion is technology dependent and for 0.18um technology a 2fF/um factor is used. Using the above gate sizing, an updated SPICE netlist for the critical path was used to obtain the new critical path delay. The results are show in Table 4.3 (under basic LE), which indicate significant timing improvement over the fixed size gate PLA, at the cost of area.

4.1.2 Modified Logical Effort Optimization

The basic LE method requires the fan-out \( (C_{\text{out}}/C_{\text{in}}) \) ratio to be larger than 1. When this ratio is 1, it sets all the gates to the same size. In our path, this causes the inverters in stage 4 to have a larger input capacitance than they should have. This increases the side load \( C_{\text{clk}} \) in stage 3 to 2.5 times its original value and causes the optimized gate size for the 2\(^{nd}\) stage inverter to be unrealistically large (i.e., FO of 11).
<table>
<thead>
<tr>
<th>Circuit</th>
<th>Input</th>
<th>Output</th>
<th>Product</th>
<th>C_clk (fF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z5xp1</td>
<td>7</td>
<td>10</td>
<td>65</td>
<td>990</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Stage</th>
<th>SE*</th>
<th>Cin</th>
<th>Cout</th>
<th>Wp (um)</th>
<th>Wn (um)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1.07</td>
<td>14.25</td>
<td>15.2</td>
<td>5.03</td>
<td>2.10</td>
</tr>
<tr>
<td>4</td>
<td>4.00</td>
<td>3.56</td>
<td>14.25</td>
<td>1.26</td>
<td>0.52</td>
</tr>
<tr>
<td>3</td>
<td>1.16</td>
<td>14.91</td>
<td>17.36</td>
<td>5.26</td>
<td>2.19</td>
</tr>
<tr>
<td>2</td>
<td>9.49</td>
<td>104.36</td>
<td>990.07</td>
<td>36.83</td>
<td>15.35</td>
</tr>
<tr>
<td>1</td>
<td>9.49</td>
<td>11.00</td>
<td>104.36</td>
<td>3.88</td>
<td>1.62</td>
</tr>
</tbody>
</table>

Table 4.2: Five stage gate optimization with modified LE

To prevent this, a variation of LE must be used to ensure when the $SE^*$ is close to 1, an optimized ratio such as FO4 (4/1) is used between the stage 4 and 5 inverters. This result was used to calculate gate sizes reported in Table 4.2. The 2nd method reduces the $C_{clk}$'s capacitive load by 25% and overall gate sizes by 18%, which are significant reduction in power and area. The SPICE simulation results are also shown in column 4 of Table 4.3, and they are very close to the basic LE.

<table>
<thead>
<tr>
<th>Circuit Timing</th>
<th>x4 with no LE (ns)</th>
<th>basic LE (ns)</th>
<th>2nd Method LE (ns)</th>
<th>% difference basic - 2nd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z5xp1 Pre-charge</td>
<td>0.83</td>
<td>0.46</td>
<td>0.47</td>
<td>45% - 43%</td>
</tr>
<tr>
<td>Evaluation</td>
<td>0.88</td>
<td>0.40</td>
<td>0.39</td>
<td>55% - 56%</td>
</tr>
</tbody>
</table>

Table 4.3: Timing comparison between fixed and optimized gate PLAs

4.1.3 Delay estimation with Logical Effort

Total delay of the path is computed by equation 4.4, which is the sum of each gate’s stage effort plus its parasitic or self capacitance. For an inverter $P = 0.5$.

$$D = \sum (LE \times FO + P)$$ (4.4)

The delay in this case is technology independent and has to be multiplied by the delay of the minimum size inverter ($\tau_{inv}$) to give the real delay for a given technology. For 0.18um technology $\tau_{inv}$ is close to 15ps. The delay calculated with logical effort for our 2 examples have a difference of 9-10% from the SPICE simulation results. This is expected as LE does
not account for slow rise/fall times, and as reported in [34] could differ as much as 30% from SPICE due to variation in input transition times. This will adversely affect the size and partitioning decisions based on LE when the input transition times vary from one node to the next (specially when some have larger loads).

