Analysis of logic block architectures and functional improvement of fine grained cells

Rohith Ramnath

University of Nevada, Las Vegas

Follow this and additional works at: https://digitalscholarship.unlv.edu/rtds

Repository Citation
https://digitalscholarship.unlv.edu/rtds/1849
ANALYSIS OF LOGIC BLOCK ARCHITECTURES AND FUNCTIONAL
IMPROVEMENT OF FINE GRAINED CELLS

by

Rohith Ramnath

Bachelor of Engineering
University of Mysore, India
2001

A thesis submitted in partial fulfillment of the requirement for the

Master of Science Degree in Electrical Engineering
Department of Electrical and Computer Engineering
Howard R. Hughes College of Engineering

Graduate College
University of Nevada, Las Vegas
August 2005
UMI Number: 1429723

INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

UMI

UMI Microform 1429723
Copyright 2006 by ProQuest Information and Learning Company. All rights reserved. This microform edition is protected against unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company
300 North Zeeb Road
P.O. Box 1346
Ann Arbor, MI 48106-1346

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
The Thesis prepared by

Rohith Ramnath

Entitled

"Analysis of Logic Block Architectures and Functional Improvement of Fine Grained Cells"

is approved in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering

Examination Committee Chair

Dean of the Graduate College

Examination Committee Member

Examination Committee Member

Graduate College Faculty Representative
ABSTRACT

Analysis of Logic Block Architectures and Functional Improvement of Fine Grained Cells

by

Rohith Ramnath

Dr. Yingtao Jiang, Examination Committee Chair
Assistant Professor
Department of Electrical & Computer Engineering
University of Nevada, Las Vegas

The first objective of this research project was to evaluate the performance of various logic block architectures in FPGAs. Since logic blocks widely vary in size, functionality and complexity, we were motivated to explore them in detail. For our study, logic blocks from Actel, Altera, Quicklogic and Xilinx were chosen along with some designs discussed in the academia. These cells were either multiplexer based or look-up-table (LUT) based. Structural VHDL models of all these blocks were constructed and benchmarks circuits were mapped. Results at this stage suggested that, although the coarse grained cells occupied more area and showed poor utilization, they were considerably faster than the fine grained cells.

The second objective was to improve the performance of the Actel Proasicplus (fine grained) logic block by enhancing its functional capabilities. During this process we came up with three modified architectures. These new
cells were laid out in MAGIC using a TSMC 0.18μm technology file with λ = 0.09μm and the extracted files were simulated in PSpice. This transistor level data helped us to estimate the area and propagation delay of the new architectures. The modified architectures were also tested for performance by implementing the previous benchmarks and a significant improvement in speed, occupied area and utilization was observed.
# LIST OF ACRONYMS

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A&lt;sub&gt;l&lt;/sub&gt;</td>
<td>Area of a Logic Block</td>
</tr>
<tr>
<td>ASIC</td>
<td>Application Specific Integrated Circuit</td>
</tr>
<tr>
<td>B&lt;sub&gt;1&lt;/sub&gt;</td>
<td>Benchmark – 1</td>
</tr>
<tr>
<td>B&lt;sub&gt;2&lt;/sub&gt;</td>
<td>Benchmark – 2</td>
</tr>
<tr>
<td>B&lt;sub&gt;3&lt;/sub&gt;</td>
<td>Benchmark – 3</td>
</tr>
<tr>
<td>B&lt;sub&gt;4&lt;/sub&gt;</td>
<td>Benchmark – 4</td>
</tr>
<tr>
<td>CAD</td>
<td>Computer Aided Design</td>
</tr>
<tr>
<td>CLB</td>
<td>Configurable Logic Block</td>
</tr>
<tr>
<td>CLK</td>
<td>Clock</td>
</tr>
<tr>
<td>CLR</td>
<td>Clear</td>
</tr>
<tr>
<td>CMOS</td>
<td>Complimentary Metal Oxide Semi-conductor</td>
</tr>
<tr>
<td>D&lt;sub&gt;l&lt;/sub&gt;</td>
<td>Delay of a Logic Block</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processing</td>
</tr>
<tr>
<td>DSPLM</td>
<td>Digital Signal Processing Logic Module</td>
</tr>
<tr>
<td>DSPFPGA</td>
<td>Digital Signal Processing Field Programmable Gate Array</td>
</tr>
<tr>
<td>DSRFPGA</td>
<td>Digit Serial Reconfigurable Field Programmable Gate Array</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array</td>
</tr>
<tr>
<td>HDL</td>
<td>Hardware Description Language</td>
</tr>
<tr>
<td>I/O</td>
<td>Input/Output</td>
</tr>
<tr>
<td>IOB</td>
<td>Input Output Block</td>
</tr>
<tr>
<td>LAB</td>
<td>Logic Array Block</td>
</tr>
<tr>
<td>LB</td>
<td>Logic Block</td>
</tr>
<tr>
<td>LC</td>
<td>Logic Cell</td>
</tr>
<tr>
<td>LE</td>
<td>Logic Element</td>
</tr>
<tr>
<td>LM</td>
<td>Logic Module</td>
</tr>
<tr>
<td>LPPGA</td>
<td>Low Power Programmable Gate Array</td>
</tr>
<tr>
<td>LUT</td>
<td>Look-Up Table</td>
</tr>
<tr>
<td>MAC</td>
<td>Multiply Accumulate</td>
</tr>
<tr>
<td>MPG</td>
<td>Mask Programmable Gate Array</td>
</tr>
<tr>
<td>MUX</td>
<td>Multiplexer</td>
</tr>
<tr>
<td>N&lt;sub&gt;L&lt;/sub&gt;</td>
<td>Number of Logic Blocks</td>
</tr>
<tr>
<td>N&lt;sub&gt;LC&lt;/sub&gt;</td>
<td>Number of Logic Blocks in the Critical Path</td>
</tr>
<tr>
<td>NMOS</td>
<td>Negative Channel Metal Oxide Semi-conductor</td>
</tr>
<tr>
<td>PLD</td>
<td>Programmable Logic Devices</td>
</tr>
<tr>
<td>PMOS</td>
<td>Positive Channel Metal Oxide Semi-conductor</td>
</tr>
<tr>
<td>RA</td>
<td>Routing Area</td>
</tr>
<tr>
<td>RAM</td>
<td>Random Access Memory</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Rp</td>
<td>Routing Delay</td>
</tr>
<tr>
<td>RTL</td>
<td>Register Transfer Level</td>
</tr>
<tr>
<td>SRAM</td>
<td>Static Random Access Memory</td>
</tr>
<tr>
<td>VHDL</td>
<td>VHSIC Hardware Description Language</td>
</tr>
<tr>
<td>VHSIC</td>
<td>Very High Speed Integrated Circuit</td>
</tr>
<tr>
<td>TSMC</td>
<td>Taiwan Semiconductor Manufacturing Corporation</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure 2.1 A Conceptual FPGA ................................................................. 06
Figure 2.2 Island Style Architecture ......................................................... 07
Figure 2.3 Row based Architecture ......................................................... 08
Figure 2.4 Basic Logic Element ............................................................... 11
Figure 2.5 Cluster based logic blocks ...................................................... 13
Figure 3.1 CMOS Inverter ...................................................................... 18
Figure 3.2 Actel's Proasicplus Logic Block ........................................... 20
Figure 3.3 Flash Switch versus Multiplexer ............................................ 22
Figure 3.4 Altera’s Stratix Logic Element ............................................... 24
Figure 3.5 Altera’s Stratix in Normal Mode .......................................... 26
Figure 3.6 Altera’s Stratix in Dynamic Arithmetic Mode .................. 26
Figure 3.7 Quicklogic’s Eclipse II Logic Cell ....................................... 32
Figure 3.8 Slice of Spartan II CLB ......................................................... 36
Figure 3.9 The DSP Logic Module ......................................................... 40
Figure 3.10 Architecture of the DSR-FPGA Logic Block ..................... 41
Figure 3.11 Digit Serial Logic Block ....................................................... 42
Figure 3.12 The Low-Power Programmable Gate Array ....................... 44
Figure 3.13 Boolean Function Implementation in Actel’s Proasicplus ...... 48
Figure 4.1 Comparison of Number of Logic Blocks for benchmark-1 ... 56
Figure 4.2 Comparison of Occupied Area for benchmark-1 ................. 57
Figure 4.3 Comparison of Delay when RD = 0 for benchmark-1 ........... 58
Figure 4.4 Comparison of Delay when RD = 10DL for benchmark-1 ...... 58
Figure 4.5 Comparison of Utilization for benchmark-1 ......................... 59
Figure 4.6 Comparison of Number of Logic Blocks for benchmark-2 ... 60
Figure 4.7 Comparison of Occupied Area for benchmark-2 .................. 61
Figure 4.8 Comparison of Delay when RD = 0 for benchmark-2 ........... 62
Figure 4.9 Comparison of Delay when RD = 10DL for benchmark-2 ...... 62
Figure 4.10 Comparison of Utilization for benchmark-2 ......................... 63
Figure 4.11 Comparison of Number of Logic Blocks for benchmark-3 ... 64
Figure 4.12 Comparison of Occupied Area for benchmark-3 ................. 65
Figure 4.13 Comparison of Delay when RD = 0 for benchmark-3 ........... 66
Figure 4.14 Comparison of Delay when RD = 10DL for benchmark-3 ...... 66
Figure 4.15 Comparison of Utilization for benchmark-3 ......................... 67
Figure 4.16 Comparison of Number of Logic Blocks for benchmark-4 ... 68
Figure 4.17 Comparison of Occupied Area for benchmark-4 ................. 69
Figure 4.18 Comparison of Delay when RD = 0 for benchmark-4 ........... 70
Figure 4.19 Comparison of Delay when RD = 10DL for benchmark-4 ...... 70
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.20</td>
<td>Comparison of Utilization for benchmark-4</td>
</tr>
<tr>
<td>4.21</td>
<td>The Proasicplus logic tile</td>
</tr>
<tr>
<td>4.22</td>
<td>Architecture of Mod1</td>
</tr>
<tr>
<td>4.23</td>
<td>Architecture of Mod2</td>
</tr>
<tr>
<td>4.24</td>
<td>Architecture of Mod3</td>
</tr>
<tr>
<td>4.25</td>
<td>Layout of Actel Proasicplus Logic Block</td>
</tr>
<tr>
<td>4.26</td>
<td>Layout of Mod1</td>
</tr>
<tr>
<td>4.27</td>
<td>Layout of Mod2</td>
</tr>
<tr>
<td>4.28</td>
<td>Layout of Mod3</td>
</tr>
<tr>
<td>A1</td>
<td>2-AND Gate using Actel's Proasicplus Logic Block</td>
</tr>
<tr>
<td>A2</td>
<td>2-OR Gate using Actel's Proasicplus Logic Block</td>
</tr>
<tr>
<td>A3</td>
<td>2-XOR Gate using Actel's Proasicplus Logic Block</td>
</tr>
<tr>
<td>A4</td>
<td>3-AND Gate using Actel's Proasicplus Logic Block</td>
</tr>
<tr>
<td>A5</td>
<td>3-NOR Gate using Actel's Proasicplus Logic Block</td>
</tr>
<tr>
<td>A6</td>
<td>DFF using Actel's Proasicplus Logic Block</td>
</tr>
<tr>
<td>B1</td>
<td>2-AND Gate using Mod1 architecture</td>
</tr>
<tr>
<td>B2</td>
<td>2-OR Gate using Mod1 architecture</td>
</tr>
<tr>
<td>B3</td>
<td>2-XOR Gate using Mod1 architecture</td>
</tr>
<tr>
<td>B4</td>
<td>3-AND Gate using Mod1 architecture</td>
</tr>
<tr>
<td>B5</td>
<td>3-NOR Gate using Mod1 architecture</td>
</tr>
<tr>
<td>B6</td>
<td>3-NAND Gate using Mod1 architecture</td>
</tr>
<tr>
<td>B7</td>
<td>3-OR Gate using Mod1 architecture</td>
</tr>
<tr>
<td>C1</td>
<td>2-AND Gate using Mod2 architecture</td>
</tr>
<tr>
<td>C2</td>
<td>2-OR Gate using Mod2 architecture</td>
</tr>
<tr>
<td>C3</td>
<td>2-XOR Gate using Mod2 architecture</td>
</tr>
<tr>
<td>C4</td>
<td>3-AND Gate using Mod2 architecture</td>
</tr>
<tr>
<td>C5</td>
<td>3-NOR Gate using Mod2 architecture</td>
</tr>
<tr>
<td>C6</td>
<td>3-NAND Gate using Mod2 architecture</td>
</tr>
<tr>
<td>C7</td>
<td>3-OR Gate using Mod2 architecture</td>
</tr>
<tr>
<td>C8</td>
<td>3-XOR Gate using Mod2 architecture</td>
</tr>
<tr>
<td>C9</td>
<td>4-AND Gate using Mod2 architecture</td>
</tr>
<tr>
<td>C10</td>
<td>4-NOR Gate using Mod2 architecture</td>
</tr>
<tr>
<td>C11</td>
<td>4-NAND Gate using Mod2 architecture</td>
</tr>
<tr>
<td>C12</td>
<td>4-OR Gate using Mod2 architecture</td>
</tr>
<tr>
<td>C13</td>
<td>5-AND Gate using Mod2 architecture</td>
</tr>
<tr>
<td>C14</td>
<td>5-NOR Gate using Mod2 architecture</td>
</tr>
<tr>
<td>C15</td>
<td>5-NAND Gate using Mod2 architecture</td>
</tr>
<tr>
<td>C16</td>
<td>5-OR Gate using Mod2 architecture</td>
</tr>
<tr>
<td>C17</td>
<td>DFF using Mod2 architecture</td>
</tr>
<tr>
<td>D1</td>
<td>2-AND Gate using Mod2 architecture</td>
</tr>
<tr>
<td>D2</td>
<td>2-OR Gate using Mod2 architecture</td>
</tr>
<tr>
<td>D3</td>
<td>2-XOR Gate using Mod2 architecture</td>
</tr>
<tr>
<td>D4</td>
<td>3-AND Gate using Mod2 architecture</td>
</tr>
<tr>
<td>D5</td>
<td>3-NOR Gate using Mod2 architecture</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>D6</td>
<td>3-NAND Gate using Mod2 architecture</td>
</tr>
<tr>
<td>D7</td>
<td>3-OR Gate using Mod2 architecture</td>
</tr>
<tr>
<td>D8</td>
<td>3-XOR Gate using Mod2 architecture</td>
</tr>
<tr>
<td>D9</td>
<td>4-AND Gate using Mod2 architecture</td>
</tr>
<tr>
<td>D10</td>
<td>4-NOR Gate using Mod2 architecture</td>
</tr>
<tr>
<td>D11</td>
<td>4-NAND Gate using Mod2 architecture</td>
</tr>
<tr>
<td>D12</td>
<td>4-OR Gate using Mod2 architecture</td>
</tr>
<tr>
<td>D13</td>
<td>5-AND Gate using Mod2 architecture</td>
</tr>
<tr>
<td>D14</td>
<td>5-NOR Gate using Mod2 architecture</td>
</tr>
<tr>
<td>D15</td>
<td>5-NAND Gate using Mod2 architecture</td>
</tr>
<tr>
<td>D16</td>
<td>5-OR Gate using Mod2 architecture</td>
</tr>
<tr>
<td>D17</td>
<td>DFF using Mod2 architecture</td>
</tr>
</tbody>
</table>
LIST OF TABLES

