High speed VLSI architectures for a digital image threshold selection algorithm.

Peng Keong Lim

University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation
https://scholar.uwindsor.ca/etd/2301

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000 ext. 3208.
NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, tests publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, R.S.C. 1970, c. C-30.
HIGH SPEED VLSI ARCHITECTURES
FOR A
DIGITAL IMAGE THRESHOLD SELECTION ALGORITHM

by

Peng Keong, Lim

A Thesis
Submitted to the Faculty of Graduate Studies
through the Department of
Electrical Engineering in Partial Fulfillment
of the Requirements for the Degree
of Master of Applied Science at
The University of Windsor

Windsor, Ontario, Canada
1987
Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmier cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

To my Family Members
ABSTRACT

The main purpose of digital image thresholding is to isolate the "objects" from their "background". This is a step usually required for many of the digital image analysis applications. In this thesis, an iterative threshold selection scheme is studied and its VLSI implementation is presented. This algorithm is simple, fast, requires less hardware, and gives a comparable result compared to other techniques.

A few high speed architectures for fixed-point division are developed based on an approximate fast log and antilog algorithm to improve the speed of the proposed threshold selection architecture. These fast log architectures can be utilized in applications where high speed is required, and a small percentage error can be tolerated. Numerical implementation, limitations and examples are presented.

The gate matrix layout technique is utilized in this thesis to optimize the silicon area. This layout technique not only reduces the design time, but also the number of errors introduced to the layout.

The schematic of each leaf-cell is discussed and the layouts for the leaf-cells are presented. The leaf-cells are simulated using LOGCAP and SPICE, and the results are provided. Finally, possible extensions to the algorithm are suggested.
ACKNOWLEDGMENTS

I would like to express my sincere thanks and gratitude to my supervisor, Dr. G. A. Jullien for his valuable suggestions and guidance during the course of this research. I also like to express my heartfelt appreciation to my co-supervisor, Dr. M. A. Sid-Ahmed for his advice, help and guidance throughout the course of this research. The comments of the other committee members are gratefully acknowledged. Additionally, I sincerely appreciate the graduate students and others for their help.

My sincerest thanks to my Family Members, without their support and love, this work would not have been accomplished.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter/Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>v</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS</td>
<td>vi</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>vii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>xi</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>xiv</td>
</tr>
<tr>
<td><strong>CHAPTER 1: INTRODUCTION</strong></td>
<td></td>
</tr>
<tr>
<td>1.1 Overview of Very Large Scale Integrated Circuit</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Overview of Enhancement Mode Transistor</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Overview of Complementary MOS (CMOS)</td>
<td>6</td>
</tr>
<tr>
<td>1.4 Appropriate Approach of VLSI Design</td>
<td>7</td>
</tr>
<tr>
<td>1.5 The Application of VLSI on Digital Image Processing</td>
<td>9</td>
</tr>
<tr>
<td>1.6 Digital Image Thresholding</td>
<td>10</td>
</tr>
<tr>
<td>1.7 Histogram</td>
<td>10</td>
</tr>
<tr>
<td>1.8 Summary</td>
<td>11</td>
</tr>
<tr>
<td><strong>CHAPTER 2: SYMBOLIC LAYOUT AND COMPUTER SIMULATION</strong></td>
<td>13</td>
</tr>
<tr>
<td>2.1 Introduction</td>
<td>13</td>
</tr>
<tr>
<td>2.2 Gate Matrix Layout Method For MOS VLSI</td>
<td>14</td>
</tr>
<tr>
<td>2.2.1 Automated Gate Matrix Layout</td>
<td>17</td>
</tr>
<tr>
<td>2.3 Circuit Simulation</td>
<td>24</td>
</tr>
<tr>
<td>2.3.1 SPICE 2: A Computer Program to Simulate Semiconductor Circuits</td>
<td>25</td>
</tr>
<tr>
<td>2.3.2 Circuit Layout and Its Associated Capacitances and Resistances</td>
<td>26</td>
</tr>
<tr>
<td>2.3.3 Drain, Source and Gate Capacitances Approximation</td>
<td>27</td>
</tr>
</tbody>
</table>
2.3.4 Routing Capacitances 29
2.3.5 Routing Resistance 35
2.3.6 Interconnect Discontinuities 32
2.3.7 Diffusion Resistance 33
2.3.8 Contact Resistance 34
2.3.9 Defining Subcircuit 35
2.3.10 Power and Ground Routing 37
2.3.11 Modeling Long Interconnects 38
2.3.12 The Concept of Equivalent Load Capacitance 40
2.3.13 Driving Large Capacitive Loads 44
2.4 Summary 45

CHAPTER 3: AN ARCHITECTURE FOR A DIGITAL IMAGE THRESHOLD SELECTION ALGORITHM 46

3.1 Introduction 46
3.2 The Hardware Structure 47
3.3 Fast Algorithm for Digital Division 50
3.4 Hardware Architecture for Fast Log Divider 57
  3.4.1 Architecture 1 58
  3.4.2 Architecture 2 67
  3.4.3 Special Architecture: Architecture 3 67
3.5 Software Simulation for Fast Log Divider 69
3.6 Software Simulation of Threshold Selection Architecture 69
3.7 Examples 71
3.8 Summary 76

CHAPTER 4: SUBSYSTEM DESIGN AND CIRCUIT SIMULATION 77

4.1 Introduction 77
4.2 Combination Adder 79
  4.2.1 Serial Adder 80
  4.2.2 Parallel Adder 81
4.3 Carry-Ripple Adder 82
  4.3.1 Static Adder 82
  4.3.2 Dynamic Adder 84
  4.3.3 Transmission Gate And Carry-Save Adders 87
4.4 Carry Look-ahead Adders 88
  4.4.1 Speed Up The Worst Case Delay 90
  4.4.2 Carry Select Adder 93
  4.4.3 Functional Simulation Of The Manchester Adder 93
4.5 Subtractor 96
  4.5.1 VLSI Layouts of Adders and Subtractors 96
4.6 Fast Comparator 103
4.7 Multiplexer 107
4.8 Synchronous Counter 109
4.9 Latch 120
4.10 Barrel Shifter 124
4.11 Layout of The Fast Log Divider and The Threshold Selection Hardware 127
4.12 Summary 127