Fortunately, it is possible to extend LE to include the input transition time effect. In [34] authors suggest an extension to LE (XLE) delay calculation as $D = t_0 + RC_{load} + K_{tran}t_{tran}$ where $t_0$ is the gate’s intrinsic delay, $t_{tran}$ is the 10-90% input transition time and $K_{tran}$ is a model dependent dimensionless coefficient. Another equation is used to model the output transition time: $t_{tran} = R_tC_{load} + t_{trans}$. Here, $R_t$ and $t_{trans}$ are model dependent coefficients. Similar to LE, an initial spice simulation on a set of unit size gates and linear regression is required to drive the model coefficients. With this method, the delay calculations will be closer to static timing analysis. They will be much faster than SPICE simulation with close accuracy.

4.1.4 Considering Power with Logical Effort

LE has only been used for delay estimation and gate sizing to optimize for delay. However, in recent years, it is power reduction that is the focus of the designers. We suggest a method that improves the power-delay product (PDP) based on LE. We introduce a smaller loading (i.e., reduce FO).

The dynamic power consumed by CMOS gates is mostly attributed to the charging and discharging of their capacitive loads. This energy is computed by the equation: $E = \frac{1}{2}C_LV_{DD}^2$. At every clock cycle the capacitive loads are both charging and discharging, so the total energy consumed is: $E = C_LV_{DD}^2$ [35]. And power is shown as the energy consumed per unit
of time, hence, the frequency of switching (charging/discharging) is added to the equation: $P = C_L V_{DD} f$ [35]. The power-delay product (PDP) is used as a measure to optimize circuit performance based on energy used and delay. Since the delay of a circuit normally sets its frequency of operation, $f = \frac{1}{2} \text{Delay}$, it can be shown with the equation 4.5:

$$PDP = C V_{DD}^2$$

(4.5)

Considering a constant power supply voltage $V_{DD}$, PDP is only a function of the capacitive loads in the circuit. For optimization in LE, the FO is selected based on the graph of delay vs. FO as shown in Figure 4.2. Plotting PDP on the same graph, we notice that by reducing our FO factor from 2.2 to 1.8, we could save 21% in PDP (energy consumption) while spending 8% in delay (see Figure 4.2). In other words, we could trade minor loss in delay for a more energy efficient circuit.

![Figure 4.2: Graph of normalized PDP and Delay vs. FO factor.](image)

reducing our FO factor from 2.2 to 1.8, we could save 21% in PDP (energy consumption) while spending 8% in delay (see Figure 4.2). In other words, we could trade minor loss in delay for a more energy efficient circuit.
Since the optimal solution of LE for delay and LE for the power-delay product (PDP) are different, we seek a new method to directly obtain the optimal sizing for PDP. We first remove the actual load and replace it with a smaller load. Then, if we compute the LE solution, the sizes will be smaller and the delays will be equal in all stages except the last one. The approach is best illustrated using an example and we use an inverter chain for this purpose. Consider the gate sizes for the following logic circuit to produce the minimum delay. Assuming that the output is 64 and we optimize for delay, then the $SE^* = 4$ and the sizes increase as shown below:

![Figure 4.3: Sizing for a load of 64](image)

The delay, total capacitance and PDP are given by:

\[
D = 3(SE^*) + 3(R) = 3(4) + 3(1/2) = 13.5 \\
C = 1 + SE^* + (SE^*)^2 = 1 + 4 + 16 = 21 \\
PDP = CxD = 21 \times 13.5 = 283.5
\]

However, if we try to reduce capacitance, and as a result PDP, by using $SE^* = 2$, then we get:

![Figure 4.4: Lower size gates with the original Cout](image)
The delay of this case is different. The first two stages have the same delay as before but the third stage has a longer delay because of the weaker inverter as follows:

\[ D = 2(2) + 2(1/2) + ((Cout/Cin)/(SE^*)N-1) + (1/2) \]
\[ = 2(2) + 2(1/2) + ((64)/(2^2)) + (1/2) = 21.5 \]

\[ C = 1 + 2 + 4 = 7 \]

\[ PDP = C \times D = 150.5 \]