Table 3.1   Area and Delay estimates of logic block components .......... 19
Table 3.2   Area and Delay of different Altera families ......................... 31
Table 3.3   Area and Delay of different Quicklogic families ..................... 35
Table 3.4   Total Delay as including routing and combinational delay .......... 49
Table 3.5   Results for benchmark-1 .................................................. 50
Table 3.6   Results for benchmark-2 .................................................. 51
Table 3.7   Results for benchmark-3 .................................................. 52
Table 3.8   Results for benchmark-4 .................................................. 53
Table 4.1   Logical capabilities of Actel Proasicplus .................................. 75
Table 4.2   Logical abilities of Mod1 .................................................. 77
Table 4.3   Logical abilities of Mod2 and Mod3 ....................................... 79
Table 4.4   Performance comparison in terms of area ................................ 86
Table 4.5   Propagation delay of the four architectures ................................ 88
Table 4.6   Comparative results for benchmark-1 .................................... 89
Table 4.7   Comparative results for benchmark-2 .................................... 89
Table 4.8   Comparative results for benchmark-3 .................................... 90
Table 4.9   Comparative results for benchmark-4 .................................... 90
ACKNOWLEDGEMENTS

I take this opportunity to thank my advisor Dr. Yingtao Jiang, for his guidance and support. I would also like to thank Dr. Rama Venkat for helping me through out my Master’s program.

I would like to thank Dr. Venkatesan Muthukumar for his valuable suggestions and comments.

I would like to thank Mr. Stan Hanel for providing access to Active-HDL whenever I needed it, without which, I’d have struggled a lot.

I would also like to thank all my friends, Anoop, Bhaarath, Girish, Kapil and Ram, who have been with me all through.

Last but not the least, I am thankful to my family and my fiancée, Brinda, for their continuous support and encouragement.
CHAPTER 1

INTRODUCTION

Field Programmable Gate Arrays (FPGAs) are now firmly established as design and prototyping tools and are showing up in a variety of applications [1] [2] [3] [4] [5]. The design of an FPGA is very similar to an MPGA (Mask Programmable Gate Array) with the exception that all its connections are user programmable. Ever since their introduction in the late 80's, different architectures have come forward but a typical FPGA consists of an array of logic blocks surrounded by a programmable interconnection network with Input/Output blocks on the periphery. As the name suggests, logic blocks are used to implement the logic, while the programmable interconnections provide the resources to connect the logic blocks to form the required circuit [6] [7] [8] [9] [10].

The main advantages of an FPGA are low non-recurring manufacturing cost and a fast turn around time [5] [10] [11] [12]. This instant programmability gives systems built with these devices a significant time-to-market advantage [6] [7] [8] [9]. However, this programmability comes at a price, since FPGAs are slower and demand more silicon area when compared to MPGAs or ASICs [3] [6] [9] [13] [14]. This is because ASICs use simple
wires for interconnections between logic gates but in FPGAs, gates are connected through programmable switches. These switches have much larger resistance and capacitance and hence are slower than the simple wires used in MPGAs [6] [13] [15].

1.1 Motivation

Over the last couple of decades, ASICs have been widely used for myriad of applications. However they suffer from very high initial investment and significant turn around time. Although FPGAs overcome these limitations, they are a compromise in terms of speed and density [3] [6] [7] [8] [14] [15]. Hence to market FPGAs as attractive choices it is essential to improve their performance and make them comparable to ASICs and MPGAs.

In general, the three main factors affecting overall FPGA performance are:

- architecture of the FPGA which includes both logic block as well as interconnection network,
- quality of CAD tools,
- the electrical transistor level design of the FPGA [13] [15].

Selecting the right architecture is one of the most important steps in building an FPGA. The architecture has three components: logic blocks, interconnects and I/O blocks [6] [7] [8] [9] [10]. The goal of this thesis is to explore the different FPGA logic block architectures, their relative strengths
and weaknesses and also to improve their performance by enhancing functionality without adding too many transistors.

1.2 Problem Definition

An FPGA logic block can be as simple as a transistor pair or as complex as a microprocessor [7]. A logic block is typically capable of implementing many different combinational and sequential functions. The current versions of these logic block architectures are either look-up-table (LUT) based or multiplexer based. Depending on the functionality of logic blocks they are classified as:

- Small: logic blocks, which can at most implement any 3 variable function.
- Medium: logic blocks, which can at most implement any boolean function between 4 and 6 variables.
- Large: logic blocks, which can implement any function greater than 6 variables.

During the course of our research we found out that for a given boolean function, we need more small grained cells than their competitors. These higher numbers of logic cells in turn relate to increased communication between logic blocks and hence prove costly in terms of interconnect resources. Although the delay through the logic block is comparatively small, routing delay which is a combination of both wire delay as well as switch
matrix delay dominates the scenario and degrades performance [12] [16] [17] [18].

Hence our purpose in this research work was to explore the merits and demerits of different logic block architectures and also to improve performance of the small grained logic cells.

1.3 Thesis Outline

This thesis has been divided into 5 chapters. Chapter 2 gives some necessary background information, including the relevant aspects of commercially available FPGAs and related prior work.

Chapter 3 describes the first phase of our research work and explains experimental procedure used to evaluate the performance of different logic blocks and presents the model used for measuring area, delay and utilization for a given logic block.

Chapter 4 presents the results obtained during the early part of this research work and describes the second phase of this project which deals with the design new functionally enhanced logic block architectures. This chapter also gives an insight towards their transistor level implementation and discusses the results obtained when these architectures were tested with benchmark circuits.

Finally, Chapter 5 draws conclusions and explains what can be done in future.
CHAPTER 2

BACKGROUND AND PREVIOUS WORK

The first half of this chapter provides an introduction to FPGAs and describes some of the present day commercial architectures, while the second half provides information about previous research work related to our study.

2.1 Field Programmable Gate Arrays

The Field Programmable Gate Array or FPGA as it is more widely called is a type of programmable device that can be configured to perform a wide variety of applications. An FPGA has three major configurable elements:

- Logic blocks
- Input/Output blocks
- Programmable Interconnects.

Logic blocks are cells that can be configured to implement boolean functions. The input/output blocks provide the interface between the package pins and internal signal lines. The programmable interconnect resources provide routing paths to connect the inputs and outputs of the logic blocks as well as input/output blocks. A user's design is implemented by specifying the simple logic function for each cell and selectively closing the switches in the
interconnect matrix. The array of logic cells and interconnects form a fabric of basic building blocks for logic circuits. Combining these basic blocks to create the desired circuit creates complex designs [7] [8] [9] [10] [13] [18]. The schematic of a conceptual FPGA is as shown below:

There are four classes of FPGA's: island style, row based, hierarchical PLD and sea-of-gates [6] [8] [9] [10] [13]. The most popular ones are the island style and row based architectures.
2.1.1 The Island Style Architecture

The island style architecture consists of an array of programmable logic blocks with vertical and horizontal programmable routing channels. The basic architecture is illustrated in figure 2.2. Xilinx FPGAs are classic examples of this kind of architecture [6] [19] [20] [38].

![Island Style Architecture Diagram](image)

Figure 2.2: Island Style Architecture

2.1.2 Row-Based Architecture:

As the name implies, this architecture has logic blocks arranged in rows with horizontal routing channel between successive rows as shown in the figure 2.3. The routing tracks within the channel are divided into one or more
segments. The length of the segments can vary from the width of a module pair to the full length of the channel. The segments can be connected at the ends using programmable switches to increase their length. Other tracks run vertically through the logic blocks. They provide connections between the horizontal routing channel and the vertical routing segments. The family of FPGAs from Actel has this kind of architecture [7] [8] [9] [21] [35].

![Diagram of Row Based Architecture](image)

Figure 2.3: Row Based Architecture

After their introduction in the mid 80's FPGAs have now become a billion dollar industry. Many companies have come up with different FPGA architectures and the most popular ones in today's world are either
multiplexer based or LUT based. Companies like Altera and Xilinx have logic blocks which use a look-up-table as their basic element while others like Actel and Quicklogic use multiplexers to build their logic cells. Both these approaches share the common goal of achieving a high logic density and speed performance.

The following are some general terms used frequently in subsequent chapters:

- **Logic Blocks**: that portion of the basic tile of an FPGA which implements both combinational and sequential logic of a circuit. An FPGA may have only one type of logic block or there may be different types of logic blocks. In either case the logic blocks are arranged in a two dimensional array.

- **Logic Block Functionality**: this term refers to the size of the granularity of a logic block. On a broader sense, the functionality of a logic block refers to the logical capabilities of that block. In general a logic block with lesser functionality is referred to as a fine grained cell and a one with greater functionality is referred to as a coarse grained cell [7] [18] [22] [23].

- **Routing Architecture**: this refers to the routing resources that are available to interconnect logic blocks. These resources have wire tracks that can be interconnected by using programmable switches.
• Input/Output Blocks: commonly referred to as IOBs, these blocks appear on the periphery of an FPGA and are used to connect logic blocks to the physical pins of the chip. Typically an IOB allows a pin to be programmed as an input, an output or a bidirectional port. Additional processing of a signal like inversion or latching may also be provided in an IOB.

• Critical Path: refers to that path in the circuit which has the largest delay from one end to the other. In an FPGA, critical path is the longest path which is terminated at both ends by an IOB and may contain one or more logic blocks between its ends.

2.2 Logic Block Architecture

Logic block in an FPGA can be implemented in ways that differ in number of inputs and outputs, amount of area consumed, complexity of logic functions that it can implement and the total number of transistors it consumes [6] [7] [12] [13] [16] [17] [22] [23] [24]. The most common versions however, are either multiplexer based or LUT based.

2.2.1 Look-Up-Table Based Logic Blocks

The basis for a LUT based logic block is an SRAM performing as a function generator [12] [24] [25] [26] [38]. The truth table for a K-input logic function is stored in a $2^K \times 1$ SRAM. The address lines of the SRAM act as inputs and the output of the SRAM provides the value of the logic function.
The advantage of look-up-tables is that they exhibit very high functionality [24]. A K-input LUT can implement any function of K-inputs and there are $2^k$ such functions. The disadvantage is that they are demanding in terms of area because the area of a logic block increases exponentially with its inputs [17] [22]. A basic model of a LUT based logic block is as shown in figure 2.4 [18] [22] [28].

![Figure 2.4: The basic logic element](image)

The area of a logic block of the above form is a function of the number of its inputs and the amount of fixed hardware it contains [6] [17] [18] [22]. The total active area for a given implementation is the product of number of logic blocks and the area of each block. The number of logic blocks is a decreasing function of K (number of inputs to a LUT) because a more functional block has higher logic capabilities and hence can implement more of the original circuit. On the other hand, the area of a logic block increases exponentially with its inputs, as a K-input LUT requires $2^K$ programming bits [12] [16] [17] [22] [23] [25] [28].
Another important aspect of LUT based logic blocks is the presence of a flip flop within each logic cell. An FPGA implemented using logic blocks without an embedded flip flop requires more logic blocks than the one with an embedded flip flop because each flip-flop must be implemented using several logic blocks [17] [22] [28]. This feature facilitates the implementation of most of the current algorithms and applications which are pipelined in order to achieve higher speeds.

Hence the two most important conclusions to be drawn are:

- the optimum number of inputs required to achieve the lowest total area lies between four and six consistently and this minima shows very little dependence on the programming technology [17] [22] [28].

- a flip flop is required within a logic block in order to reduce overall chip area [17] [22].