CHAPTER 5: EXTENSIONS AND CONCLUSIONS 130
5.1 Introduction 130
5.2 Limitation 131
5.3 Possible Extensions and Improvements 131
  5.3.1 Square-Root Circuit 131
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.3.2 The CCMOS Latches</td>
<td>133</td>
</tr>
<tr>
<td>5.3.3 Double Metal CMOS</td>
<td>133</td>
</tr>
<tr>
<td>5.3.4 Parallel Decoding Counter</td>
<td>134</td>
</tr>
<tr>
<td>5.3.5 Circuit Modifications</td>
<td>135</td>
</tr>
<tr>
<td>5.4 Conclusion</td>
<td>137</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>139</td>
</tr>
<tr>
<td>APPENDIX I: FORTRAN PROGRAM FOR AUTOMATIC THRESHOLD</td>
<td>147</td>
</tr>
<tr>
<td>SELECTION ALGORITHM</td>
<td></td>
</tr>
<tr>
<td>APPENDIX II: FORTRAN PROGRAM FOR FAST LOG ERROR</td>
<td>149</td>
</tr>
<tr>
<td>ANALYSIS</td>
<td></td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Enhancement Mode Transistor</td>
<td>3</td>
</tr>
<tr>
<td>1.2</td>
<td>Channel Form at Enhancement Mode Transistor</td>
<td>4</td>
</tr>
<tr>
<td>1.3</td>
<td>Nonsaturation Mode</td>
<td>5</td>
</tr>
<tr>
<td>1.4</td>
<td>Saturation Mode</td>
<td>5</td>
</tr>
<tr>
<td>1.5</td>
<td>CMOS Inverter</td>
<td>6</td>
</tr>
<tr>
<td>1.6</td>
<td>Gray Level Histogram</td>
<td>11</td>
</tr>
<tr>
<td>2.1</td>
<td>Gate Matrix Representation of the Comparator</td>
<td>16</td>
</tr>
<tr>
<td>2.2</td>
<td>Layout of The Comparator</td>
<td>17</td>
</tr>
<tr>
<td>2.3</td>
<td>Schematic of N-part Comparator</td>
<td>18</td>
</tr>
<tr>
<td>2.4</td>
<td>Comparator Realized With Six Tracks</td>
<td>19</td>
</tr>
<tr>
<td>2.5</td>
<td>Interval Graph of Figure 1.4</td>
<td>20</td>
</tr>
<tr>
<td>2.6</td>
<td>Connection Graph of The Comparator</td>
<td>21</td>
</tr>
<tr>
<td>2.7</td>
<td>The Final Interval Graph</td>
<td>23</td>
</tr>
<tr>
<td>2.8</td>
<td>A MOS Transistor</td>
<td>27</td>
</tr>
<tr>
<td>2.9</td>
<td>Wire Capacitance: Parameters</td>
<td>29</td>
</tr>
<tr>
<td>2.10</td>
<td>Coupling Capacitance</td>
<td>30</td>
</tr>
<tr>
<td>2.11</td>
<td>Interconnect Discontinuities</td>
<td>33</td>
</tr>
<tr>
<td>2.12</td>
<td>Example of 2 Transistors Sharing A Common Node</td>
<td>35</td>
</tr>
<tr>
<td>2.13</td>
<td>Equivalent Circuit of Figure 2.12</td>
<td>36</td>
</tr>
<tr>
<td>2.14</td>
<td>Model For Vss and Vdd Lines Which Includes Bounding Wires</td>
<td>37</td>
</tr>
<tr>
<td>2.15</td>
<td>Ladder Equivalent Circuit for Long Wire</td>
<td>39</td>
</tr>
<tr>
<td>2.16</td>
<td>Segmentation of Long Polysilicon Line</td>
<td>40</td>
</tr>
<tr>
<td>2.17</td>
<td>Layout of Two Cascaded Transistors</td>
<td>41</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>2.18</td>
<td>Cascaded Inverters</td>
<td>44</td>
</tr>
<tr>
<td>3.1</td>
<td>Histogram Distribution</td>
<td>47</td>
</tr>
<tr>
<td>3.2</td>
<td>The Block Diagram of The Proposed Architecture</td>
<td>49</td>
</tr>
<tr>
<td>3.3</td>
<td>Linear Interpolation</td>
<td>51</td>
</tr>
<tr>
<td>3.4</td>
<td>Architecture 1: 4-Bit Divider</td>
<td>58</td>
</tr>
<tr>
<td>3.5</td>
<td>Schematic of Decoder-1</td>
<td>61</td>
</tr>
<tr>
<td>3.6</td>
<td>Schematic of Decoder-2</td>
<td>61</td>
</tr>
<tr>
<td>3.7</td>
<td>Schematic of Decoder-3</td>
<td>64</td>
</tr>
<tr>
<td>3.8</td>
<td>Schematic of Decoder-4</td>
<td>64</td>
</tr>
<tr>
<td>3.9</td>
<td>Architecture 2: Divider for a Large Number of Bits</td>
<td>66</td>
</tr>
<tr>
<td>3.10</td>
<td>Architecture 3: Special Purpose Divider</td>
<td>68</td>
</tr>
<tr>
<td>3.11</td>
<td>Fast Log Error Analysis</td>
<td>70</td>
</tr>
<tr>
<td>3.12</td>
<td>The Original Image of a Bridge</td>
<td>71</td>
</tr>
<tr>
<td>3.13</td>
<td>The Histogram of The Bridge Image</td>
<td>72</td>
</tr>
<tr>
<td>3.14</td>
<td>The Image After Thresholding</td>
<td>72</td>
</tr>
<tr>
<td>3.15</td>
<td>An Image With Surface Flaws</td>
<td>73</td>
</tr>
<tr>
<td>3.16</td>
<td>Histogram Distribution of Figure 3.14</td>
<td>74</td>
</tr>
<tr>
<td>3.17</td>
<td>Thresholded Image (1st. Optimum Threshold Level)</td>
<td>74</td>
</tr>
<tr>
<td>3.18</td>
<td>Thresholded Image (2nd. Optimum Threshold Level)</td>
<td>75</td>
</tr>
<tr>
<td>3.19</td>
<td>Thresholded Image (3rd. Optimum Threshold Level)</td>
<td>75</td>
</tr>
<tr>
<td>4.1</td>
<td>Block Diagram of The Threshold Selection</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Architecture</td>
<td></td>
</tr>
<tr>
<td>4.2</td>
<td>Logic Diagram of a Full Adder</td>
<td>78</td>
</tr>
<tr>
<td>4.3</td>
<td>Logic Diagram of a Serial Adder</td>
<td>80</td>
</tr>
<tr>
<td>4.4</td>
<td>Parallel Adder</td>
<td>81</td>
</tr>
<tr>
<td>4.5</td>
<td>Fully Static Adder</td>
<td>82</td>
</tr>
<tr>
<td>4.6</td>
<td>Dynamic Adder</td>
<td>83</td>
</tr>
</tbody>
</table>