While this result demonstrates that a lower PDP can be obtained, the question remains as to the optimal solution to the PDP. The general solution would proceed as follows:

\[ D = (N-1)(SE^* + P) + \frac{(C_{out}/C_{in})}{(SE^*)^{N-1}} + P \]
\[ C = 1 + \frac{SE^*}{LE_1} + \frac{(SE^*)^2}{LE_1LE_2} + \ldots + \frac{(SE^*)^{N-1}}{LE_1\ldots LE_{N-1}} \quad (4.4) \]

\[ PDP_{LE} = C \times D \]
\[ \frac{\partial PDP_{LE}}{\partial SE^*} = 0 \]

The solution to the last equation would produce the optimal SE* for PDP. While this is only a preliminary result, it points the way to a LE solution that optimizes energy rather than delay. Many extensions are needed for gates other than inverters, as well as the inclusion of sideloads and branching factors.

### 4.2 Using Partitioning for PLA delay Improvement

Following the basic area/delay comparisons between PLA and SC-ASIC, we need to investigate methods to improve PLA solutions where applied to large combinational blocks. As we discussed in Chapter 3, for most well-populated PLAs, the number of product terms sets the minimum timing requirement. Hence reducing the number of product terms in a PLA is the first step in improving its speed. It is possible to partition a given circuit to separate its
input/output/product terms. However, as shown in Figure 4.5a, this create feedback loops between the resulting PLAs, which is undesirable.

One way to remove such problems is to start from each required output and include all fanins back to the primary inputs. Conceptually, this is shown in Figure 4.5b. The right to left funnel, starts from the output and includes all its product and input terms. Our goal is to reduce the PLA delay by partitioning its outputs and generating a number of independent PLAs.

4.2.1 Minimum Delay PLA (single output)

Recall that more than two-thirds of the benchmark circuits had a PLA delay higher than their SC-ASIC implementation. We first partitioned these circuits to single output PLAs and measured their critical path delays individually. The percentage difference of the critical path delay of the full PLA implementation of each circuit and its slowest single output PLA is plotted in Figure 4.6.

In each circuit, the slowest outputs typically depend on the largest number of the product terms. In circuits which one output is dependent on most of the product terms, the slowest
Figure 4.6: Percentage difference of delay between full and single-output PLA implementations.

Single output PLA is not much faster than the full circuit. Hence, these are the circuits with less than 20% difference between their slowest single output PLA and their full PLA implementation. On the other hand, for the circuits in which the outputs are dependent on a balanced number of product terms, their single output PLA implementation has only 20% delay of the full PLA. Therefore, by partitioning this circuit, we could run it at 5 times its full PLA speed. On average, the slowest single output PLA has 44% delay margin compared to the full PLA implementation of the benchmark circuit. Figure 4.7 shows the percentage of the circuits that fall under each category. In 8% of the circuits, the difference between their full PLA delay and the slowest single-output PLA was less than 20%. Other 33% had a difference of 20% to 40%. For the rest, about 59%, the difference ranges from 40% to 81%.

Single output partitioning, however, is very area intensive and for outputs with only a few product terms, it is very inefficient. Hence, the outputs should be combined when they share input and product terms, until we reach our timing ceiling.
4.2.2 Optimizing with Multiple Output PLA

As it was shown in section 3.6.4, there is a strong correlation between the number of product terms in a PLA and its critical path delay. To reduce the total size of the PLAs generated to implement a circuit but still meet the timing constraint, we need to distribute the product terms among the partitioned circuits. We used the following heuristic for the slowest outputs:

1. list the outputs which add the least number of new product terms
2. then the outputs which add the least number of new inputs

Notice that we are not considering the common product terms, as two outputs could have common product terms and yet their number of uncommon product terms could be even larger. The outputs at the bottom of the initial list normally are the 2nd slowest, since they have the 2nd largest number of product terms. Furthermore, they do not have much in common with the first one. These 2 slow outputs (with the largest product terms,) provide us with the starting seeds. By assigning the outputs on each list to the corresponding slow output, we can form PLAs with a balanced number of product terms. We plotted the area and
delay change for PLA partitions of a sample circuit in Figure 4.8. The values are normalized with respect to the full PLA implementation of the circuit. It is clear that in general by packing more outputs, we tend to reduce the overall area while increasing the delay. Since capacitive loads also increase with area, we need to make a choice between area/power vs. delay. The effect of area increase, however, is reduced for the circuits with a much larger number of product terms than input/output.