A cluster based approach has also been explored in these LUT based logic blocks. The structure of the cluster based logic block is illustrated in figure 2.5. Each cluster has N identical logic blocks and there exists full connectivity among all the logic blocks within a cluster [14] [32] [33]. Most of the recent logic blocks by Altera and Xilinx have cluster based logic cell. A cluster size of 4 has been found to be most area efficient. These cluster based logic blocks have very little placement time associated with them [15] [29] [30].
Another variation in look-up-table based blocks is the use of heterogeneous logic cells [31]. This structure utilizes LUTs of different sizes within a logic block in order to improve performance. The heterogeneous logic blocks of sizes (6,4), (5,2), (4,2) and (4,3) were experimentally found to outperform the most energy-efficient 4-input homogeneous logic block.
architectures [13] [15] [31]. The Xilinx Spartan XL is a perfect example of a heterogeneous logic block [38].

2.2.2 Multiplexer based Logic Blocks:

The multiplexer based logic blocks implement different logic functions by connecting each of its inputs either to a constant or to a signal. For example consider a 2-1 multiplexer with select input ‘s’, inputs ‘a’ & ‘b’ and output ‘f’ = sa + s’b. By setting signal ‘b’ to logic ‘0’, the multiplexer can implement the AND function f =sa and by connecting input ‘a’ to logic ‘1’, the multiplexer implements an OR function. Hence by connecting together a number of multiplexers and basic logic gates, a logic block can be constructed which can implement large number of functions [6] [7] [35].

Multiplexer-based logic blocks have the advantage of providing large degree of functionality for a relatively small number of transistors. This is however, achieved at the expense of a large number of inputs which place a high demand on routing resources [17] [18] [22] [23]. When compared to the LUT based architectures, the multiplexer logic blocks are fine grained, but show higher utilization factors, since it is easier to use small logic gates efficiently. The disadvantage of these logic blocks is that they require a relatively large number of wire segments and programmable switches. Such routing resources are costly both in terms of area and delay [17] [18] [22] [23] [28]. Hence researchers are constantly trying to improve the functionality of the logic block so as to reduce the overhead on interconnect resources.
CHAPTER 3

EXPERIMENTAL PROCEDURE

This chapter presents our experimental setup. Our research work is split up into two phases. In the first phase, we analyzed the performance of various commercial as well as academic logic block architectures by modeling them in VHDL and analyzing them with benchmark circuits in order to verify their functionality. We also came up with a modeling scheme to estimate the area occupied and propagation delay of these designs. This data gave us a clear indication of the strengths and weaknesses of various logic cells.

In the second phase, we concentrated on improving the performance of Actel’s Proasicplus logic cell, which falls under the category of small grained logic blocks. This boost in performance was achieved by enhancing the functionality of the logic cell without adding a lot of silicon overhead. Three promising architectures are suggested and their capabilities explored. They were also built at transistor level in a 0.18µm CMOS process in order to estimate area and propagation delay.
3.1 Structural Models of Logic Blocks using VHDL

Aldec's Active-HDL was used to build the logic blocks at RTL level. To facilitate design mapping and data hierarchy, we used a structural method to create these logic blocks. The VHDL codes of all the logic blocks were tested with benchmarks and were also compiled and simulated.

3.2 Area and Delay Models for FPGA Logic Blocks

This section describes a method, based on transistor count, for modeling parameters like area and delay of various FPGA logic block architectures.

3.2.1 Background

Since most of the FPGAs came out in the CMOS era, it is safe to assume that the prevailing logic style used to design the various components was CMOS. The main reason for this is its robustness. The CMOS logic style is very reliable and is the least prone to external influences like noise, garbage data etc. However it is true that, CMOS logic style occupies redundant area and hence need not have been employed throughout the chip. Since transistor level information is not available in any of the datasheets, we are forced to assume that the CMOS logic style prevails all over the chip.

Most FPGA logic blocks use components like inverters, AND gates, OR gates, EXOR gates and D flip flops to name a few. So, to estimate the area or delay of the logic block we first have to approximate the area and delay of the elements that make up a logic block.
3.2.2 Area and Delay Modeling of the Elements of a Logic Block

Estimates of area and delay of any gate can be obtained by analyzing these components at transistor level. Consider the simple case of an inverter, the transistor level schematic of a CMOS inverter is as shown in the following figure. For reasons of symmetry, the inverter is assumed to have equal rise and fall times, which makes the PMOS approximately twice as wide as the NMOS [39] [40]. The area occupied by such an inverter is assumed to be ‘A’ sq. units and the propagation delay is assumed to be ‘D’ time units. From this, the area of a single NMOS is \( \frac{A}{3} \) while the area of the PMOS is \( \frac{2A}{3} \).

Figure 3.1: CMOS Inverter
Extending the same methodology to a 2-input NAND gate, we have:

Area of a 2-NAND gate is ‘2A’ Sq. Units

Delay of a 2-NAND gate is ‘D’ Time Units.

Using the same concept, area and delay of various other logic gates were estimated and the data is presented below in table 3.1.

Table 3.1: Area and Delay estimates of various components

<table>
<thead>
<tr>
<th>Component</th>
<th>Area (A sq. units)</th>
<th>Delay (D time units)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inverter</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NAND gate</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>NOR gate</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>AND gate</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>OR gate</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>EXOR gate</td>
<td>2.30</td>
<td>2</td>
</tr>
<tr>
<td>2 to 1 Mux</td>
<td>3.60</td>
<td>2</td>
</tr>
<tr>
<td>SRAM</td>
<td>2.66</td>
<td>2</td>
</tr>
<tr>
<td>FLASH Switch</td>
<td>0.66</td>
<td>1</td>
</tr>
<tr>
<td>D Flip Flop</td>
<td>12.33</td>
<td>5</td>
</tr>
<tr>
<td>2-LUT</td>
<td>18.64</td>
<td>5</td>
</tr>
<tr>
<td>3-LUT</td>
<td>36.94</td>
<td>6</td>
</tr>
<tr>
<td>4-LUT</td>
<td>71.56</td>
<td>7</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
For our analysis purposes we have chosen the following architectures:

- Actel’s Proasicplus
- Altera’s Flex6k, Flex10k, Apex II, Apex 20k, Mercury and Stratix
- Quicklogic’s PASIC1, PASIC 3 and Eclipse II
- Xilinx’s Spartan XL and Spartan II
- DSP logic Module
- Digit Serial Reconfigurable Field Programmable Gate Array
- Low Power Programmable Gate Array

3.3 Actel’s Proasicplus

Actel’s Proasicplus is their latest FPGA family. This is a 3.3V, 0.22μm, 4-layer CMOS process. The architecture of the Proasicplus logic cell is as shown in the following figure [35].

![Diagram of Actel's Proasicplus logic block.](image)

Figure 3.2: Actel's Proasicplus logic block. Courtesy: Actel's Datasheets [35].
3.3.1 Logic Cell Architecture

The Proasicplus FPGA series consists of many multiplexer based logic tiles that can be configured by programming the appropriate flash switch interconnections. Flash switches are distributed throughout the device to provide non-volatile, reconfigurable interconnect programming. The logic cell has three inputs and one output and is capable of implementing most of the 3-variable functions. The inputs are available in both true and complemented forms and either of them can be chosen. The cell can also be configured as a latch or a flip-flop [35].

3.3.2 Routing Structure

The routing structure of the Proasicplus FPGA has four levels of hierarchy in the form of local resources, long line resources, high-speed long lines and high performance global lines. The local lines allow the output of each logic tile to be directly connected to any of its eight neighboring inputs. The long lines vary in length by 1, 2 or 4 logic tiles. All the logic tiles are capable of driving signals onto these long lines, which can in turn access every input of every logic tile. The lines provide routing resources for higher fan-out connections. The high-speed long lines span the entire length and breadth of the device and are used for very high fan-out nets. The global lines are used to distribute clocks, resets and other high fan-out nets which require minimum skew [35].
3.3.3 VHDL Modeling

A VHDL model of the above logic cell was constructed using Aldec's Active-HDL tool. To facilitate design mapping, a structural approach was followed. Since VHDL is mainly an RTL level tool, the FLASH switches were represented as multiplexers with select lines provided by the user. The functionality of the logic cell was verified by implementing a few basic gates.

![Flash Switch vs. Multiplexer](image)

Figure 3.3: Flash Switch vs. Multiplexer

3.3.4 Area and Delay of Actel’s Proasicplus Logic Cell:

From the schematic of the Proasicplus logic cell, we can make out that this logic block has the following components:

- Inverters (7)
- 2:1 Mux (2)
- 2-NAND Gate (2)
- Flash Switches (14)

Based on the data from table 3.1 we can estimate the total area required for one logic tile as:

\[
\text{Total Area} = 7 \times 1A + 2 \times 3.6A + 2 \times 2A + 14 \times 0.66A
\]
The combinational delay of the logic block can easily be approximated by identifying the critical path in the logic block. The critical path in Actel’s ProASICplus logic tile consists of the following components:

- Inverter
- Flash Switch
- 2:1 Mux
- Inverter
- Flash Switch
- 2:1 Mux
- 2-NAND Gate
- Inverter

Based on the data from table 3.1, the worst-case delay through this logic cell can be estimated as:

**Total Delay = D + D + 2D + D + D + 2D + D + D**

**Total Delay = 10D Time Units.**

3.4 Altera’s Stratix

The most recent FPGA family from Altera is the Stratix series, which is based on a 1.5V, 0.13μm, all layer copper SRAM process. These devices have logic blocks spread out in rows and columns. The basic block of the Stratix device is called the “Logic Element” (LE) whose architecture is as shown in the figure below.
3.4.1 Logic Cell Architecture

A four input LUT forms the core of the Logic Element and hence each LE can implement any 4-variable function. Along with the LUT, each LE also has a register and dedicated paths for carry-select and cascade chain functions. The carry-select chain is used for high speed arithmetic functions like counters and adders and the cascade chain is used for wide-input functions such as comparators. The register is programmable and can be configured as a D, T, SR or JK flip-flop. Because each LE contains a flip-flop, pipelined designs can easily be implemented. For combinational functions, the flip-flop can be bypassed and the output of the LUT drives the output of
Array Blocks (LABs). Each LAB comprises of 10 logic elements, control signals and local interconnect, LUT chain and register chain connection lines. The LAB provides a coarse grained structure and facilitates efficient routing, device utilization and high performance.

3.4.2 Modes of Operation

The Stratix operates in either normal mode or dynamic arithmetic mode. In normal mode, the LE can be used to implement combinational functions and general logic applications. The four data inputs and the carry-in signal form the inputs to the logic element. The LE output can be either registered or unregistered. Each LE can use the LUT chain connections to drive its combinatorial output directly to the next LE within the same LAB. The architecture of the Stratix device in its normal mode is as shown in figure 3.5.

As the name suggests the dynamic arithmetic mode is used to implement arithmetic functions like adders, counters, accumulators etc. An LE in dynamic arithmetic mode uses 4 two input LUTs which are configurable as a dynamic adder/subtractor. The first two LUTs compute two summations based on a possible carry-in of 1 or 0, while the other two LUTs generate carry outputs for the two chains of a carry select circuitry. As before, an LE can drive registered and unregistered versions of the LUT output. The structure of the LE in this mode is as shown in figure 3.6 [36]:
Figure 3.5: Altera's Stratix in Normal Mode. Courtesy Stratix Datasheet

Figure 3.6: Altera's Stratix in Arithmetic Mode. Courtesy Stratix Datasheet.
3.4.3 Routing Structure

A series of column and row interconnects of varying lengths and speed, provide signal interconnects within the device. Within the LAB, the routing structure consists of local interconnect, LUT chain and register chain. The local interconnect provides connections between logic elements of the same LAB. The LUT chain connections transfer the output of one LE’s LUT to the adjacent LE for fast sequential LUT connections within the same LAB. The register chain connections transfer the output of one LE register to the adjacent LE within an LAB.

The MultiTrack interconnects provide routing resources throughout the device. These consist of continuous, performance optimized, routing lines of different lengths and speeds, and provide connections within a block or between two blocks. The MultiTrack interconnects have row and column interconnects that span fixed distances. The row interconnects provide resources for LABs within the same row. These include:

- Direct link interconnects between LABs and adjacent blocks
- R4 interconnects traversing four blocks to the right or left
- R8 interconnects traversing eight blocks to the right or left
- R24 interconnects traversing the entire length of the device

The column interconnects are very similar to the row interconnect but are used for vertical communication. These include

- LUT chain interconnects within an LAB
• Register chain interconnects within an LAB
• C4 interconnects traversing four blocks in up or down direction
• C8 interconnects traversing eight blocks in up or down direction
• C16 interconnects traversing throughout the device [36]

3.4.4 VHDL Modeling

The core of the logic cell, which is the LUT, was created on the basis of a multiplexer tree. The normal mode and arithmetic modes were constructed separately and benchmarks were implemented.

3.4.5 Area and Delay Model of Stratix LE

The area and delay estimates vary with the mode of operation. For normal mode, we can observe that the LE consists of the following components:

- 2:1 Mux (7)
- 4-LUT (1)
- 2-XOR Gate (1)
- 2-AND Gate (2)
- 2-NAND Gate (2)
- 3-NOR Gate (1)
- 4-1 Mux (1)
- D-flip flop (1)

Hence the total area occupied by the logic element is

\[ \text{Total Area} = 7 \times 3.6A + 1 \times 71.56A + 1 \times 12.33A + 1 \times 2.3A + 2 \times 3A + 2 \times 2A \]
Total Area = 139.19 A Sq.Units.