xii
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.7</td>
<td>Domino Adder</td>
<td>85</td>
</tr>
<tr>
<td>4.8</td>
<td>Pseudo NMOS Full Adder</td>
<td>85</td>
</tr>
<tr>
<td>4.9</td>
<td>NOR Adder</td>
<td>86</td>
</tr>
<tr>
<td>4.10</td>
<td>Zipper CMOS Adder</td>
<td>86</td>
</tr>
<tr>
<td>4.11</td>
<td>Carry Save Adder</td>
<td>87</td>
</tr>
<tr>
<td>4.12</td>
<td>Four-Bit Carry Look-Ahead Circuits by Hewlett Packard</td>
<td>88</td>
</tr>
<tr>
<td>4.13</td>
<td>Optimum Layout Via Functional Cell Realization</td>
<td>89</td>
</tr>
<tr>
<td>4.14</td>
<td>Manchester Carry Chain</td>
<td>89</td>
</tr>
<tr>
<td>4.15</td>
<td>Four-Bit Manchester Carry Chain</td>
<td>90</td>
</tr>
<tr>
<td>4.16</td>
<td>Improved Manchester Look-Ahead Circuitry</td>
<td>91</td>
</tr>
<tr>
<td>4.17</td>
<td>Schematic of Manchester Carry Look-Ahead Adder</td>
<td>92</td>
</tr>
<tr>
<td>4.18</td>
<td>Carry Select Adder</td>
<td>93</td>
</tr>
<tr>
<td>4.19</td>
<td>Schematic and Simulated Results of the Carry-Propagate and Carry-Generate Stages</td>
<td>94</td>
</tr>
<tr>
<td>4.20</td>
<td>Schematic and Simulated Results of the SUM Stage</td>
<td>95</td>
</tr>
<tr>
<td>4.21</td>
<td>Simulated Results of a 4-Bit Look-Ahead Adder</td>
<td>97</td>
</tr>
<tr>
<td>4.22</td>
<td>Schematic of a 4-Bit Subtractor</td>
<td>98</td>
</tr>
<tr>
<td>4.23</td>
<td>Layout of a Carry-Generate and Carry-Propagate Circuitry</td>
<td>99</td>
</tr>
<tr>
<td>4.24</td>
<td>SPICE Simulation of the Carry-Generate Stage</td>
<td>99</td>
</tr>
<tr>
<td>4.25</td>
<td>SPICE Simulation of the Carry-Propagate Stage</td>
<td>100</td>
</tr>
<tr>
<td>4.26</td>
<td>SUM Stage of The Adder</td>
<td>100</td>
</tr>
<tr>
<td>4.27</td>
<td>SPICE Simulation of the SUM Stage</td>
<td>101</td>
</tr>
<tr>
<td>4.28</td>
<td>Layout of the 4-Bit Manchester Carry Chain</td>
<td>101</td>
</tr>
<tr>
<td>4.29</td>
<td>Layout of the 8-Bit Manchester Carry Look-Ahead Carry Select Adder</td>
<td>102</td>
</tr>
<tr>
<td>4.30</td>
<td>Layout of the 8-Bit Carry Look-Ahead Adder</td>
<td>103</td>
</tr>
<tr>
<td>Figure</td>
<td>Page</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>------</td>
<td></td>
</tr>
<tr>
<td>4.31</td>
<td>CMOS Comparator</td>
<td>105</td>
</tr>
<tr>
<td>4.32</td>
<td>Schematic and Simulated Result for a 4-Bit Comparator</td>
<td>106</td>
</tr>
<tr>
<td>4.33</td>
<td>Layout of a 2-Bit Comparator</td>
<td>107</td>
</tr>
<tr>
<td>4.34</td>
<td>Two Way Multiplexer</td>
<td>108</td>
</tr>
<tr>
<td>4.35</td>
<td>VLSI Layout of a Multiplexer</td>
<td>108</td>
</tr>
<tr>
<td>4.36</td>
<td>Gate Schematic of a 4-Bit Up Counter</td>
<td>109</td>
</tr>
<tr>
<td>4.37</td>
<td>Static D_Flip-Flop</td>
<td>110</td>
</tr>
<tr>
<td>4.38</td>
<td>Transmission Gate Static Flip-Flop</td>
<td>110</td>
</tr>
<tr>
<td>4.39</td>
<td>Stage 1 of The Counter</td>
<td>112</td>
</tr>
<tr>
<td>4.40</td>
<td>Stage 2 of The Counter</td>
<td>113</td>
</tr>
<tr>
<td>4.41</td>
<td>Layout of Stage 1</td>
<td>114</td>
</tr>
<tr>
<td>4.42</td>
<td>SPICE Simulation of Stage 1</td>
<td>114</td>
</tr>
<tr>
<td>4.43</td>
<td>VLSI Layout of Stage 2</td>
<td>115</td>
</tr>
<tr>
<td>4.44</td>
<td>Layout of The 16-Bit Counter</td>
<td>116</td>
</tr>
<tr>
<td>4.45</td>
<td>Gate Schematic of a 4-Bit Down-Counter</td>
<td>117</td>
</tr>
<tr>
<td>4.46</td>
<td>Stage 1 of The Down-Counter</td>
<td>118</td>
</tr>
<tr>
<td>4.47</td>
<td>Stage 2 of The Down-Counter</td>
<td>118</td>
</tr>
<tr>
<td>4.48</td>
<td>Layout of a 5_BIT Down-Counter</td>
<td>119</td>
</tr>
<tr>
<td>4.49</td>
<td>Layout of a 4-Bit Down-Counter</td>
<td>120</td>
</tr>
<tr>
<td>4.50</td>
<td>Static Master-Slave Latch</td>
<td>120</td>
</tr>
<tr>
<td>4.51</td>
<td>CCMOS Latch</td>
<td>121</td>
</tr>
<tr>
<td>4.52</td>
<td>Layout of a Static Latch</td>
<td>121</td>
</tr>
<tr>
<td>4.53</td>
<td>Layout of a CCMOS Latch</td>
<td>123</td>
</tr>
<tr>
<td>4.54</td>
<td>Schematic and Simulation of a Barrel Shifter</td>
<td>125</td>
</tr>
<tr>
<td>4.55</td>
<td>Layout of an 8-Bit Barrel Shifter</td>
<td>126</td>
</tr>
<tr>
<td>4.56</td>
<td>Layout of The Fast Log Divider</td>
<td>128</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>--------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.57</td>
<td>The Complete Layout of The System</td>
<td>129</td>
</tr>
<tr>
<td>5.1</td>
<td>The Schematic of a CCMOS Latch</td>
<td>132</td>
</tr>
<tr>
<td>5.2</td>
<td>The VLSI Layout of a CCMOS Latch</td>
<td>132</td>
</tr>
<tr>
<td>5.3</td>
<td>CMOS Implementation of a 7-Input AND Gate</td>
<td>134</td>
</tr>
<tr>
<td>5.4</td>
<td>Architecture For Square-Root</td>
<td>136</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>20</td>
</tr>
<tr>
<td>2.2</td>
<td>22</td>
</tr>
<tr>
<td>2.3</td>
<td>32</td>
</tr>
<tr>
<td>3.1</td>
<td>59</td>
</tr>
<tr>
<td>3.2</td>
<td>62</td>
</tr>
<tr>
<td>3.3</td>
<td>63</td>
</tr>
<tr>
<td>3.4</td>
<td>63</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION

1.1 Overview of Very Large Scale Integrated Circuit

Today, very large scale integration (VLSI) has a great effect on consumer, industrial, and military electronics. VLSI systems are particularly well suited for high volume applications where increased functionality at reduced cost provides a key comparative advantage.

Historical evidence indicates that chip complexity has grown exponentially since 1960. This rapid rate of growth was observed and predicted by Gordon Moore of Intel back in 1979 [1]. Carver A. Mead from Caltech pictured that if the spacing between conductors in a typical chip is scaled to one city block, the circuit can then be thought of as a multi-level road network. In the mid 1960's, the complexity of a chip was comparable to that of the street network of a small town. Most people can navigate such a network by memory without difficulty. In the late 1970's, many microprocessors were introduced to the market, the complexity of those microprocessors were comparable to the entire Los Angeles basin. Mead was predicted that the complexity of a chip design will be capable of planning a street network covering all of California and Nevada at urban densities when 1 um technology is solidly in place and the ultimate 1/4 um technology will be capable of producing chips whose complexity rivals an urban network covering the entire North American continent [2,71].
Here, in Windsor, very large scale integrated (VLSI) circuits are currently fabricated through the Canadian Microelectronics Corporation (CMC) using MOS technology. MOS transistors are either p-channel (pMOS) or n-channel (nMOS). In this thesis, a popular MOS technology called complementary-symmetry MOS (CMOS) is applied to the proposed logic circuits. CMOS circuits use both n-channel and p-channel devices on the same IC chip. The CMOS integrated circuits fabricated through CMC use transistors of the enhancement type. In the next section, the enhancement mode transistor action will be discussed.

1.2 Overview of Enhancement Mode Transistor

The normal conduction characteristics of a MOS transistor can be categorized as cut-off, nonsaturation, and saturation regions.

Figure 1.1 shows the structure of an n-channel enhancement type MOS transistor. nMOS devices are formed in a p-type substrate doping level. The drain and source regions are formed by two heavily doped n regions. A polysilicon gate is deposited on a layer of insulation over the region between source and drain. If the gate is left floating, or if $V_d = V_s = V_{gs} = 0$, the path from the drain to source includes two series diodes back-to-back, which means that no current can flow. To cause current to flow from drain to source we first have to create an n-channel. With the applied positive
Figure 1.1: Enhancement Mode Transistor

Voltage $V_{gs}$, electrons from the substrate are attracted and accumulated at the surface beneath the oxide layer. The applied voltage $V_{gs}$ has to be equal to or greater than a threshold value $V_t$ in order to form an n-channel. This means that no appreciable current $I_{ds}$ will flow until $V_{gs}$ is greater than $V_t$ and $V_{ds}$ is positive. Increasing $V_{gs}$ will cause the created channel to deepen (become enhanced) as shown in Figure 1.2.

Now, if we let $V_{gs} = V_t$, and increase $V_{ds}$, the resistive drop along the channel causes the voltage between gate and channel to vary with distance along the channel. Since $V_{gs}$ is constant, the channel will remain at a constant depth at the source end. However, at the drain end, the effective voltage is equal to the difference between the gate and the drain, or $V_{gd}$. Hence, the channel will have a tapered
Figure 1.2: Channel Form At Enhancement Mode Transistor

shape, being deepest at the source end and shallowest at the drain end, as shown in Figure 1.3. For all voltages \( V_{ds} \leq V_{gs} - V_t \), the device is in the nonsaturated region of operation, where the channel current \( I_{ds} \) is a function of both gate and drain voltages.