From the benchmark circuits, we chose three examples for partitioning, which had large product terms and long delays. We also included the result of partitioning a circuit with long delay, but significantly smaller number of product terms (b10). The results are tabulated in Table 4.4. The total delay, input/product term numbers and area for the full PLA are also listed after each two partitions. The percentage difference between the delay of the partitioned PLAs and their full version is listed in the 6\textsuperscript{th} column. The comparison shows that the delays follow the partitions' number of product terms.
The area differences listed in the 7th column indicate that for the 1st and 3rd circuits, the partitioned PLAs take close to the same area as the original PLAs. The partitions for the 2nd and last circuit (rd84 and b10), however, take about 8-10% more since their combined numbers of products are 15-32% larger than the original circuit. It should be noted that the area calculation does not take into account the overhead of PLA placement. Assuming that best PLA layout practices are used, this overhead could be much less than 10% of the PLA sizes [36].

In conclusion, by partitioning a circuit to smaller PLAs, we gain in timing and do not pay much in area. Furthermore, since the area has not increased significantly (close to 10%), the total capacitive loads will also remain the same and as a result the total dynamic power will not increase significantly.

The output packing to partition a circuit to more than 2 PLAs, however, requires a more rigorous algorithm and it is a partitioning problem, which gets more difficult to solve as the number of inputs and outputs increase. It is an NP-hard problem [37] and we need to use heuristics to simplify the search in the solution space for the optimal solution.
CHAPTER 5  CONCLUSIONS AND FUTURE DIRECTIONS

The trend in DSM fabrication technology favours structured architectures. However, standard ASIC generates random gate and interconnect patterns that makes their fabrication with the new DSM technologies very difficult. In order to consider structured logic arrays as an alternative to SC-ASIC, they have to be competitive in timing, power, and area. We considered a number of structured architectures and found structured ASIC (SA) and PLA to meet the uniformity requirement and have the potential to compete with SC-ASIC. Since SA leaves a few top metal layer for user configuration, it reduces the level of interconnect uniformity. The SA designs are on average 20% slower and use 3 times more power than the SC-ASIC designs. Hence, we focused our research on PLAs as a structured architecture that could fill this gap or improve on SC-ASIC designs.

In Chapter 3, we presented the methods used to compare PLA and SC-ASIC architectures. First, for a set of benchmark circuits, we generated their SC-ASIC layouts and measured their gate area and delay. Then, based on the clocked-delayed PLA structure, we developed some formulas to calculate PLA delay and area for the same benchmark circuits. The delay and area results for both sets of experiments were tabulated. The results indicated that:

1. The PLA delay has a close correlation to its number of product terms
2. The SC-ASIC flow was able to achieve 2ns timing constraint or reduce it for a number of smaller circuits
3. For about 30% of the benchmark circuits their PLA delay was equal or lower than their SC-ASIC implementation
4. The circuits with slower PLAs tend to have more than 100 product terms
It was also observed that there are faster PLA designs than the conventional clocked-delayed PLA, which could allow circuits with up to 200 product terms to match their SC-ASIC implementation in PLA. The correlation between PLA’s IPO numbers and the side loads in its critical path was also presented.

In order to verify the accuracy of our PLA delay estimations, SPICE simulations were done on the netlist of the PLA critical paths with the capacitive loads associated with the benchmark circuits. The results indicated that our calculation methods’ results are about 10% lower than the SPICE Results for most of the circuits. For circuits with more than 300 product terms, the results are closer or somewhat higher. This is because of the linearity of our calculations and nonlinearity of the buffers, whose response changes with the slower rise/fall times.