The worst-case path has the following blocks:

- 4-1 Mux
- 4-LUT
- 2-1 Mux
- 2-1 Mux
- 2 AND Gate
- D-flip flop
- 2-1 Mux

Hence the delay of this logic tile is estimated as:

\[
\text{Total Delay} = 4D + 7D + 2D + 2D + 2D + 5D + 2D
\]

\[
\text{Total Delay} = 24D \text{ Time Units}
\]

In its dynamic arithmetic mode of operation, the LE comprises of:

- 2-1 Mux (11)
- 2-LUT (4)
- 2-XOR Gate (1)
- 2-AND Gate (3)
- 2-NAND Gate (1)
- 3-NOR Gate (1)
- 4-1 Mux (1)
- D-flip flop (1)
Hence the total area occupied is approximately equal to:

\[ \text{Total Area} = 11 \times 3.6A + 4 \times 18.64A + 1 \times 12.33A + 1 \times 2.3A + 1 \times 10.8A + 1 \times 7A + 1 \times 2A + 3 \times 3A \]

\[ \text{Total Area} = 157.59 \text{ A Sq.Units.} \]

From the above schematic we can make out that the worst case path has the following components:

- 2-XOR Gate
- 2-LUT
- 2-1 Mux
- 2-1 Mux
- 2-1 Mux
- 2-AND Gate
- D-flip flop
- 2-1 Mux

Based on the data from table 3.1 we can estimate the total combinational delay for one logic tile as:

\[ \text{Total Delay} = 2D + 5D + 2D + 2D + 2D + 2D + 5D + 2D \]

\[ \text{Total Delay} = 23 \text{ D Time Units} \]

In addition to the Stratix family other families like Flex, Apex and Mercury were also analyzed. Structural models of logic elements in different modes from these families were constructed and tested for logic
implementation. Furthermore, the area and delay of the LEs were computed and is presented in the following table:

<table>
<thead>
<tr>
<th>Device Family</th>
<th>Operating Mode</th>
<th>Area (A Sq.Units)</th>
<th>Delay (D Time Units)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flex6k</td>
<td>Normal</td>
<td>119.69</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Arithmetic</td>
<td>118.41</td>
<td>15</td>
</tr>
<tr>
<td>Flex10k</td>
<td>Normal</td>
<td>131.89</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Arithmetic</td>
<td>123.41</td>
<td>15</td>
</tr>
<tr>
<td>Apex II</td>
<td>Normal</td>
<td>126.89</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Arithmetic</td>
<td>122.01</td>
<td>15</td>
</tr>
<tr>
<td>Apex 20k</td>
<td>Normal</td>
<td>126.89</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Arithmetic</td>
<td>122.01</td>
<td>15</td>
</tr>
<tr>
<td>Mercury</td>
<td>Normal</td>
<td>134.09</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Arithmetic</td>
<td>144.29</td>
<td>18</td>
</tr>
</tbody>
</table>

3.5 Quicklogic

The newest addition to the fleet of FPGAs from Quicklogic is the Eclipse series. This FPGA family is based on a 3.3V, 0.25μm, 5-layer CMOS process, whose architecture is as shown in the next page [37].
3.5.1 Logic Tile Architecture

The Eclipse family is a dual register multiplexer based logic cell which is designed for wide fan-in and multiple, simultaneous output functions. The logic cell consists of two 6-input AND gates, four 2-input AND gates, seven 2-to-1 multiplexers and two D flip flops. The cell has a fan-in of 30 and is claimed to be able to fit functions which are as wide as 17 simultaneous inputs. The logic cell has 6 simultaneous outputs, two of which can be registered. This high logic capacity and wide fan-in accommodates many user functions with a single level of logic delay [37].
3.5.2 Routing Structure

Six types of routing resources are provided: short wires, dual wires, quad wires, express wires, distributed networks and default wires. Short wires span the length of one logic cell in the vertical direction. Dual wires run horizontally and cover two logic cells. Short and Dual wires are primarily used for local connections. Quad wires span four logic cells and are used for the implementation of medium fan-out nets. Express lines and distributed networks run the length of the FPGA and carry signals which require low skews. These lines have higher capacitance than a quad line or a short line but have lesser capacitance than shorter wires connected to run the entire length of the device. Also, the resistance is reduced as no intermediate switches are required. These lines are used for long routes or high fan-out nets [37].

3.5.3 VHDL Modeling

A structural VHDL model comprising of all the components was built and tested for its operation.

3.5.4 Area and Delay of the Eclipse Logic Cell

The Eclipse logic cell has the following components:

- 2-1 Mux (7)
- 2-AND Gate (4)
- 6-AND Gate (2)
- D-flip flop (2)
The total area is approximately equal to

\[
\text{Total Area} = 7 \times 3.6A + 2 \times 12.33A + 4 \times 3A + 2 \times 14.33A
\]

\[
\text{Total Area} = 90.52 \text{ A Sq.Units}
\]

The delay of the logic cell is estimated by following the critical path which consists of the following components:

- Inverter
- 6-AND Gate
- 2-1 Mux
- 2-1 Mux
- 2-1 Mux
- D-flip flop

Hence the delay through the block is equal to

\[
\text{Total Delay} = D + 2D + 2D + 2D + 2D + 5D
\]

\[
\text{Total Delay} = 14D \text{ Time Units}
\]

Along with the Eclipse family other families like PASIC 1 and PASIC 3 were also analyzed. Structural models of these logic cells were constructed and tested for logic implementation. As in previous cases, their area and delay were also estimated and the values are as shown in the following table.
Table 3.3: Area and Delay of Quicklogic Families

<table>
<thead>
<tr>
<th>Device Family</th>
<th>Area (A Sq. Units)</th>
<th>Delay (D Time Units)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PASIC 1</td>
<td>71.79</td>
<td>12</td>
</tr>
<tr>
<td>PASIC 3</td>
<td>82.59</td>
<td>14</td>
</tr>
</tbody>
</table>

3.6 Xilinx

Xilinx introduced the concept of FPGAs with its XC2000 series and their latest FPGA family is the Virtex series but the architecture of the logic block is very similar to their predecessors, i.e. the Spartan II family. This series is based on a 2.5V, 0.18μm CMOS process. It has a regular, flexible programmable architecture comprised of Configurable Logic Blocks (CLBs), which are in turn surrounded by a perimeter of programmable Input/Output blocks (IOBs). These functional elements are interconnected by a powerful hierarchy of versatile routing channels.

3.6.1 Logic Cell Architecture

The basic building block of the Spartan II FPGA is the logic cell (LC). Each LC consists of a four-input function generator, fast-carry logic and a storage element. Each CLB has four LCs, arranged in two similar slices. The following figure shows one slice of this CLB.
Each logic cell is capable of implementing any 4-input function and can also be configured a 16 * 1 bit synchronous RAM. In addition to this, the four LCs can be combined to implement additional combinational logic of five or six input variables and claims that it can also implement some functions of nine inputs and some selected functions of nineteen inputs.

Figure 3.8: Slice of the Spartan II CLB. Courtesy Spartan Datasheet.
3.6.2 Routing Structure

Routing resources of Xilinx are very robust. It includes local routing, general purpose routing, dedicated routing, I/O routing and global routing. The local routing resources provide interconnections among LUTs, flip flops, internal feedback paths that provide high speed connections within a CLB and direct paths for horizontally adjacent CLBs. The general purpose routing is located in horizontal and vertical routing channels associated with rows and columns of CLBs. This provides resources for adjacent CLBs in both horizontal and vertical directions. The general routing resource includes 24 single length lines, 96 buffered hex length lines, out of which one-third are bidirectional and 12 buffered bidirectional long lines. The I/O routing provides additional routing resources around the periphery of the device. This additional routing is called the VersaRing. In Spartan II dedicated routing is provided for 3-state busses in the horizontal direction and the carry signal in the vertical direction. Lastly global routing resources distribute clocks and other signals with very high fan-outs throughout the device [19] [20] [38].

3.6.3 VHDL Modeling

For easier design mapping, structural VHDL models were constructed. The LUT was based on the concept of multiplexer tree. A single logic cell was built its functional capabilities were explored by implementing some basic gates.
3.6.4 Area and Delay Models of the Spartan II Family

From the above schematic we can make out that this logic block has the following components:

- 2-1 Mux (20)
- 4-1 Mux (8)
- Carry Logic (4)
- 2-XOR Gate (4)
- 2-AND Gate (4)
- 4-LUT (4)
- D-flip flop (4)

Based on the data from table 3.1 we can estimate the total area required for one logic block as:

\[
\text{Total Area} = 20 \times 3.6A + 8 \times 10.8 + 4 \times 24.6A + 4 \times 2.3A + 4 \times 3A + 4 \times 71.56A + 4 \times 12.33A
\]

Total Area = 613.56 A Sq.Units.

From the above schematic we can make out that the worst case path has the following components:

- 4-LUT
- 2-1 Mux
- 2-1 Mux
- 2-OR Gate
- 4-1 Mux
• 2·1 Mux
• D-flip flop

This delay can be estimated as:

$$\text{Total Delay} = 7D + 2D + 2D + 4D + 2D + 5D$$

$$\text{Total Delay} = 24D \text{ Time Units.}$$

In the same lines we also modeled the Spartan XL architecture and its area and delay turned out to be 286.32A Sq units and 24D Time Units respectively.

3.7 DSP FPGA

The DSP logic module appeared in the IEEE transactions on VLSI Systems in the year 1995. This article describes an EXOR based logic block designed primarily for DSP applications. The architecture of this DSP-FPGA is as shown in the following figure. As can be observed, this fine grain logic cell has 5-inputs and 2-outputs and shows high functionality. Basic gates were implemented and the logic cell was tested for its capabilities. It shows poor performance for sequential functions as the logic cell does not have a register. This is a serious flaw, since most of the current designs are pipelined [32].

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
3.7.1 Area and Delay of the DSP FPGA:

From the above schematic we can make out that this logic block has the following components:

- 2-1 Mux (1)
- Inverter (1)
- 2-AND Gate (2)
- 2-XOR Gate (2)
- 2-NOR Gate (1)

With this, area of the logic block is estimated as

\[ \text{Total Area} = 1 \times 3.6A + 1 \times A + 2 \times 3A + 2 \times 2.3A + 1 \times 3A \]

\[ \text{Total Area} = 18.2 \text{ A Sq.Units.} \]

From the above schematic we can make out that the worst case path has the following components:

- Inverter
- 2-1 Mux
- 2-AND Gate
- 2-NOR Gate

Based on the data from table 3.1 we can estimate the total combinational delay for one logic tile as:

\[
\text{Total Delay} = D + 2D + 2D + D
\]

\[
\text{Total Delay} = 6D \text{ Time Units.}
\]

3.8 Digit Serial Reconfigurable FPGA

The digit serial reconfigurable FPGA is very similar to the DSP-FPGA but also utilizes the concept of cluster based logic blocks. The basic cell of this logic block array is called as a Logic Module and is as shown in the following figure.

![Figure 3.10: Architecture of the Logic Module for DSR-FPGA](image)

A logic block array is made up of four such logic modules and hence a wide variety of boolean functions can be implemented. In addition to the logic
modules, the cell also comprises of fast carry logic circuitry and a register array. Each logic array block has a total of 26 inputs and 9 outputs. This logic block satisfies the requirement of rapid prototyping and efficient implementations of digit serial DSP applications. The structure of the digit serial logic block is as shown below [33]:

![Digit Serial Logic Block](image)

3.8.1 Area and Delay of the DSRFPGA Logic Module:

The LM has the following components

- 2:1 Mux (1)
- Inverter (1)
- 2-AND Gate (3)
- 2-XOR Gate (2)
2-OR Gate (1)

Based on the data from table 3.1 we can estimate the total area required for one logic block as:

\[
\text{Total Area} = 1 \times 3.6A + 1 \times 1A + 3 \times 3A + 2 \times 2.3A + 1 \times 4A
\]

\[
\text{Total Area} = 22.2 \text{ A Sq.Units}
\]

The delay through the logic module is estimated based on the components in the critical path

- 2-XOR Gate
- 2-AND Gate
- 2-1 Mux
- 2-AND Gate
- 2-OR Gate

Hence the total delay is equal to

\[
\text{Total Delay} = 2D + 2D + 2D + 2D + 2D
\]

\[
\text{Total Delay} = 10 \text{ D Time Units}
\]

3.9 Low Power Programmable Gate Array

The low power programmable gate array is an energy efficient FPGA architecture. The functionality of this logic cell is based on a cluster of 3-input LUTs as shown in the figure below. The 3-LUT is implemented as a multiplexer tree. The control signals to the multiplexer act as inputs to the LUT. The inputs to the LUT are stored in memory cells, while the functionality of the LUT is controlled by programming the memory cells.
based on the truth table of the required function. This clustering scheme makes it possible to combine the results of the four 3-input LUTs in various ways. All the outputs of the logic block can be registered if required [34].

![Low Power Programmable Gate Array](image)

**Figure 3.12: The Low Power Programmable Gate Array**

### 3.9.1 Area and Delay of LPPGA

The logic cell has the following components

- 2-1 Mux (9)
- 4-1 Mux (1)
- 3-LUT (4)
- Inverter (11)
- D-flip flop (3)

From table 3.1, the estimated area is equal to

\[
\text{Total Area} = 9 \times 3.6 + 1 \times 10.8A + 4 \times 36.94A + 11 \times A + 3 \times 12.33A
\]

Total Area = 238.95 A Sq.Units.