It follows that increasing \( V_{ds} \) will cause the channel resistance to increase, when \( V_{ds} \) is increased to a level greater than \( V_{gs} - V_t \), the resistive drop \( V_{gs} - V_t \) takes place over less than the whole length of the channel and the channel width approaches zero at the drain end. This pinch-off condition is shown in Figure 1.4. Diffusion current completes the path from source to drain in this case, causing the channel to exhibit a high resistance and behave as a constant current source. This region, known as saturation, is characterized by almost constant current for any increase of \( V_{ds} \) above \( V_{ds} = V_{gs} - V_t \) [6,7,8,9].
Figure 1.3: Nonsaturation Mode

Figure 1.4: Saturation Mode
1.3 Overview of Complementary MOS (CMOS)

CMOS, stands for complementary MOS. A CMOS circuit utilizes both nMOS and pMOS devices. In the case of fully static CMOS, when pMOS is conductive, nMOS is nonconductive and vice-versa. Figure 1.5 illustrates the static inverter. The source of the p-device is connected to the positive terminal of the power supply (Vdd), and the source of the n-device is connected to the negative terminal (usually ground) of the power supply (Vss).

![CMOS Inverter Diagram]

Figure 1.5: CMOS Inverter

When the input is high (5V), the nMOS transistor is conductive and the pMOS transistor is not conductive. When the input is low (0V), the pMOS transistor is conductive and the nMOS transistor is not conductive. Since only one of the devices conducts for static logic level, no current flows from Vdd to Vss (except some very small leakage current).
1.4 Appropriate Approach to VLSI Design.

It is very important, if not compulsory, for a VLSI designer to know how to design large complex systems in a reasonable time and to expend reasonable effort yet obtain an acceptable design. It has been reported that even in the case of a relatively straightforward random logic circuit with only 500 transistors, it will take two man-months to design and implement the circuit [7]. For a rough approximation, the proposed image thresholding chip involves more than 6000 transistors. If a linear relationship exists between complexity and design time, this chip will require 24 man-months in designing and implementing alone. Needless to say, background preparation and literature surveys are required prior to design and implementation. Hence, at least 3 full years should ordinarily be required for a designer to complete this project.

Fortunately, some systematic and carefully chosen architectures will speed-up the design time as well as reduce the implementation area. The guidelines may be as follows:

(i). In any engineering design, the objective of designing the system has to be defined first. This is the most important and the biggest step, hence, a careful consideration of the system requirements are required.

(ii). If the system is large, subsystems partitioning is required. System partitioning will simplify complex systems providing the subsystems have minimum interdependence and
complexity of interconnection.

(iii). A most acute problem in the design of large systems is the interconnections. The communication paths may take up as much as 40% of the chip area even if the design is straightforward [7]. Therefore, careful consideration of the interrelationship between subsystems should be given the highest priority early in the design process and a communication strategy should be evolved and adhered to throughout the design process.

(iv). The design process starts by setting out a floorplan of block diagrams with main interconnection routes among the subsystems.

(v). If some of the first level subsystems are large, they may be able to be partitioned again into second level subsystems. In this case, floorplans are also required for the subsystems in order to show how to map the subsystems onto silicon.

(vi). Design the conventional circuit symbols and/or logic symbols for the subsystems (i.e. Adders, Flip-flops, AND gates, OR gates, etc.). It is very important to design the structure to be as regular as possible so that the design is largely a matter of duplication.

(vii). Carry out the schematic simulation to verify the functions of the subsystem (i.e. via SPICE and/or LOGCAP).

(viii). Draw suitable (stick) diagrams of the leaf cell of the subsystems (in this thesis, a Gate Matrix diagram approach is followed and will be discussed in Chapter 2).
Care must be taken in allocating the interconnection routes (i.e. inputs/outputs, VDD, VSS paths) within and among the leaf cells.

(ix). As the design progresses, an interactive process of modification will take place between stages (ii) to (viii).

(x). By using a CAD Workstation, convert each cell to a layout. Bear in mind that each library-cell that is created has to be centered to an accuracy of 1 μm, for CMC requirements.

(xi). Obtain the CIF code from the workstation, and carry out a design rule check on each cell.

(xii). Carry out the layout level simulation by SPICE. All the actual physical dimension of the transistors and their interconnections have to be provided as SPICE input parameters in order to obtain correct results.

1.5 The Applications of VLSI on Digital Image Processing

Recently, a shift of direction has been observed in the design of commercial vision systems for robot control and industrial inspection. The development of fully automatic vision systems for applications such as object identification and pattern recognition is a great challenge to the robotic industries. Until very recently, a primary difficulty in the development of such systems has been the hardware complexities. Solutions to this problem lie in the current development of very large scale integration (VLSI) microelectronics. In this thesis, a possible solution to an
automatic image threshold selection algorithm is proposed in the VLSI design context.

1.6 Digital Image Thresholding

The purpose of image thresholding is to partition a digital image into disjoint (nonoverlapping) regions. Thresholding is a particularly useful region approach technique for scenes containing solid objects resting upon a contrasting background. When using thresholding for image segmentation, a threshold level has to be determined first. All pixels with gray levels below the threshold fall into the background and all pixels with gray levels higher than or equal to the threshold fall into the object and vice-versa [3,4,5].

1.7 Histogram.

One of the simplest and most useful tools in digital image processing is the gray level histogram. As shown in Figure 1.6, the gray level histogram is a function showing, for each gray level, the number of pixels in the image that have that gray level.

If the histogram is bimodal, it suggests that the image consists of two populations. If this is the case, it would be simple to threshold the image at a level that would separate the two populations. All pixels in the image below the selection threshold level would be rendered black, while those above that level would become white, thus reducing the multilevel image into a binary one.

The problem of threshold selection becomes a little
more complex, however, when the histogram is not clearly bimodal. In such a case, it is difficult to select one level that would separate the image into the "object" and "background".

A possible solution to this problem is to utilize an automatic threshold selection algorithm. This algorithm and its hardware architecture will be discussed in Chapter 3 of this thesis.

![Figure 1.6: Gray Level Histogram](image)

1.8 Summary

An introduction to the VLSI design and its applications on digital image processing was presented in this chapter. The cut-off, non-saturation, and saturation modes of an enhancement type transistor were studied. The appropriate approach to VLSI design was summarized in section 4 of this chapter. In the last portion of this chapter, the
overview of the digital image thresholding and the histogram of an image were presented.
CHAPTER 2

SYMBOLIC LAYOUT AND COMPUTER AIDED SIMULATION

2.1 Introduction

With the rapid evolution of VLSI technologies, the design of integrated circuits is becoming increasingly complex and difficult. Among the various design phases of VLSI, "layout" accounts for a major portion of design turn-around-time. A typical example is the Z8000 microprocessor where approximately 6,600 man-hours (3 man-years) of layout time was required. As a result, approximately 13,000 man-hours was needed to produce this chip [72,73]. Today, many research institutions such as the research labs of AT&T have launched advances in CAD for VLSI. The editor of the series on Advances CAD for VLSI, T. Ohtsuki, has noted that this is a period when no-one can be a specialist in all of the topics in CAD for VLSI, and the whole area is beyond the scope of a single volume.

In this chapter, Gate Matrix symbolic layout technique developed in AT&T is studied. This technique offers the advantages of hand-packed mask design with regards to density of layout, while also having advantages over manual layout with respect to the design time of a circuit and reduction in the number of errors introduced into a design.

Where the design process has not been fully automated, the designer must normally check for functional error by simulating the operation of his circuit or system. A method of SPICE simulation is described in this chapter. The circuit
layout and its associated capacitances and resistances such as drain, gate, source capacitances, routing capacitances, routing resistance, diffusion resistance, contact resistance and the effect of power and ground routing is studied.

2.2 Gate Matrix Layout Method For MOS VLSI

In the past, the basic cells were designed manually by sketching the layout on quadrille paper and thereafter entering a description of the cells into the computer using a special purpose layout description language which defined the shapes, their mask levels and coordinates. More recently, however, basic cell design has been made efficient by using symbolic layout techniques. This enables transistors, contacts and interconnections to be represented by symbols which are subsequently replaced by the necessary geometries to realize these function on the layout.