The area comparison showed that the PLA implementations with similar delay take a lot more area than SC-ASIC. A strong correlation of 94% was found between PLA area and its number of product terms. And for 80% of the circuits with less than 100 product terms, their PLA/SC-ASIC area ratio was found to be less than 5. The area estimation was also verified with the result of PLA layout generator tool “MPLA” and scaling the result to match 0.18 micron CMOS technology. We found that the PLA implementation of circuits with less than 200 product terms could be faster than their SC-ASIC counterparts. However, PLAs tend to take much larger area. Finally, we showed that the dynamic power consumption in a PLA circuit depends on its total capacitive loads. We also found that the activity factor is relatively high (α = 0.8–0.9). By calculating the α factor for each technology and PLA architecture, we could measure its power consumption without the need for a full SPICE simulation.
In Chapter 4, we argued that logical effort (LE) could be used for delay calculation and gate optimization of the PLA critical path. We showed that, using LE gate sizing, a delay improvement of 45% to 55% is achievable. Furthermore, the power-delay product (PDP) of an inverter chain was examined to show that the gate optimization could be adjusted in order to reduce overall energy consumption of the circuit. We showed in one example that at the cost of 8% delay in gate sizing, we could improve PDP factor by 21%.

We also presented methods to partition a slow circuit into 2 or more PLAs, such that each would be independent and meet the timing constraint. It was shown on sample circuits that outputs could be packed based on their number of product terms. It was concluded that this is only useful if one output does not depend on most of the product terms. In case of the balanced partitions, it was shown that we could divide the PLA delay by 2 when we cut its number of product terms by half. It was also shown that the area overhead of partitioning the PLAs with large number of product terms is less than 10%.

5.1 Future Work

PLAs look promising as a future structured logic fabric. The next step is to study PLA area reduction techniques. By reducing area, hence total capacitance, we will also reduce the PLA's total power consumption. Reducing total power consumption of PLAs is important to make them a viable replacement for SC-ASIC. Furthermore, with access to a PLA layout in a specific technology, power analysis could be performed to investigate its power performance as compared to other architectures.

Although PLAs have been studied as LUTs for configurable fabrics [17] and as tiles for structured ASIC [21], they have not been studied as stand alone fabric in parallel with structured ASIC. They are designed to be uniform and may only need the first 2 metal layers.
However, we did not look into ways to ensure uniformity at global routing between PLA blocks. It is a challenge that requires further study. Also, we did not look into PLA design improvement. There has been some recent work for improving power and delay in clocked PLAs [11,38]. They require implementation to use for comparison with other architectures.

5.2 Contributions

In this work we were able to:

- Created a flow to generate capacitive loads and calculate critical path delays for clock-delayed PLA structures. The flow is supported by Perl scripts and well documented for adjustments to match different technologies.

- Generated area delay information for the SC-ASIC implementation of the benchmark circuits and created a flow that could be used for future comparative studies.

- Investigated power analysis for PLA structure and a method to calculate the activity factor for a given technology and PLA architecture. We found it to range between 0.8 to 0.9 for our example

The summary of this research’s contributions is as follows:

1. Presented a method to conduct LE analysis on PLA critical path and to do gate sizing with side-load optimization. Gate sizing could improve PLA critical path delay by as much as 55% at expense of extra area and power.

2. Discussed a method to improve energy efficiency when using LE gate sizing. While gate sizing, we can limit our gain in delay in order to achieve energy efficiency in our circuit.
3. Investigated partitioning methods to subdivide slow PLAs to smaller and faster ones and showed their relative delay/area cost. Partitioning a circuit with a large number of product terms according to its outputs could improve delay with little or no area and power penalty.
REFERENCES


[24] Berkeley PLA Test Set, January 30, 1988; included under espresso-examples directory with the SIS package.


[26] TSMC 0.18UM LOGIC 1P6M SALICIDE 1.8V/3.3V SPICE MODELS.

[27] Espresso: Berkeley logic level optimizer included with SIS package.