The critical path consists of:
- 2-1 Mux
- Inverter
- 3-LUT
- 4-1 Mux
- 2-1 Mux
- 2-1 Mux
- D-flip flop
- 2-1 Mux

Hence the delay through this logic block is equal to

\[
\text{Total Delay} = 2D + D + 6D + 4D + 2D + 5D + 2D
\]

Total Delay = 22D Time Units.

3.10 Performance Evaluation of Logic Blocks

The performance of the logic blocks is evaluated by implementing benchmarks on each of them. The following metrics were used to see the relative merits and demerits of logic blocks:
- Number of Logic Blocks: Represented as ‘N_l’, this refers to the actual number of cells that are required in order to successfully map the design on to logic blocks. This metric is inversely related to the functionality of the logic cell. That is, a logic cell with high functionality (coarse grained) will use fewer logic blocks to implement any design than a logic cell with lesser functionality (fine grained).

- Area: This parameter is directly related to the number of logic blocks. Area occupied by a particular design is given by

\[ \text{Area Occupied} = N_l \times A_l + R_a \]

Where \( N_l \) is the number of logic blocks required for a given design, \( A_l \) is the area of each block and \( R_a \) is the routing area.

- Delay: The delay for a given design is a function of the delay through the logic block and the routing delay. Hence the total delay is represented as

\[ \text{Total Delay} = N_{LC} \times D_l + (N_{LC} - 1 \times R_d) \]

Where \( N_{LC} \) is the number of logic blocks in the critical path, \( D_l \) is the delay through the logic block and \( R_d \) is the routing delay.

Of the two delays, the delay through the logic block can be estimated since each block has a specific architecture made up a number of gates. But, the routing delay is much harder to calculate, as this depends on the length of the wires, width of the channel, type of switches, switch
matrix etc. Because of this, the routing delay is estimated to be zero, twice, four times or ten times the delay through the logic block.

- **Utilization:** The utilization of a logic block is represented as a percentage. This metric gives a clear indication of the specific number of gates that are used to implement a given function in a logic block.

### 3.11 Benchmark Implementations

To understand the abilities of the logic blocks, we implemented a few benchmark circuits on each of the architectures discussed previously. Area and delay of each implementation were also calculated for performance comparison. The following benchmarks were chosen:

- 16-variable Boolean Function
- 32-bit Adder
- 16-bit Multiplier
- 16-bit MAC Unit

#### 3.11.1 Boolean Function

We implemented the following 16-variable Boolean function on all the logic block architectures:

\[
Q = ABCD + EFGH + (IJKL)' + (M + N + O + P)
\]

As can be observed, the equation is in sum of products form and the ‘+’ sign in the above function represents an ‘OR’ operation. In order to explain
the performance evaluation scheme adopted, let us consider Actel's Proasicplus logic cell as an example.

When implemented on Actel's logic blocks, the Boolean function was mapped as shown in figure 3.13.

![Diagram of Boolean Function Implementation in Actel's Proasicplus](image)

**Figure 3.13: Boolean Function Implementation in Actel's Proasicplus**

Based on the above schematic, the following parameters are estimated:

- **Area Required**:
  
  \[ = (N_L \times A_L) + R_A \]
  
  \[ = 13 \times 27.44 \text{ A Sq. Units} + R_A \]
  
  \[ = 356.72 \text{ A Sq. Units} + R_A \]

- **Total Delay**:
  
  \[ = (N_L \text{ in the critical path} \times D_L) + (N_L - 1 \times R_D) \]
  
  \[ = (4 \times 6D) + (3 \times R_D) \]
\[ = 24D + (3 \times R_D) \]

Now if we consider that the routing delay could be zero, twice, four times or ten times the average combinational delay, we have.

Table 3.4: Total Delay as a function of both routing and combinational delay

<table>
<thead>
<tr>
<th>Routing delay</th>
<th>R_D = 0 *D_{LB}</th>
<th>R_D = 2 * D_{LB}</th>
<th>R_D = 4 * D_{LB}</th>
<th>R_D = 10 *D_{LB}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Delay</td>
<td>24D</td>
<td>84D</td>
<td>144D</td>
<td>324D</td>
</tr>
</tbody>
</table>

- Utilization Factor: Since we have manually mapped the above boolean function onto the logic blocks, we can also estimate how much of the logic block's resources were utilized for implementation purposes. That is, we know that we have the following components in Actel's Proasicplus logic block:
  1. Inverters (7)
  2. 2-1 Mux (2)
  3. 2-NAND Gate (2)
  4. Flash Switches (14)

So, with 13 logic blocks the total number of components is 325, but to implement the Boolean function we use only around 112 components. Hence the average utilization factor is 34.46%.
A similar analysis was done on all the logic block architectures and the data collected is presented in the following table:

Table 3.5: Results for Benchmark-1

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Nl</th>
<th>Dl</th>
<th>Area</th>
<th>Rd=0</th>
<th>Rd=2D</th>
<th>Rd=4D</th>
<th>Rd=10D</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proasicplus</td>
<td>13</td>
<td>06</td>
<td>356.72</td>
<td>24</td>
<td>84</td>
<td>144</td>
<td>324</td>
<td>34.46</td>
</tr>
<tr>
<td>Flex6k</td>
<td>5</td>
<td>13</td>
<td>598.45</td>
<td>26</td>
<td>62</td>
<td>98</td>
<td>206</td>
<td>40.00</td>
</tr>
<tr>
<td>Flex10k</td>
<td>5</td>
<td>13</td>
<td>659.45</td>
<td>26</td>
<td>66</td>
<td>106</td>
<td>226</td>
<td>30.76</td>
</tr>
<tr>
<td>Apex II</td>
<td>5</td>
<td>13</td>
<td>634.45</td>
<td>26</td>
<td>66</td>
<td>106</td>
<td>226</td>
<td>33.33</td>
</tr>
<tr>
<td>Apex 20k</td>
<td>5</td>
<td>12</td>
<td>634.45</td>
<td>26</td>
<td>66</td>
<td>106</td>
<td>226</td>
<td>33.33</td>
</tr>
<tr>
<td>Mercury</td>
<td>5</td>
<td>09</td>
<td>670.45</td>
<td>18</td>
<td>54</td>
<td>90</td>
<td>198</td>
<td>15.38</td>
</tr>
<tr>
<td>Stratix</td>
<td>5</td>
<td>11</td>
<td>695.95</td>
<td>22</td>
<td>70</td>
<td>118</td>
<td>262</td>
<td>18.75</td>
</tr>
<tr>
<td>Pasic 1</td>
<td>3</td>
<td>07</td>
<td>215.37</td>
<td>14</td>
<td>28</td>
<td>42</td>
<td>84</td>
<td>60.00</td>
</tr>
<tr>
<td>Pasic 3</td>
<td>3</td>
<td>09</td>
<td>247.77</td>
<td>18</td>
<td>36</td>
<td>54</td>
<td>108</td>
<td>66.66</td>
</tr>
<tr>
<td>Eclipse</td>
<td>3</td>
<td>09</td>
<td>271.56</td>
<td>18</td>
<td>36</td>
<td>54</td>
<td>108</td>
<td>57.77</td>
</tr>
<tr>
<td>Spartan XL</td>
<td>3</td>
<td>09</td>
<td>858.96</td>
<td>18</td>
<td>66</td>
<td>114</td>
<td>258</td>
<td>41.66</td>
</tr>
<tr>
<td>Spartan II</td>
<td>2</td>
<td>11</td>
<td>1227.12</td>
<td>22</td>
<td>70</td>
<td>118</td>
<td>262</td>
<td>10.41</td>
</tr>
<tr>
<td>DSPLM</td>
<td>12</td>
<td>06</td>
<td>218.40</td>
<td>30</td>
<td>78</td>
<td>126</td>
<td>270</td>
<td>60.71</td>
</tr>
<tr>
<td>DSRFPGA</td>
<td>13</td>
<td>10</td>
<td>288.60</td>
<td>50</td>
<td>130</td>
<td>210</td>
<td>450</td>
<td>34.50</td>
</tr>
<tr>
<td>LP_PGA</td>
<td>5</td>
<td>18</td>
<td>1194.75</td>
<td>36</td>
<td>80</td>
<td>124</td>
<td>256</td>
<td>50.51</td>
</tr>
</tbody>
</table>
3.11.2 32-bit Adder

A 32-bit ripple carry adder was implemented on all the logic blocks and the various performance parameters were computed. This data is presented in the following table.

Table 3.6: Results for Benchmark-2

<table>
<thead>
<tr>
<th>Architecture</th>
<th>NL</th>
<th>DL</th>
<th>Area</th>
<th>Rd=0</th>
<th>Rd=2D</th>
<th>Rd=4D</th>
<th>Rd=10D</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proasicplus</td>
<td>384</td>
<td>06</td>
<td>12072.9</td>
<td>576</td>
<td>2476</td>
<td>4376</td>
<td>10076</td>
<td>34.66</td>
</tr>
<tr>
<td>Flex6k</td>
<td>32</td>
<td>10</td>
<td>3789.1</td>
<td>320</td>
<td>1250</td>
<td>2180</td>
<td>4970</td>
<td>40.00</td>
</tr>
<tr>
<td>Flex10k</td>
<td>32</td>
<td>10</td>
<td>3949.1</td>
<td>320</td>
<td>1250</td>
<td>2180</td>
<td>4970</td>
<td>36.36</td>
</tr>
<tr>
<td>Apex II</td>
<td>32</td>
<td>10</td>
<td>3904.3</td>
<td>320</td>
<td>1250</td>
<td>2180</td>
<td>4970</td>
<td>36.36</td>
</tr>
<tr>
<td>Apex 20k</td>
<td>32</td>
<td>10</td>
<td>3904.3</td>
<td>320</td>
<td>1250</td>
<td>2180</td>
<td>4970</td>
<td>36.36</td>
</tr>
<tr>
<td>Mercury</td>
<td>32</td>
<td>13</td>
<td>4617.2</td>
<td>416</td>
<td>1532</td>
<td>2648</td>
<td>5996</td>
<td>57.89</td>
</tr>
<tr>
<td>Stratix</td>
<td>32</td>
<td>09</td>
<td>5298.8</td>
<td>288</td>
<td>1714</td>
<td>3140</td>
<td>7418</td>
<td>43.47</td>
</tr>
<tr>
<td>Pasic 1</td>
<td>128</td>
<td>07</td>
<td>9189.1</td>
<td>448</td>
<td>1960</td>
<td>3472</td>
<td>8008</td>
<td>90.00</td>
</tr>
<tr>
<td>Pasic 3</td>
<td>128</td>
<td>09</td>
<td>10571.5</td>
<td>576</td>
<td>2340</td>
<td>4104</td>
<td>9396</td>
<td>76.92</td>
</tr>
<tr>
<td>Eclipse</td>
<td>128</td>
<td>09</td>
<td>12610.5</td>
<td>576</td>
<td>2340</td>
<td>4104</td>
<td>9396</td>
<td>66.66</td>
</tr>
<tr>
<td>Spartan XL</td>
<td>16</td>
<td>11</td>
<td>3149.5</td>
<td>176</td>
<td>0896</td>
<td>1616</td>
<td>3776</td>
<td>41.66</td>
</tr>
<tr>
<td>Spartan II</td>
<td>08</td>
<td>11</td>
<td>4940.4</td>
<td>088</td>
<td>0424</td>
<td>0760</td>
<td>1768</td>
<td>25.00</td>
</tr>
<tr>
<td>DSPLM</td>
<td>32</td>
<td>06</td>
<td>870.4</td>
<td>192</td>
<td>0564</td>
<td>0936</td>
<td>2052</td>
<td>97.47</td>
</tr>
<tr>
<td>DSRFPGA</td>
<td>32</td>
<td>10</td>
<td>934.4</td>
<td>320</td>
<td>0940</td>
<td>1560</td>
<td>3420</td>
<td>100.00</td>
</tr>
<tr>
<td>LP_PGA</td>
<td>32</td>
<td>12</td>
<td>7646.4</td>
<td>384</td>
<td>1748</td>
<td>3112</td>
<td>7204</td>
<td>42.85</td>
</tr>
</tbody>
</table>
3.11.3 16-bit Multiplier

The following table compares the performance of various logic blocks when a 16-bit multiplier was implemented. The basic block of this design is a four-bit multiplier cell.