Many symbolic layout methods have been proposed in the last 15 years. Rockwell International and American Microsystems International have made use of character-based symbolic layout [74,75,76]. Caltech and MIT used stick diagrams to convey layer information through the use of a color code [77,78,32]. Virtual grid symbolic layout method was reported by Weste [79,80]. Many symbolic layout systems also available in the literature for the design-rule free symbolic layout [81,82,83]. In this thesis, a well known Gate Matrix method developed in Bell-Laboratories is followed [84,85,86,66].

There are a number of considerations which need to be
taken into account in the development of a topological style which best serve the requirements for VLSI circuit layouts [84].

i. The style should have an ordered structure

ii. The topological style should not sacrifice very much silicon area at the expense of layout ease.

iii. The layout method should be adaptable to a team effort.

Frequently, to expedite the chip layout process, the layout task for a VLSI circuit is divided among several chip designers, each operating on different portions of the chip. Problems often arise when the different portions of the chip are finally combined. Thus the time savings derived from dividing the layout task may be offset by increased chip area usage caused by complexities of interconnection between the different portions of the chip.

The basic concept of the gate matrix style is to superimpose all transistors of the standard cell method onto the wiring channel so that all of the transistors that have a common input are placed on a common polysilicon line. This line acts both as the gate of the transistors and as the connection among all commonly gated transistors. The polysilicon lines, which are equally spaced and parallel, become the columns of the gate matrix. The rows are formed by grouping the diffusion marks of transistors which associate with one another either in a serial or parallel fashion.

The unique characteristics of the gate matrix approach
are summarized as follows [84]:

i. Polysilicon runs only in one direction and is constant in width and pitch.

ii. Diffusion runners exist between polysilicon columns.

iii. Metal runs in both directions and is constant in width except for power buses.

iv. Transistors can exist only on the polysilicon columns.

Figure 2.1: Gate Matrix Representation of a Comparator

Figures 2.1 and 2.2 show the gate matrix representation and the actual layout of a CMOS comparator.
2.2.1 Automated Gate Matrix Layout

Given a logic cell and a set of overlapping intervals, our problem is to place these intervals on as few tracks as possible. In order to make the gate matrix method more efficient, it is necessary to shuffle the input polysilicon lines until the best layout can be obtained (smallest silicon area).

In CMOS technology, the gates of each complementary pair of P and N transistors are connected together so that each such pair can be placed on the same gate (poly) line. Because of this, it is only necessary to consider either the P or the N transistor group. Once a layout of one group is done, the layout of the other group is automatically
determined.

In the last section, it was shown that the gate matrix has an abstract representation in which each poly is a vertical line and the diffusion areas and metal conductors are represented by horizontal line segments (nets) which are connected to the gate (poly) lines. An example is shown in Figure 2.3.

The abstraction of a gate matrix, layout of Figure 2.3 is shown in Figure 2.4 [87,88]. The layout requires six tracks. Since there are as many gate lines as there are transistors gates, minimization of the number of tracks is equivalent to minimization of area. A method with fewer tracks will now be discussed.

![Diagram](image)

Figure 2.3: Schematic of N-part Comparator
Figure 2.4: A Comparator Realized With Six Tracks

Before going further, we defined an interval graph $G(V, E)$ in which $V = \{ vi \mid vi \text{ is an interval} \}$ is the vertex set and $E = \{ (vi, vj) \mid \text{intervals } vi \text{ and } vj \text{ overlap} \}$ is the edge set.

If we draw horizontal intervals, which are drawn corresponding to nets, an interval graph corresponding to the gate sequence can be obtained. The set of vertices of the interval graph is the set of nets. The vertices are adjacent if and only if the corresponding intervals intersect when they are placed on the same track. It should be noted that the intervals are closed, and hence two intervals intersect each other even if their common point represents an end point.
of the respective interval. The interval graph which corresponds to the placement in Figure 2.4 is shown in Figure 2.5.

Figure 2.5: Interval Graph of Figure 2.4

The net list specification can be given by a table listing all the gates and the nets which are associated to each gate. The net list of Figure 2.4 is given as follows:

<table>
<thead>
<tr>
<th>Gate</th>
<th>Nets</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2,4,3</td>
</tr>
<tr>
<td>\overline{A}</td>
<td>1,6,5</td>
</tr>
<tr>
<td>B</td>
<td>1,6,3,5</td>
</tr>
<tr>
<td>\overline{B}</td>
<td>4,6,2,3</td>
</tr>
</tbody>
</table>

Table 2.1: The Net List

Based on this, a graph can be constructed, called a connection graph,

\[ H = (V,E) \]
where

$$V = \{ v_i \in V \mid v_i \text{ is a net} \} \quad \text{and}$$

$$E = \{ (v_i, v_j) \mid \text{a gate to which both } v_i \text{ and } v_j \text{ are connected} \}$$

For the example of Table 2.1, its connection graph is given in Figure 2.6.

![Connection Graph](image)

**Figure 2.6: Connection Graph of The Comparator**

A lower bound on the minimum number of tracks is found to be the largest number of net connected to any gate. In an undirected graph $G$, a set of vertices $C$ is called a clique if every two vertices of $C$ are connected by an edge. Let $n$ be a non-negative integer, then an $n$-clique is defined as a loopless graph with exactly $n$-vertices and $\frac{1}{2}n(n-1)$ edges, each pair of vertices being joint by a single link. The clique of an interval graph has its vertices corresponding to mutually overlapping intervals. A clique is called dominant.
if it is maximal, i.e., if it is not a proper subset of another clique. The dominant clique of Figure 2.5 has a size of 6 and Figure 2.6 has a size of 4. If the connection graph is an interval graph itself, then a layout is optimum since the clique number equals a lower bound on the minimum of tracks required. Therefore, one simple way of getting optimum tracks is to obtain a connection graph from the given schematic, then to find a subgraph by adding a set of edges, which is an interval graph and has the least clique number.

Let $G = (V,E)$ be an interval graph with dominant cliques $C_1, C_2, C_3, \ldots, C_p$. Then a dominant clique versus vertex matrix $A = [a_{ij}]$ is defined by

$$a_{ij} = \begin{cases} 1, & \text{if vertex } i \in C_j \\ 0, & \text{otherwise} \end{cases}$$

Next identify all the dominant cliques of $H$ and construct the dominant clique versus vertex matrix shown in Table 2.2.

$$A = \begin{bmatrix} 1 & 2 \\ 1 & 1 & 0 \\ 2 & 0 & 1 \\ 3 & 1 & 1 \\ 4 & 0 & 1 \\ 5 & 1 & 0 \\ 6 & 1 & 1 \end{bmatrix}$$

**Table 2.2: Dominant Clique Versus Vertex Matrix**
Since the dominant clique has a size of 4, which is equal to the lower bound on the minimum track required, the algorithm will be terminated here, and the interval graph can be constructed as shown in Figure 2.7. The gate matrix layout of the above graph is shown in Figure 2.1.

![Diagram of interval graph]

Figure 2.7: The Final Interval Graph.

Note that if we have more than 2 dominant cliques, a way to obtain an interval graph from a connection graph is simply to fill each row without the consecutive ones property with ones. Let $A'$ be the filled matrix. The maximum clique size of the interval graph derived from $A'$ is denoted as an upper bound on the maximum number of tracks required. If the
upper bound equals the lower bound, then an optimum solution is obtained. If not, more complex algorithms [87] can be followed to obtain the optimum result.

2.3 Circuit Simulation

From the sixties until now, the emphasis in computer-aided circuit analysis was a general purpose program. ECAP [90], which was one of the earliest widely available program, was replaced by many newer ones such as SPICE [91].

These simulators usually yield the analysis of transient behavior, direct-current performance, stationary alternating-current performance, temperature, signal distortion, noise interference, sensitivity, and parameter optimization of the electronic circuits [92].

While the design process simulation has a wide range of applications [93,94], this thesis has used it extensively for the following:

i. Timing analysis: Various delays exit in the circuit which may affect the performance. It is important that a timing analysis is performed on parts of the circuit whose delay may be critical to the operation of the circuit.

ii. Function verification: To further increase the simulation speed, it is necessary to abandon analogue waveform analysis and instead use some form of gate-level logic simulation.

iii. Sensitivity analysis: During a circuit level simulation of a cell, the circuit component values are made temperature sensitive, hence the effects of temperature upon
the circuit performance can be observed. Two main simulators are used in this project. Namely: the SPICE circuit simulation program and the LOGCAP logic level simulation program. Most of the schematics have been simulated using LOGCAP prior to chip layout procedure and can be found in the next chapter. In the next two sections, we are going to look at the procedure of using the SPICE simulator.