Table 3.7: Results for Benchmark-3

<table>
<thead>
<tr>
<th>Architecture</th>
<th>NL</th>
<th>DL</th>
<th>Area</th>
<th>Rd=0</th>
<th>Rd=2D</th>
<th>Rd=4D</th>
<th>Rd=10D</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proasicplus</td>
<td>4992</td>
<td>06</td>
<td>156948.48</td>
<td>1962</td>
<td>8482</td>
<td>15002</td>
<td>34582</td>
<td>40.00</td>
</tr>
<tr>
<td>Flex6k</td>
<td>928</td>
<td>10</td>
<td>110212.16</td>
<td>1133</td>
<td>4493</td>
<td>7853</td>
<td>17933</td>
<td>40.00</td>
</tr>
<tr>
<td>Flex10k</td>
<td>928</td>
<td>10</td>
<td>116695.36</td>
<td>1133</td>
<td>4493</td>
<td>7853</td>
<td>17933</td>
<td>36.36</td>
</tr>
<tr>
<td>Apex II</td>
<td>928</td>
<td>10</td>
<td>114474.56</td>
<td>1133</td>
<td>4493</td>
<td>7853</td>
<td>17933</td>
<td>36.36</td>
</tr>
<tr>
<td>Apex 20k</td>
<td>928</td>
<td>10</td>
<td>114474.56</td>
<td>1133</td>
<td>4493</td>
<td>7853</td>
<td>17933</td>
<td>36.36</td>
</tr>
<tr>
<td>Mercury</td>
<td>928</td>
<td>13</td>
<td>131289.92</td>
<td>1465</td>
<td>5497</td>
<td>9529</td>
<td>21625</td>
<td>44.82</td>
</tr>
<tr>
<td>Stratix</td>
<td>928</td>
<td>09</td>
<td>147677.12</td>
<td>1019</td>
<td>6171</td>
<td>11323</td>
<td>26779</td>
<td>35.08</td>
</tr>
<tr>
<td>Proasic 1</td>
<td>3840</td>
<td>07</td>
<td>267993.60</td>
<td>1631</td>
<td>7199</td>
<td>12767</td>
<td>29471</td>
<td>87.66</td>
</tr>
<tr>
<td>Proasic 3</td>
<td>3840</td>
<td>09</td>
<td>309465.60</td>
<td>2097</td>
<td>8593</td>
<td>15089</td>
<td>34577</td>
<td>74.87</td>
</tr>
<tr>
<td>Eclipse</td>
<td>3776</td>
<td>09</td>
<td>372011.52</td>
<td>2097</td>
<td>8593</td>
<td>15089</td>
<td>34577</td>
<td>66.89</td>
</tr>
<tr>
<td>Spartan XL</td>
<td>464</td>
<td>11</td>
<td>132852.48</td>
<td>627</td>
<td>3315</td>
<td>6003</td>
<td>14067</td>
<td>39.36</td>
</tr>
<tr>
<td>Spartan II</td>
<td>232</td>
<td>11</td>
<td>143273.92</td>
<td>319</td>
<td>1663</td>
<td>3007</td>
<td>7039</td>
<td>27.87</td>
</tr>
<tr>
<td>DSPLM</td>
<td>1438</td>
<td>06</td>
<td>39116.60</td>
<td>786</td>
<td>2346</td>
<td>3906</td>
<td>8586</td>
<td>84.80</td>
</tr>
<tr>
<td>DSRFPGA</td>
<td>1184</td>
<td>10</td>
<td>34572.80</td>
<td>1170</td>
<td>3490</td>
<td>6370</td>
<td>14170</td>
<td>81.68</td>
</tr>
<tr>
<td>LP_PGA</td>
<td>1056</td>
<td>12</td>
<td>252331.20</td>
<td>1404</td>
<td>6508</td>
<td>11612</td>
<td>26924</td>
<td>42.85</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
3.11.4 16-bit MAC Unit

In order to evaluate the logic blocks for pipelined applications, a 16-bit multiply and accumulate unit was implemented. This data is tabulated below:

Table 3.8: Results for Benchmark-4

<table>
<thead>
<tr>
<th>Architecture</th>
<th>N_L</th>
<th>D_L</th>
<th>Area</th>
<th>Rd=0</th>
<th>Rd=2D</th>
<th>Rd=4D</th>
<th>Rd=10D</th>
<th>Utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proasicplus</td>
<td>5408</td>
<td>06</td>
<td>170027.5</td>
<td>2544</td>
<td>11004</td>
<td>19464</td>
<td>44844</td>
<td>43.55</td>
</tr>
<tr>
<td>Flex6k</td>
<td>960</td>
<td>15</td>
<td>114001.2</td>
<td>1613</td>
<td>5933</td>
<td>10253</td>
<td>23213</td>
<td>40.33</td>
</tr>
<tr>
<td>Flex10k</td>
<td>960</td>
<td>15</td>
<td>120644.4</td>
<td>1613</td>
<td>5933</td>
<td>10253</td>
<td>23213</td>
<td>36.66</td>
</tr>
<tr>
<td>Apex II</td>
<td>960</td>
<td>15</td>
<td>118378.8</td>
<td>1613</td>
<td>5933</td>
<td>10253</td>
<td>23213</td>
<td>36.66</td>
</tr>
<tr>
<td>Apex 20k</td>
<td>960</td>
<td>15</td>
<td>118378.8</td>
<td>1613</td>
<td>5933</td>
<td>10253</td>
<td>23213</td>
<td>36.66</td>
</tr>
<tr>
<td>Mercury</td>
<td>960</td>
<td>18</td>
<td>135907.2</td>
<td>2041</td>
<td>7225</td>
<td>12409</td>
<td>27961</td>
<td>45.43</td>
</tr>
<tr>
<td>Stratix</td>
<td>960</td>
<td>09</td>
<td>152976.0</td>
<td>1755</td>
<td>8379</td>
<td>15003</td>
<td>34875</td>
<td>36.08</td>
</tr>
<tr>
<td>Pasic 1</td>
<td>3904</td>
<td>12</td>
<td>384622.0</td>
<td>2239</td>
<td>9343</td>
<td>16447</td>
<td>37759</td>
<td>87.82</td>
</tr>
<tr>
<td>Pasic 3</td>
<td>3904</td>
<td>14</td>
<td>322431.3</td>
<td>2833</td>
<td>11121</td>
<td>19241</td>
<td>44273</td>
<td>75.00</td>
</tr>
<tr>
<td>Eclipse</td>
<td>3904</td>
<td>14</td>
<td>376814.0</td>
<td>2833</td>
<td>11121</td>
<td>19241</td>
<td>44273</td>
<td>66.93</td>
</tr>
<tr>
<td>Spartan XL</td>
<td>480</td>
<td>20</td>
<td>137433.6</td>
<td>947</td>
<td>4403</td>
<td>7859</td>
<td>18227</td>
<td>40.00</td>
</tr>
<tr>
<td>Spartan II</td>
<td>240</td>
<td>16</td>
<td>148214.4</td>
<td>447</td>
<td>2175</td>
<td>3903</td>
<td>9087</td>
<td>27.77</td>
</tr>
<tr>
<td>DSPLM</td>
<td>2526</td>
<td>06</td>
<td>68707.20</td>
<td>7314</td>
<td>21930</td>
<td>36546</td>
<td>80394</td>
<td>90.26</td>
</tr>
<tr>
<td>DSRFPFA</td>
<td>2208</td>
<td>10</td>
<td>64473.60</td>
<td>11410</td>
<td>34210</td>
<td>57010</td>
<td>125410</td>
<td>95.65</td>
</tr>
<tr>
<td>LP_PGA</td>
<td>1088</td>
<td>17</td>
<td>259977.6</td>
<td>1948</td>
<td>8460</td>
<td>14972</td>
<td>34508</td>
<td>42.96</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CHAPTER 4

RESULTS AND DISCUSSIONS

The first half of this chapter interprets the collected data. With the help of some useful graphs, the next few pages show the relative performance of logic blocks for each benchmark circuit. This analysis emphasizes the fact that coarse grained logic blocks are faster than the fine grained versions. So, the next phase in our research was to try and improve the functionality of the most common fine grain architecture, Actel's Proasicplus logic tile. The second half of this chapter describes the methodology and presents three novel architectures which show significant improvement in terms of area, delay or both.

4.1 Boolean Function

4.1.1 Number of Blocks

The following graph shows the variation in the number of logic blocks used among different logic block architectures. This plot clearly indicates that, for successful implementation of the boolean function, the fine grained cells use up more logic blocks than the coarse grained cells. Among these logic block
architectures, the Proasicplus, and the DSR FPGA logic cells use the highest number of logic blocks (13), while the Spartan II uses the least (2).

Figure 4.1: Comparison of the number of logic blocks for B1

4.1.2 Occupied Area:

The area occupied by the entire design among different architectures is compared in the following graph. We can observe that the coarse grained logic cells occupy more area than the fine grained versions. However this gap reduces as the designs get more and more complex. For this benchmark, the Spartan II occupies the highest area and the DSPLM occupies the least amount of area.
4.1.3 Delay

The total delay as a function of both combinational as well as routing delay is plotted for two cases. The first graph depicts the total delay purely as a function of the combinational delay through the logic blocks. This analysis indicates that the delay is least in case of Spartan XL, Mercury, Pasic 3 and Eclipse and is highest for DSR FPGA.

The second graph is a plot of the total delay when routing is also considered. As can be seen, the delay is least in the case of Pasic 3 and Eclipse architectures, but is highest in DSR FPGA.
Figure 4.3: Comparison of total delay when $R_d = 0$ for B1

Figure 4.3: Comparison of total delay when $R_d = 10D_t$ for B1
4.1.4 Utilization

The utilization factor for all the architectures is compared in the following graph. The utilization factor varies quite a lot between different logic cells. In general, the utilization factor is high for fine grained cells than coarse grained cells. From the graph, we see that, utilization is highest for Pasic 3 and lowest for Spartan II.

![Comparison of Utilization Factor for B1](image)

Figure 4.5: Comparison of Utilization Factor for B1

4.2 32-bit Adder

4.2.1 Number of Logic Blocks

A comparison between different architectures with respect to the number of logic blocks used to implement a 32-bit ripple carry adder is shown in the following figure.
As before, the number of logic blocks required for the design varies with functionality. Actel's Proasicplus logic cell requires the highest number of blocks while the Spartan II requires the least. Although the DSP LM and the DSR FPGA are fine grained logic cells, their design is optimal for the implementation of an adder and hence the number of logic cells is not very high.

4.2.2 Occupied Area

A plot comparing the occupied area for the design is presented. As seen in the graph, the fine grained cells occupy more area than the coarse grained ones. The occupied area is highest for Eclipse and is least for DSP LM.
4.2.3 Delay

Figure 4.8 is a graph comparing the total delay which is plotted as a function of combinational delay of the logic blocks. This graph shows that even without any routing issues, the fine grained cells have larger delays than the coarse grained cells and the same performance is observed when routing is considered. The Spartan II proves to be the fastest logic cell while the Actel Proasicplus is the slowest.
Figure 4.8: Comparison of Total delay when \( R_D = 0 \) for B2

Figure 4.9: Comparison of Total Delay with \( R_D = 10D_L \) for B2
4.2.4 Utilization Factor

The variation in the utilization factor among different logic block architectures is plotted below. We can observe that the fine grained logic cells like DSP LM, DSR FPGA and Pasic 1 achieve very high utilization factors, while the coarse grained cells like Spartan II show poor utilization.

![Comparison of Utilization factor for B2](image)

Figure 4.10: Comparison of Utilization factor for B2

4.3 16-bit Multiplier

4.3.1 Number of Logic Blocks

The number of logic cells required to implement a 16-bit multiplier in different architectures varies as shown in the graph below. The results obtained are very similar to the previous benchmarks. Actel’s Proasicplus
takes up the highest number of logic blocks while Xilinx's Spartan II requires the least.

![Comparison of number of logic blocks for B3](image)

**Figure 4.11**: Comparison of number of logic blocks for B3

### 4.3.2 Occupied Area

The area occupied by the design is compared among various logic cells. Since the design requires such a huge number of logic blocks, the area occupied by the design is directly related to the number of logic blocks. As can be observed, the Eclipse logic cell marks the upper bound while the DSP LM marks the lower bound.
4.3.3 Delay

The total delay of the design as a function of combinational delay alone is plotted below. From the graph, we can observe that the delay is largest for the Proasicplus architecture and smallest for the Spartan II logic cell. A similar performance is observed when routing delay is also considered. This can be attributed to the number of logic blocks in the critical path, which is considerably lesser for a coarse grained cell than a fine grained logic block.
Figure 4.13: Comparison of total delay when $R_D = 0$ for B3

Figure 4.14: Comparison of total delay when $R_D = 10D_L$ for B3
4.3.4 Utilization Factor

The following graph compares the utilization factor amongst different logic block architectures. Similar to previous occasions, the fine grained cells show better utilization factors than the coarse grained architectures.

![Utilization Factor Graph](image)

Figure 4.15: Comparison of Utilization Factor for B3

4.4 16-bit MAC Unit

4.4.1 Number of Logic Blocks

The variation in the number of logic blocks is shown in the following graph. Architectures with a memory element in their design require lesser cells to implement the MAC unit. The key point to be noticed is the increase in the number of logic blocks by DSP LM and the DSR FPGA. This is due to the fact that these cells do not have a flip-flop associated with their design,
and hence require a very high number of logic blocks. Actel's Proasicplus requires the highest number of logic blocks and the Spartan II architecture uses the least number of cells.

![Comparison of the number of logic blocks for B4](image)

Figure 4.16: Comparison of the number of logic blocks for B4

4.4.2 Occupied Area

In previous cases we had observed that the area of the implemented design was high for a coarse grained cell than a fine grained cell. However in this case, the difference isn't much since the fine grained cells lack a memory element. Quicklogic's Pasic 1 sets the upper limit on the occupied area while DSR FPGA sets the lower limit. The following graph shows the area occupied among different architectures.
4.4.3 Delay

The graph below compares the total delay with respect to the combinational delay alone. As can be seen, due to the nature of the benchmark circuit, the architectures with a memory element achieve better speeds than the ones without any memory element. The delay is largest for the DSR FPGA and is smallest for the Spartan II.
Figure 4.18: Comparison of total delay when $R_D = 0$ for B4

Figure 4.19: Comparison of Total Delay when $R_D = 10D_L$ for B4
4.4.4 Utilization Factor

Similar to previous benchmarks, the utilization factor was better for the fine grained cells than the coarse grained ones. Spartan II again marks the lower bound while the DSR FPGA marks the upper bound.

![Comparison of Utilization Factor for B4](image)

**Figure 4.20:** Comparison of Utilization Factor for B4

4.5 Observations

The results provided in the earlier sections are further analyzed and meaningful conclusions drawn. The following pages provide an insight towards the architectural design of a logic block.
4.5.1 Granularity

From the results obtained, we know that the granularity of a logic cell plays a very important role in its performance. For complex designs fine grained cells (logic cells with little functionality) are much slower than the coarse grained cells (logic cells with high functionality). This is because of the increased number of logic blocks needed to implement a particular design. Hence it is imperative that a logic block be more functional. On the other hand, the design of a logic block cannot be very complicated either. This would increase the combinational delay through the logic block to such a high extent that we wouldn’t gain anything in the end.

The other aspect associated with granularity is the area of the logic cell. A coarse grained logic block takes up much more area than the fine grained block. But this factor is overshadowed by the gain in speed as discussed later.

4.5.2 Utilization Factor

One of the unique aspects in our study of FPGA logic block architectures is the estimation of utilization factor. Our research shows that the fine grained logic cells achieve higher utilization factors than their coarse grained competitors. The price associated with this improvement is speed again. In a fine grained cell, there are very few logic gates present in the architecture. These gates are highly essential for the logic cell to perform its operations. Where as, in a coarse grained cell, we can observe a redundancy with respect
to the logic gates. This is provided for the sole purpose of increasing flexibility, which is an important factor for FPGAs as they are used for a wide variety of applications. These highly flexible architectures show very promising performances with respect to speed.

4.5.3 Speed of the design

The two important aspects separating an FPGA from an ASIC are density and speed of a particular design. FPGAs are primarily used for prototyping and low cost design implementations and in order to show comparable performance to an ASIC, there is a need for these FPGAs to be faster. From the data previously obtained, we can easily make out that FPGAs whose logic blocks have higher degrees of functionality are much faster than the ones with lesser functionality.

The above analysis holds good in spite of the fact that the coarse grained cells have very large combinational delays through the logic block. Since the delay of an implemented design on an FPGA is dominated by routing, the improvement in speed is quite significant for a high functionality logic block due to the reduced number of stages in the critical path of the design.

For example, consider the implementation of a 16-bit Multiplier. The Xilinx Spartan II, which is a coarse grained logic block, approximately has 29 logic blocks in its critical path while the Actel Proasicplus, which is a fine grained cell, has around 326 logic blocks. This factor directly reflects on the
implementation speed between the two architectures. The Xilinx Spartan II is around 5 times faster than the Actel Proasicplus.

Generalizing on the above result, this is the main reason why look-up-table based FPGAs have dominated the programmable logic market for so long and still continue to do so. Since LUTs offer much greater functionality and speed than any other kind of programmable architecture they are best suited for a wide range of end user applications.

4.5.4 Utility of a Flip-Flop

Most of the present day algorithms and designs need certain degree of pipelining in order to achieve better speeds. This makes it necessary for a logic block to have a register as part of its architecture. For example, consider the implementation of the 16-bit MAC unit. From the data, it is quite clear that logic blocks with memory elements in their design are much faster and than the ones without it. Even among the fine grained cells, Actel's Proasicplus, which can be configured as a D flip-flop, is at least 3 times faster than DSR FPGA which cannot be configured as a flip-flop. Hence in all of today's commercial architectures, we can observe the ability of the logic cell to be configured as a memory element.

4.6 Functional Improvement of Actel's Proasicplus

4.6.1 Actel's Proasicplus:

The architecture of the original Proasicplus is shown below.
4.6.1.1 Logic Capabilities

The following table lists the boolean functions that can be implemented by a single logic block. The first column specifies the boolean function that is being implemented while the next four columns provide the configuration data that is required in order to implement that particular function.

From the table we can clearly make out that the Proasicplus logic cell can implement all 2-variable and some of the 3-variable functions. This block also has the ability to be configured as a D-flip-flop. The only drawback is that, when the cell is configured as a memory element, it cannot be used to implement any logic. In other words, this logic cell cannot register its own output.
Table 4.1: Logical Capabilities of the Actel Proasicplus

<table>
<thead>
<tr>
<th>Boolean Functions</th>
<th>Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
</tr>
<tr>
<td>2-AND Gate</td>
<td>0</td>
</tr>
<tr>
<td>2-OR Gate</td>
<td>X</td>
</tr>
<tr>
<td>2-XOR Gate</td>
<td>X</td>
</tr>
<tr>
<td>3-AND Gate</td>
<td>X</td>
</tr>
<tr>
<td>3-NOR Gate</td>
<td>X'</td>
</tr>
<tr>
<td>D-Flip Flop</td>
<td>D</td>
</tr>
<tr>
<td>3-NAND Gate</td>
<td>Not Possible</td>
</tr>
<tr>
<td>3-OR Gate</td>
<td>Not Possible</td>
</tr>
<tr>
<td>3-XOR Gate</td>
<td>Not Possible</td>
</tr>
</tbody>
</table>

4.6.2 Motivation for Improvement

The architecture of the Actel Proasicplus clearly indicates that the logic cell can be used to implement boolean functions or can be configured as a flip-flop. But the two operations cannot be implemented simultaneously. Hence our goal was to modify the original architecture, so that the logic cell could register its own output. In this process, we came up with three promising architectures whose abilities are either same or at least twice as much as the original Proasicplus logic tile.
4.7 Modified Architectures

This section gives a brief introduction to the modified architectures and also provides an insight on their functional capabilities.

4.7.1 Mod1: Logic Cell with D-Flip Flop

The first modified architecture is as presented below. An important feature of this architecture is the inclusion of the D-flip flop. As shown in the following table, the functional ability of this logic cell is very similar to the original Proasicplus, except for the fact that this modified logic block can be used for sequential as well as combinational purposes simultaneously. Also the second output line that was added provides better flexibility and improves performance.

![Architecture of Mod1](https://via.placeholder.com/150)

**Figure 4.22: Architecture of Mod1**
Table 4.2: Logical Abilities of Mod1

<table>
<thead>
<tr>
<th>Boolean Functions</th>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>2-AND Gate</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>2-OR Gate</td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>2-XOR Gate</td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>3-AND Gate</td>
<td>X</td>
<td>Y'</td>
</tr>
<tr>
<td>3-NOR Gate</td>
<td>X'</td>
<td>Y</td>
</tr>
<tr>
<td>3-NAND Gate</td>
<td>X</td>
<td>Y'</td>
</tr>
<tr>
<td>3-OR Gate</td>
<td>X'</td>
<td>Y</td>
</tr>
<tr>
<td>D-Flip Flop</td>
<td>D</td>
<td>CLK</td>
</tr>
<tr>
<td>3-XOR Gate</td>
<td>Not possible</td>
<td></td>
</tr>
</tbody>
</table>

4.7.2 Mod2:

The second modified architecture is presented in the following figure. This logic cell has 8 inputs and 4 outputs and can be analyzed by splitting it into two sections. Each section has 4 inputs and 2 outputs and is capable of implementing all the Boolean functions that the original Proasicplus cell can. By combining two such halves, the resulting logic block is doubly efficient. The functional abilities of this block are listed in the table below. This logic
block can implement all 2 and 3-variable functions and most of 4 and 5 variable functions. Apart from all this, the logic cell can also be configured as a D flip-flop.

Figure 4.24: Architecture of Mod2
Table 4.3: Logical Capabilities of Mod2 and Mod3

<table>
<thead>
<tr>
<th>Boolean Functions</th>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>2-AND Gate</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>2-OR Gate</td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>2-XOR Gate</td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>3-AND Gate</td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>3-NOR Gate</td>
<td>X'</td>
<td>Y</td>
</tr>
<tr>
<td>3-NAND Gate</td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>3-OR Gate</td>
<td>X'</td>
<td>Y</td>
</tr>
<tr>
<td>3-XOR Gate</td>
<td>X</td>
<td>Y</td>
</tr>
<tr>
<td>4-AND Gate</td>
<td>W</td>
<td>X'</td>
</tr>
<tr>
<td>4-NOR Gate</td>
<td>W'</td>
<td>X</td>
</tr>
<tr>
<td>4-NAND Gate</td>
<td>W</td>
<td>X'</td>
</tr>
<tr>
<td>4-OR Gate</td>
<td>W'</td>
<td>X</td>
</tr>
<tr>
<td>4-XOR Gate</td>
<td>Not Possible</td>
<td></td>
</tr>
<tr>
<td>5-AND Gate</td>
<td>P</td>
<td>Q'</td>
</tr>
<tr>
<td>5-NOR Gate</td>
<td>P'</td>
<td>Q</td>
</tr>
<tr>
<td>5-NAND Gate</td>
<td>P</td>
<td>Q'</td>
</tr>
<tr>
<td>5-OR Gate</td>
<td>P'</td>
<td>Q</td>
</tr>
<tr>
<td>5-XOR Gate</td>
<td>Not Possible</td>
<td></td>
</tr>
<tr>
<td>D-Flip Flop</td>
<td>0</td>
<td>Clk</td>
</tr>
</tbody>
</table>
4.7.3 Mod3

One of the architectural features in the previous modifications as well as the original Proasicplus logic cell is the choice in inputs. That is, all the inputs are available in both true and complemented forms. But the design of the logic cell, allows only one of these inputs to be used at any given point in time. Hence in our third modification, we designed the logic block without this flexibility. As can be seen in the following figure, this approach reduced the total number of gates by a big margin without any loss of functionality. This cell also can implement all 2 and 3 variable functions and most of 4 and 5 variable functions. The configuration of the logic block for various Boolean functions is similar to the previous version and is as tabulated in table 3.

![Architecture of Mod3](image)

**Figure 4.24: Architecture of Mod3**
4.8 Transistor Level Modeling

In order to better understand the area and delay requirements of the above architectures, we also did a transistor level analysis. Following CMOS logic style, the pre-layout simulations were done in Pspice with a 0.18μm model card. The layout was carried out in MAGIC and the extracted files were later simulated again in Pspice and verified for timing issues.

4.8.1 Layout Specifications

In this era of deep submicron devices, we used a TSMC 0.18μm technology file with λ = 0.09μm, for our layout purposes. This technology file consists of 6 metal layers and 1 poly layer and is for 1.8 volt applications.

Since none of the transistor level information is available in any of the data sheets, the devices were assumed to have minimum feature size. That is the width of the transistor was equal to 0.36μm and the length of the transistor was 0.18μm. This also ensured that all the architectures were optimized for area. Of the available 6 metal layers, two were used for this study. Metal1 was used for all inter-cell routing and Metal2 was used for all intra-cell routing.

The following four figures represent the layouts of all the architectures.
Figure 4.25: Layout of Actel Proasicplus Logic Block
Figure 4.26: Layout of Mod1
Figure 4.27: Layout of Mod2
4.9 Performance Comparison

Performance comparison among these architectures is two fold. The first metric is area, which can be represented in terms of number of transistors and actual layout area. However, it is important to note that the layout area does not serve as a metric since different designers might layout the same design differently. Hence, transistor count is used as the deciding factor among different architectures. For all the architectures, the layout area and the number of transistors are presented in the following table.
Table 4.4: Performance comparison in terms of Area

<table>
<thead>
<tr>
<th>Architecture</th>
<th># of Transistors</th>
<th>Area (in mm²)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proasicplus</td>
<td>74</td>
<td>6.89</td>
</tr>
<tr>
<td>Mod1</td>
<td>66</td>
<td>6.73</td>
</tr>
<tr>
<td>Mod2</td>
<td>112</td>
<td>9.53</td>
</tr>
<tr>
<td>Mod3</td>
<td>48</td>
<td>4.00</td>
</tr>
</tbody>
</table>

As seen in the table, the original Proasicplus design requires about 74 transistors and an area of 6.89 Sq mm. The first modified architecture (the one with the flip-flop) has a transistor count of 66 which is about 10.8% less than the original cell. Although the second modified architecture needs 112 transistors, this design is twice as capable as the original cell. That is, the original logic block needs at least 148 transistors (excluding routing) to perform the same functions as the mod2 architecture. Hence on the whole, we save about 24.3% in area. The most area efficient amongst all the above architectures is the mod3. This design offers high functionality with a very low transistor count (48) and hence saves about 67.5% in area. From an area perspective, mod3 is the most efficient.