2.3.1 SPICE2: Simulation Program with Integrated Circuit Emphasis \[91\]

SPICE offers the most detailed level of simulation normally used for design. It models electrical circuits consisting of transistors, capacitors, resistors, etc., to determine the static and transient behavior of node voltages and branch currents within the network. The primary input to this circuit simulator is a list of labeled circuit elements and their connections, including definable waveform generators at input nodes. There is also a secondary input of process parameters such as threshold voltage, oxide thickness, transconductance parameter, etc., for each type of element to be modeled. The secondary inputs are defined in the Model card. For a CMOS process, two sets of parameters would be required, one for the p-channel devices and one for n-channel devices.

While all of the secondary input parameters to SPICE are given by CMC (Canadian Microelectronics Corp.), the choice of the correct set of primary input parameters is very important to the accuracy of the simulation. This is a potentially confusing area because of the variety of ways in
which a circuit can be simulated. Misunderstandings can easily lead to large simulation errors, perhaps orders of magnitude. It is therefore essential that the parameters are defined correctly in SPICE simulation.

SPICE can be used in schematic level simulation or layout level simulation. Usually, schematic level simulation gives less accurate outputs because most of the primary input parameters are estimated (AD, AS, PD, PS etc.). A schematic level is recommended only to verify the circuit functions and approximate the delays of the circuit prior to the actual layout of the circuit. After the circuit has been laid-out (on DAISY), a layout level simulation is required to obtain more accurate results.

2.3.2 Circuit Layout and Its Associated Capacitances and Resistances

Before we can analyze the transient characteristics of MOS circuitry, we must know the actual physical dimensions of the transistors and their interconnections so that the capacitances which limit switching speed can be calculated.

Since most of the parasitic capacitances have been considered in the secondary input parameters, the total load capacitance required to be considered in primary input parameters are gate capacitance, diffusion capacitance (drain and source capacitance), and routing capacitance. The characteristics of an MOS capacitor are available in many books [94,95,32,6] and will not be discussed here.
The parameters SPICE uses to calculate the gate, source, and drain capacitances will be discussed first. Following this, routing capacitance and resistance will be estimated.

2.3.3 Drain, Source and Gate Capacitances Approximation

Figure 2.8 shows a layout of a MOS transistor. Circuit simulators such as SPICE usually require entry of dimensions so the program may calculate capacitances on circuit nodes. Figure 2.8 shows a transistor with a gate width of 9*0.6 μm (Note: The transistor will be scaled to 0.6 of the actual layout size). This is the distance between the wall of the field diffusion and is entered on SPICE device cards as channel width \( W \). The transistor in Figure 2.8 also has a gate length of 5*0.6 μm, and is entered on SPICE.

Figure 2.8: A MOS Transistor
device cards as channel length L. To complete the required inputs for SPICE, device cards must be prepared with data on source and drain areas and parameters. Refer to Figure 2.8:

Source, drain area = 13*9*0.6*0.6 sq. um

\[ \text{Source, drain area} = 13 \times 9 \times 0.6 \times 0.6 \text{ sq. um} \]

\[ = 42.12 \text{ sq. um.} \]

Source, drain parameters = (9+9+9+4+4+9)*0.6

\[ \text{Source, drain parameters} = (9 + 9 + 9 + 4 + 4 + 9) \times 0.6 \]

\[ = 26.4 \text{ um.} \]

With the above inputs, the SPICE device card can be entered as:

\[ \text{Mxxxxxxxx ND NG NS NB MNAME L W AD AS PD PS} \]

where

ND = Drain Node  \quad \text{NG = Gate Node}

NS = Source Node  \quad \text{NB = Substrate}

MNAME = Model Name  \quad \text{L = Length of Channel}

W = Width of Channel  \quad \text{AD = Drain Area}

AS = Source Area  \quad \text{PD = Drain Parameter}

PS = Source Parameter  \quad \text{Mxxxxxxxx = Device Number}
2.3.4 Routing Capacitances

Let $L$, $W$, $T$, and $H$ represent the wire length, wire width, thickness of the wire, and the oxide height respectively and are shown in Figure 2.9.

![Diagram of wire capacitance parameters](image)

Figure 2.9: Wire Capacitance: Parameters

The capacitance per unit length of a single, isolated line can be computed as follows [101]:

$$C = \varepsilon_0 \varepsilon_{ox} \left[ 1.15 \left( \frac{W}{H} \right) + 2.8 \left( \frac{T}{H} \right)^{0.222} \right]$$

2.1

The first term in above equation relates to the lower and upper surfaces of the conductor, and the second term represents the side-wall contribution (fringe effect). When a finite line is taken into consideration, the above equation has to be modified and the influence of the four corners of a finite wire has to be taken into account. It can be written
as [101]:

\[ C = \varepsilon_0 \varepsilon_{\text{ox}} \left[ 1.15 \left( \frac{LW}{H} \right) + 1.4 \left( \frac{ZW + 2L}{H} \right)^{0.222} + 4.12H \left( \frac{T}{H} \right)^{0.708} \right] \]

when \( L \gg W \),

\[ C = \varepsilon_0 \varepsilon_{\text{ox}} \left[ 1.15 \left( \frac{L}{H} \right) + 2.8L \left( \frac{T}{H} \right)^{0.222} + 4.12H \left( \frac{T}{H} \right)^{0.728} \right] \]

The capacitance of each conductor does not follow the above equation when parallel lines are considered because of the presence of interelectrode (coupling) capacitance, as shown in Figure 2.10.

Figure 2.10: Coupling Capacitance
The capacitance can be expressed as

\[ C = C_a + C_{ab} \]

where

\[ C_a \text{ (intrinsic)} \]

follows equations 1 and

\[ C_{ab} = \varepsilon_0 \varepsilon_{ox} \left[ 0.03 \begin{bmatrix} W \\ H \end{bmatrix} + 0.83 \begin{bmatrix} T \\ H \end{bmatrix} - 0.07 \begin{bmatrix} T \\ H \end{bmatrix} 0.222 \right] \begin{bmatrix} S \\ H \end{bmatrix} - 1.34 \]

\[ = 2.4 \]

\[ S \]

is the spacing between the two parallel conductors. The relative error of equation 4 is less than 10% with the constraints:

\[ \frac{W}{H} < 10, \frac{T}{H} < 10, \text{ and } \frac{S}{H} < 10 \]

It is noted that the above equations are valid if the medium above the conductor is air. More often, metal lines are covered by an overlay of different material. This would make the analysis of the capacitance much more complex. Since this thesis is mainly based on computer simulation, the theory of coupled microstrip structures [96,97] will not be covered here.

In this thesis, the parameters for routing capacitances between metal and poly layers and the substrate are approximated and given by CMC. Therefore, the routing capacitances can be approximated using \( C = A \cdot C_p \) where

\[ A = \text{Area of the parallel plate capacitance} \]

\[ C_p = \text{Parameters for metal-field, metal-poly, metal-diffusion, or poly-field as shown in Table 2.3} \]
Capacitance (pf/sq. um)

<table>
<thead>
<tr>
<th>Material</th>
<th>Capacitance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gate (Cox)</td>
<td>6.9 E -4</td>
</tr>
<tr>
<td>Metal-Field</td>
<td>2.7 E -5</td>
</tr>
<tr>
<td>Metal-Poly</td>
<td>5.0 E -5</td>
</tr>
<tr>
<td>Metal-Diffusion</td>
<td>5.0 E -5</td>
</tr>
<tr>
<td>Poly-Field</td>
<td>6.0 E -5</td>
</tr>
</tbody>
</table>

Table 2.3: Parameters for Routing Capacitances

2.3.5 Routing Resistance

Consider a uniform slab of conducting material, the resistance can be expressed as

\[ R = \frac{R_s}{w} \]

where

\[ R_s = \text{Sheet resistance, having units of ohm/square} \]

(Provided by CMC)

\[ l \text{ and } w = \text{length and width of the conductor} \]

Note that \( R_s \) is completely independent of the area of the square. For example, a square slab of material of side 1 um has the same resistance as a square slab of the same material of side 1 km.