The second metric used for performance comparison is the propagation delay of a logic block. This delay varies with the function that is being implemented. As shown below, the propagation delay was measured for
different boolean functions. The original Proasicplus has an average delay of 228.4 ps. However the Mod1 architecture, which offers the same functionality, has an average delay of 206.42 ps and hence proves to be around 9.6% faster. For the same functionality, the Mod2 and Mod3 architectures have an average delay of 195.5 ps and 170.21 ps respectively, which means that the speed of the Mod2 is better by at least 14.4% and that of Mod3 by 25.2%. This improvement in delay is also evident in designs that exploit the full functionality of these logic cells.
Table 4.5: Propagation Delay of the four architectures

<table>
<thead>
<tr>
<th>Boolean Function</th>
<th>Proasicplus</th>
<th>Mod1</th>
<th>Mod2</th>
<th>Mod3</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-AND Gate</td>
<td>236.5</td>
<td>227.5</td>
<td>228.5</td>
<td>187.5</td>
</tr>
<tr>
<td>2-OR Gate</td>
<td>236.5</td>
<td>227.5</td>
<td>228.5</td>
<td>187.5</td>
</tr>
<tr>
<td>2-XOR Gate</td>
<td>230.5</td>
<td>229.5</td>
<td>185.25</td>
<td>187.5</td>
</tr>
<tr>
<td>3-AND Gate</td>
<td>230.0</td>
<td>232.5</td>
<td>212.5</td>
<td>187.5</td>
</tr>
<tr>
<td>3-NOR Gate</td>
<td>208.0</td>
<td>230.5</td>
<td>190.5</td>
<td>189.0</td>
</tr>
<tr>
<td>3-NAND Gate</td>
<td>155.0</td>
<td>172.5</td>
<td>128.25</td>
<td></td>
</tr>
<tr>
<td>3-OR Gate</td>
<td>142.5</td>
<td>149.75</td>
<td>128.0</td>
<td></td>
</tr>
<tr>
<td>3-XOR Gate</td>
<td>386.5</td>
<td>386.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-AND Gate</td>
<td>420.0</td>
<td>386.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-NOR Gate</td>
<td>416.5</td>
<td>386.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-NAND Gate</td>
<td>372.0</td>
<td>342.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-OR Gate</td>
<td>380.5</td>
<td>342.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4-XOR Gate</td>
<td>Not Possible</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-AND Gate</td>
<td>422.5</td>
<td>386.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-NOR Gate</td>
<td>416.5</td>
<td>386.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-NAND Gate</td>
<td>376.5</td>
<td>342.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-OR Gate</td>
<td>380.5</td>
<td>342.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5-XOR Gate</td>
<td>Not Possible</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Among the four architectures, Mod3 is the fastest and offers very high functionality and occupies the least amount of area. These architectures were also tested with the previous benchmarks and the results are as shown below.

Table 4.6: Comparative Results for Benchmark-1

<table>
<thead>
<tr>
<th>Architecture</th>
<th>N</th>
<th>D</th>
<th>Area</th>
<th>Rd=0</th>
<th>Rd=2D</th>
<th>Rd=4D</th>
<th>Rd=10D</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proasicplus</td>
<td>13</td>
<td>06</td>
<td>356.7</td>
<td>24</td>
<td>84</td>
<td>144</td>
<td>324</td>
<td>34.46</td>
</tr>
<tr>
<td>Mod1</td>
<td>10</td>
<td>6</td>
<td>279.3</td>
<td>22</td>
<td>94</td>
<td>166</td>
<td>382</td>
<td>51.10</td>
</tr>
<tr>
<td>Mod2</td>
<td>5</td>
<td>13</td>
<td>250.8</td>
<td>28</td>
<td>56</td>
<td>84</td>
<td>168</td>
<td>48.82</td>
</tr>
<tr>
<td>Mod3</td>
<td>5</td>
<td>10</td>
<td>112.4</td>
<td>20</td>
<td>40</td>
<td>60</td>
<td>120</td>
<td>64.24</td>
</tr>
</tbody>
</table>

Table 4.7: Comparative results for Benchmark-2

<table>
<thead>
<tr>
<th>Architecture</th>
<th>N</th>
<th>D</th>
<th>Area</th>
<th>Rd=0</th>
<th>Rd=2D</th>
<th>Rd=4D</th>
<th>Rd=10D</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proasicplus</td>
<td>224</td>
<td>06</td>
<td>7042.56</td>
<td>576</td>
<td>2476</td>
<td>4376</td>
<td>10076</td>
<td>32.00</td>
</tr>
<tr>
<td>Mod1</td>
<td>192</td>
<td>06</td>
<td>5362.5</td>
<td>352</td>
<td>1864</td>
<td>3376</td>
<td>7912</td>
<td>45.37</td>
</tr>
<tr>
<td>Mod2</td>
<td>80</td>
<td>10</td>
<td>4815.36</td>
<td>320</td>
<td>1636</td>
<td>2952</td>
<td>6900</td>
<td>50.00</td>
</tr>
<tr>
<td>Mod3</td>
<td>80</td>
<td>10</td>
<td>1798.4</td>
<td>240</td>
<td>1180</td>
<td>2120</td>
<td>4940</td>
<td>71.42</td>
</tr>
</tbody>
</table>
Table 4.8: Comparative Results for Benchmark-3

<table>
<thead>
<tr>
<th>Architecture</th>
<th>N_L</th>
<th>D_L</th>
<th>Area</th>
<th>Rd=0</th>
<th>Rd=2D</th>
<th>Rd=4D</th>
<th>Rd=10D</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proasicplus</td>
<td>4128</td>
<td>06</td>
<td>156948.48</td>
<td>1962</td>
<td>8482</td>
<td>15002</td>
<td>34562</td>
<td>40.00</td>
</tr>
<tr>
<td>Mod1</td>
<td>3648</td>
<td>10</td>
<td>101888.64</td>
<td>1200</td>
<td>6408</td>
<td>11616</td>
<td>27240</td>
<td>45.34</td>
</tr>
<tr>
<td>Mod2</td>
<td>1456</td>
<td>10</td>
<td>73032.96</td>
<td>1093</td>
<td>5629</td>
<td>10165</td>
<td>23773</td>
<td>50.00</td>
</tr>
<tr>
<td>Mod3</td>
<td>1456</td>
<td>10</td>
<td>32730.88</td>
<td>820</td>
<td>4060</td>
<td>7300</td>
<td>17020</td>
<td>71.42</td>
</tr>
</tbody>
</table>

Table 4.9: Comparative results for benchmark-4

<table>
<thead>
<tr>
<th>Architecture</th>
<th>N_L</th>
<th>D_L</th>
<th>Area</th>
<th>Rd=0</th>
<th>Rd=2D</th>
<th>Rd=4D</th>
<th>Rd=10D</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Proasicplus</td>
<td>4384</td>
<td>06</td>
<td>137832.96</td>
<td>2541</td>
<td>10981</td>
<td>19421</td>
<td>44741</td>
<td>38.66</td>
</tr>
<tr>
<td>Mod1</td>
<td>3840</td>
<td>06</td>
<td>107251.20</td>
<td>1552</td>
<td>8296</td>
<td>15040</td>
<td>35272</td>
<td>45.82</td>
</tr>
<tr>
<td>Mod2</td>
<td>1568</td>
<td></td>
<td>78650.88</td>
<td>1425</td>
<td>7333</td>
<td>13241</td>
<td>30965</td>
<td>48.52</td>
</tr>
<tr>
<td>Mod3</td>
<td>1568</td>
<td></td>
<td>35248.64</td>
<td>1069</td>
<td>5289</td>
<td>9509</td>
<td>22169</td>
<td>67.85</td>
</tr>
</tbody>
</table>

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
CHAPTER 5

CONCLUSIONS AND FUTURE WORK

5.1 Summary and Contributions

Currently, most of the logic blocks are either LUT based or multiplexer based. LUT based logic blocks offer very high functionality but also need more area, which grows exponentially with its inputs. The multiplexer based logic blocks can be chosen as an alternative but their functionality is not as high as their competitors. However, these logic blocks are much more area efficient and are a lot faster.

Our study was aimed at analyzing the logic block architectures of various commercial as well as academic FPGAs and to better understand the relative merits and demerits of each cell. For this purpose, we modeled the architectures at RTL level using Aldec’s Active-HDL and then implemented benchmarks on them. The data collected revealed that, as the number of logic blocks required for a given design increased, the multiplexer based logic cells performed much slower than the look-up-table based logic cells. We found out that among all the tested architectures, the Xilinx Spartan II was the fastest and has the highest functionality.
Our analysis pointed that the lack of speed in the fine grained cells was directly related to their functionality. This was further tested with the Actel Proasicplus logic cell. Three new architectures were proposed whose logical abilities were better than the Proasicplus. When tested with the same benchmarks again, these new cells proved to be much faster and were more area efficient when compared to the original Proasicplus. Extending our analysis to transistor level, these cells were laid out in MAGIC using the TSMC 0.18μm technology file with $\lambda = 0.09\mu m$. It was then observed that, although all the new blocks were faster, the Mod3 achieved the best area delay product.

5.2 Future Work

In future it would be interesting to extend the transistor level analysis to estimate power. With increasing need for low power, low energy devices, an analysis of power consumption and dissipation could be very fruitful for future programmable solutions.

The other possible research area is to improve logic block architectures for specific applications. For example, the DSP LM was designed for DSP related applications. However due to the lack of a memory element in the logic block, this architecture fails to impress when used for the implementation of pipelined designs. Hence, future work could be concentrated on improving
these architectures so that the gap between FPGAs and ASICs could be reduced at least for some specific cases.

Finally, it would be beneficial to evaluate the logic cells using the latest CAD tools. One of the major hurdles in this study was the lack of these tools. Hence, in future with our synthesizable VHDL codes, other tools can be integrated to perform tasks like technology mapping as well as placement and routing. This will enable us to emulate the architectures in a better way thereby more meaningful results can be obtained. Also, CAD support will help us in implementing bigger and more complex benchmarks for performance evaluation.
APPENDIX A

CONFIGURATIONS SCHEMATICS OF ACTEL

PROASICPLUS LOGIC BLOCK

Figure A1: 2-AND Gate using Actel's Proasicplus Logic Block

Figure A2: 2-OR Gate using Actel's Proasicplus Logic Block
Figure A3: 2-XOR Gate using Actel's Proasicplus Logic Block

Figure A4: 3-AND Gate using Actel's Proasicplus Logic Block
Figure A5: 3-NOR Gate using Actel Proasicplus Logic Block

Figure A6: D Flip-Flop using Actel's Proasicplus Logic Block
APPENDIX B

CONFIGURATIONS SCHEMATICS OF MOD1 ARCHITECTURE

Figure B1: 2-AND Gate using Mod1 architecture

Figure B2: 2-OR Gate using Mod1 architecture
Figure B3: 2-XOR Gate using Mod1 architecture

Figure B4: 3-AND Gate using Mod1 architecture
Figure B5: 3-NOR Gate using Mod1 architecture

Figure B6: 3-NAND Gate using Mod1 architecture
Figure B7: 3-OR Gate using Mod1 architecture
APPENDIX C

CONFIGURATIONS SCHEMATICS OF MOD2 LOGIC BLOCK

Figure C1: 2-AND Gate using Mod2 architecture
Figure C2: 2-OR Gate using Mod2 architecture

Figure C3: 2-XOR Gate using Mod2 architecture
Figure C4: 3-AND Gate using Mod2 architecture

Figure C5: 3-NOR Gate using Mod2 architecture

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure C6: 3- NAND Gate using Mod2 architecture

Figure C7: 3- OR Gate using Mod1 architecture
Figure C8: 3-XOR Gate using Mod2 architecture

Figure C9: 4-AND Gate using Mod2 architecture

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure C10: 4-NOR Gate using Mod2 architecture

Figure C11: 4-NAND Gate using Mod2 architecture
Figure C12: 4-OR Gate using Mod2 architecture

Figure C13: 5-AND Gate using Mod2 architecture

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure C14: 5-NOR Gate using Mod2 architecture

Figure C15: 5-NAND Gate using Mod2 architecture

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure C16: 5-OR Gate using Mod2 architecture

Figure C17: D Flip Flop using Mod2 architecture
APPENDIX D

CONFIGURATIONS SCHEMATICS OF MOD3 LOGIC BLOCK

Figure D1: 2-AND Gate using Mod3 architecture
Figure D2: 2-OR Gate using Mod3 architecture

Figure D3: 2-XOR Gate using Mod3 architecture

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure D4: 3-AND Gate using Mod3 architecture

Figure D5: 3-NOR Gate using Mod3 architecture

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
Figure D6: 3-NAND Gate using Mod3 architecture

Figure D7: 3-OR Gate using Mod3 architecture
Figure D8: 3-XOR Gate using Mod3 architecture

Figure D9: 4-AND Gate using Mod3 architecture
Figure D10: 4-NOR Gate using Mod3 architecture

Figure D11: 4-NAND Gate using Mod3 architecture
Figure D12: 4-OR Gate using Mod3 architecture

Figure D13: 5-AND Gate using Mod3 architecture
Figure D14: 5-NOR Gate using Mod3 architecture

Figure D15: 5-NAND Gate using Mod3 architecture
Figure D16: 5-OR Gate using Mod3 architecture

Figure D17: D Flip Flop using Mod3 architecture
BIBLIOGRAPHY


118

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.


[22] J.S. Rose, R.J. Francis, P. Chow, and D. Lewis, "The Effect of Logic Block Complexity on Area of Programmable Gate Arrays", in the *IEEE Custom Integrated Circuits Conference*, San Diego, May 1989, pp. 5.3.1 - 5.3.5.


120


[35] Actel Datasheets

[36] Altera Datasheets

[37] Quicklogic Datasheets

[38] Xilinx Datasheets


122

Reproduced with permission of the copyright owner. Further reproduction prohibited without permission.
VITA

Graduate College
University of Nevada, Las Vegas

Rohith Ramnath

Home Address:
1600, E. Rochelle Ave., Apt 42
Las Vegas, NV-89119.

Degrees:
Bachelor of Engineering, Electronics and Communication Engineering,
2001, University of Mysore, India

Thesis Title:
Analysis of logic block architectures and functional improvement of
fine grained cells.

Thesis Examination Committee:
Chairperson, Dr. Yingtao Jiang, Ph. D.
Committee Member, Dr. Emma Regentova, Ph. D.
Committee Member, Dr. Venkatesan Muthukumar, Ph. D.
Graduate College Representative, Dr. Ajit Roy, Ph. D.