2.3.6 Interconnect Discontinuities

Figure 2.11 shows the discontinuities in the layout of interconnections. These discontinuities will introduce further parasitic capacitances and inductances. However, these parasitic parameters can be ignored when the frequency of operation is below 1 GHz [98].
2.3.7 Diffusion Resistance

From CMC, the typical value of N+ Diffusion resistance is given as 22.3 ohms/sq. and the P+ Diffusion resistance is given as 71.6 ohms/sq. It is noted that the p-type diffusion has higher resistivity than the n-type diffusion and, if diffusion must be used as an interconnect ---- something which is never recommended, anyway ------ it is advisable to use n-type diffusion.
2.3.8 Contact Resistance

Contact resistance has commonly been neglected because of its very small influence in overall performance. The parameters provided by CMC for a 3 x 3 um Contact & P+ Diffusion is 103 ohms, and minimum 3 x 3 um Contact & N+ Diffusion is 47 ohms. In this thesis, it is assume that current flowing through the contact is distributed uniformly all over the contact area. In reality, current is crowded at the leading edge of the contact. This means that the potential drop across the interfacial layer is not constant [100].
2.3.9 Defining Subcircuits

Figure 2.12 shows an example of 2 transistors sharing a common n diffused region. In the case of two devices sharing a common node (e.g. drain-drain, drain-source or source-source), it does not matter how node area and parameter are allocated between the devices.

Figure 2.12: Example of 2 Transistors Sharing A Common Node

When the same type of device connections appear in many locations of a designed circuit, it is more convenient to define them as subcircuits. In SPICE, the group of element cards which immediately follow the .SUBCT card define the subcircuit. Figure 2.13 shows the equivalent circuit of Figure 2.12 in layout form. Table 2.3 shows the parameters provided by CMC to calculate the routing resistances and capacitances. The SPICE subcircuit can be defined as follow:
Figure 2.13: Equivalent Circuit of Figure 2.12

.SUBCKT PT2 1 2 3 5 7
CPT21 4 0 0.00108PF
CPT22 6 0 0.00108PF
RPT21 1 4 110
RPT22 1 4 110
MPT21 3 4 5 8 PMO AD=42.12P AS=42.12P PD=26.4U PS=26.4U
MPT22 5 6 7 8 PMO AS=42.12P PS=26.4U AD=12.96P PD=10.2U
.END PT2

From the above SPICE deck, the gate, source, and drain capacitances are calculated by SPICE. The routing resistance can be obtained as $22 \times 5 = 110$ ohm and routing capacitance is calculated as $50 \times 0.36 \times 6 \times 10^{-5} = 0.00108$ pf. Note that drain and source resistances are calculated by SPICE with NRD
NRS = 1.0. NRD and NRS designate the equivalent number of squares of the drain and source diffusions; these values multiply the sheet resistance RSH specified on the .MODEL card for an accurate representation of the parasitic series drain and source resistances of each transistor.

2.3.10 Power and Ground Routing

Line inductance may be critical for power and ground lines where large currents are involved. The presence of large inductance may create noise problem. A model for the Vdd and Vss lines is shown in Figure 2.14 [102].

Figure 2.14: Model For Vss and Vdd Lines Which Includes Bounding Wires
The inductance of a coupled microstrip structure can be expressed as follows:

\[
L_{\text{intrinsic}} = \frac{\mu_0}{2} \frac{\varepsilon_0}{\left(\frac{1}{C_{\text{od}}} + \frac{1}{C_{\text{ev}}}\right)} \text{H/cm}
\]

\[
L_{\text{coupling}} = \frac{\mu_0}{2} \frac{\varepsilon_0}{\left(\frac{1}{C_{\text{ev}}} - \frac{1}{C_{\text{od}}}\right)} \text{H/cm}
\]

where \( \mu_0 \) is the permeability and \( C_{\text{ev}} \) and \( C_{\text{od}} \) are the capacitances for the even and odd modes. Typical values for conductor inductance are in the mH/cm range.

2.3.11 Modeling Long Interconnects

Many factors will influence the propagation delay of a signal, namely, the impedance of the driving source, the load impedance and the distributed resistance and capacitance of the wire. A single capacitor will suffice for metal interconnections because of its very low resistance. When the material with higher resistance such as polysilicon, is considered, a simple resistor-capacitor model will introduce significant inaccuracy, and cause as much as 30% error or more [103]. In this case, a model with distributed parameters has to be used. Figure 2.15 shows the \( \pi \)-ladder circuit for a long wire. It is reported that a 3-element \( \pi \)-ladder circuit provides an error of only 3%, and increasing number of stages does not significantly improve the accuracy of the model [104].
It is also shown by Sakurai [104] that for very large values of C and R, the delay is mainly dictated by the RC constant of the interconnect; wider driving gates cannot decrease this delay. Therefore, the use of polysilicon and diffusion as long interconnects are never recommended. If polysilicon has to be used as long interconnect, one possible strategy to optimize the speed is to segment the line into several sections and insert buffers with these sections as shown in Figure 2.16.

![Figure 2.15: Ladder Equivalent Circuit For Long Wire](image)

Figure 2.15: Ladder Equivalent Circuit For Long Wire
Figure 2.16: Segmentation of Long Polysilicon Line

2.3.12 The Concept of Equivalent Load Capacitance

It is convenient to employ a standard unit of capacitance which can be given a value appropriate to the technology but which can be used in calculations without associating it with an absolute value. The unit is denoted C[] and is defined as 'gate to channel capacitance' or 'equivalent gate load' of a minimum size (3um x 3um) transistor.

C[] may be calculated as follows:
- Gate area = 3 um x 3 um = 9 sq. um
- Capacitance value (From CMC) = 6.9 E -4 pf/um^2
- Standard value C[] = (9 x 6.9E-4) pf
  = 6.21 E -3 pf.

Figure 2.17 shows the layout of two inverters cascaded together with a smaller inverter connected to a larger inverter. Now, we would like to determine the overall load
that the first inverter sees. Basically, this load can be divided into three main capacitances [103]:

(1). Gate capacitance of second inverter.

(2). Stray capacitance of the interconnect between the output of the first inverter and the input to the second one.

(3). Intrinsic output capacitance of the first inverter, which is basically the sum of the drain-substrate junction capacitances of the two devices.

Figure 2.17: Layout of Two Cascaded Transistors

Note that the poly-metal overlap capacitance of the contact is ignored here.
Step 1:

From Table 2.3; Metal-Field Capacitance is given as

\[ 2.7 \times 10^{-5} \text{ pf/\mu m}^2 \]

due to the metal-poly contact, we have two branches of polysilicon interconnect with a total length of 81 \mu m.

Therefore, the capacitance can be calculated as:

\[
C_{\text{poly}} = 0.6 \times (2.7 \times 10^{-5}) \times 270 \times 3 \text{ pf} = 0.02187 \text{ pf}
\]

The total capacitance introduced by the interconnect is therefore:

\[
C = 0.02187 + 8.748 \times 10^{-3}
\]

\[
= 0.030618
\]

\[ \approx 4.9 \text{ C[\mu m]} \]

that is, the interconnect has the same gate capacitance of 4.9 minimum-size gates.

Step 2:

From Figure 2.17, it can be seen that the n-channel device of the second inverter has capacitance \((14 \times 0.6)/3 = 2.8 \text{ C[\mu m]}\), while the p-channel device has capacitance \((41 \times 0.6)/3 = 8.2 \text{ C[\mu m]}\). The total is therefore:

\[
C_{\text{inv}} = (8.2 + 2.8) \text{ C[\mu m]} = 11 \text{ C[\mu m]}
\]

Step 3:

In this thesis, the drain capacitances are calculated by SPICE. For manual calculation, the drain diffusion
capacitance, with the dimensions shown in Figure 2.17, may be approximated in the following manner [6]:

\[ C = \frac{C_{ja}}{d} + \frac{C_{jp}}{j} (2a + 2b) \]

where

\[ C_{ja} = \text{junction capacitance per sq. um.} \]

\[ C_{jp} = \text{periphery capacitance per um.} \]

a = width of diffusion region.

b = extent of diffusion region.

From CMC, \( C \) is given as

\[ C_{ja} = 4.4 \times 10^{-4} \text{ pf/um}^2 \text{ for n-device} \]

and

\[ C_{jp} = 1.5 \times 10^{-4} \text{ pf/um}^2 \text{ for p-device} \]

\( C \) is given as

\[ C_{ja} = 4.0 \times 10^{-4} \text{ pf/um}^2 \text{ for both p and n devices.} \]

Therefore,

\[ C_{n-device} = 4.4 \times 10^{-4} \times 9 \times 9 \times 0.6 \times 0.6 + 4 \times 10^{-4} (10.8 + 10.8) \]

\[ = 0.0214 \text{ pf} \]

\[ C_{p-device} = 1.5 \times 10^{-4} \times 9 \times 9 \times 0.6 \times 0.6 + 4 \times 10^{-4} (10.8 + 10.8) \]

\[ = 0.0130 \text{ pf} \]

\[ C_{out} = C_{n-device} + C_{p-device} \]

\[ = 0.0214 + 0.013 \]

\[ = 0.0344 \text{ pf} \]

\[ \approx 5.5 \text{ C}[] \]

The total load that the first inverter sees is therefore:

\[ C_{total} = C_{interconnect} + C_{inv} + C_{out} \]

\[ = 4.9 + 11 + 5.5 \]

\[ = 21.4 \text{ C}[] \]
From the above calculations, we can see that if we only consider the input capacitance of the second inverter, which is \(11 \, C[]\) as the equivalent gate load, this would create an error of about 49%.

2.3.13 Driving Large Capacitive Loads

The problem of driving comparatively large capacitive loads arises when signals must be propagated from the chip to off-chip destinations. One way to drive a large capacitive load is to use cascaded inverters each one of which is larger than the preceding stage by a width factor \(f\) shown in Figure 2.18.

![Cascaded Inverters Diagram](image)

**Figure 2.18: Cascaded Inverters**

Let \(N\) be the number of stages, \(CL\) be the off-chip load, and let

\[
Y = \frac{CL}{C[]} = f^N
\]

We now need to determine the value of \(f\) which will minimize
the overall delay for a given value of $Y$ and from the
definition of $Y$

$$\ln (Y) = N \ln(f)$$

That is

$$N = \frac{\ln (Y)}{\ln (f)}$$

With large $f$, $N$ decreases but the delay per stage
increases. Let the delay per stage be $fd$, it can be seen that
the delay is proportional to

$$Nfd = \frac{\ln (Y)}{\ln (f)}$$

The total delay is minimized if $f$ assumes the value $e$
(base of natural logarithms); that is, each stage should be
approximately 2.7 times wider than its predecessor [7].

2.4 Summary

The automatic gate matrix layout technique was
presented in this chapter. This layout technique not only
reduces the design time of a circuit but also the number of
errors introduced into a design. The SPICE simulation
technique was also discussed in this chapter. Following that,
the determination of routing capacitances and resistance were
discussed.
CHAPTER 3

AN ARCHITECTURE FOR A DIGITAL IMAGE THRESHOLD SELECTION ALGORITHM

3.1 Introduction

A common method of obtaining a threshold level for an image with a bi-modal gray level histogram is to select a gray level value that lies in the valley between the two peaks [3]. If the gray level histogram is uni-modal or multi-level [33,34], however, these histogram distributions provide an inadequate means for threshold selection. For images that do not have contrast differences between the object and the background, it becomes difficult to select an optimum threshold level. Thresholding at too high a level results in a loss of information, while thresholding at low levels can give rise to undesirable background clutter. Otsu and others proposed threshold selection methods using image statistics [35,36,37,39]. Rosenfeld, on the other hand, developed different techniques for threshold selection such as image modification [38,40], image smoothing and averaging [41,42], and histogram concavity analysis [43]. There are also various other techniques for image segmentation that have been proposed [44,45,46]. In this chapter, a new architecture suitable for single-chip VLSI implementation of automatic threshold selection is proposed. This architecture is software simulated and the results are discussed in this chapter.
3.2 The Hardware Structure

This algorithm assumes that an initial guess of the threshold, \( T \) is given. This threshold value is then improved on iteratively by first obtaining the mean of the distribution below and above the assumed threshold value, namely \( S_1 \) and \( S_2 \) as shown in Figure 3.1. The average value of \( S_1 \) and \( S_2 \) is then calculated. If the average value differs from the assumed threshold value \( T \), then this average value is chosen to be the new threshold value and the process is repeated until the two values converge.

![Histogram Distribution](image)

**Figure 3.1: Histogram Distribution**

M. A. Sid-Ahmed and N. Rajandran [47], Ridler and Calvard [48] have proposed hardware architectures for the implementation of the above algorithm. These structures work well but require extensive hardware. In this thesis, a
hardware structure that require less components is proposed. A VLSI implementation of the proposed architecture will also be investigated.

For a VLSI implementation, it is necessary to have an architecture that satisfies two criteria:

(i) Low hardware requirements.

   (Implemented such that the chip occupies less than 5000 × 5000 um square using 3um CMOS technology)

(ii) High speed.

The proposed architecture is shown in Figure 3.2. This architecture works as follows:

Initially, all the components are reset with the exception that an initial threshold value is read in using an 8-bit latch. The digital image is fed into the architecture pixel by pixel. If the gray level value is greater than the initial threshold value then the input pixel is fed to adder 1 and counter 1 increases its value by one. If the incoming signal has a lower gray level value than the initial threshold value, the input digital signal is fed to adder 2 and counter 2 increases its value by one. The process continues until the last pixel of the input image has been received. The outputs of the adders and counters are now used to compute the mean of the object and background gray levels. The new threshold value can be obtained by summing the two mean values and saving the 8 most significant bits of the result.
The new threshold value is then compared to the old threshold value. If the two values are not equal, the most recent threshold value is used as the initial guess and inputted to the 8-bit latch. The process is repeated on the input image until the two threshold values are the same.

Figure 3.2: The Block Diagram of The Proposed Architecture
3.3 Fast Algorithm for Digital Division

As shown in Figure 3.2, the proposed hardware requires two binary dividers. Each of these dividers will be used to obtain the value of $Si/Ci$ (i can either be 1 or 2), where $Si$ is a 24-bit integer and $Ci$ is a 16-bit integer. In this section, a fast algorithm for dividing based on logarithmic conversion is investigated.

A fast algorithm for digital logarithmic conversion was developed by G. A. Jullien which only requires simple computer manipulations [49, 50]. This algorithm will increase the computational speed at the expense of increased error. It is based upon a linear interpolation between values of $\log_2 R^n$ for successive integer values of $n$, (see Figure 3.3). This results in the approximation of the logarithmic function.

$$L_R(y) \approx \log_R 2^n + \left[ \frac{y}{2^n} - 1 \right] \log_R 2; \quad 2^n \leq y \leq 2^{n+1}$$

3.1

The fractional error can be expressed as

$$e_f = \frac{L_R(y)}{\log_R(y)} + |$$

$$= -\left( \frac{\log_2 2^n}{\log_2 y} + \left[ \frac{y}{2^n} - 1 \right] \frac{\log_2 2}{\log_2 y} \right) + 1$$

3.2

It can be seen from Equation (3.2) that $e_f$ is independent of $R$, and hence $R$ can be selected for ease of implementation.
For \( R = 2; \)
\[
L (Y) = (n-1) + Y \left( \frac{2^{-n}}{2^2} \right) = z \tag{3.3}
\]
The approximate anti-log of \( z \) is then
\[
Y = L^{-1} \left( z \right) = (z + 1 - n) 2 \tag{3.4}
\]
Therefore, the procedure for obtaining \( L_Y(Y) \), where \( Y \)
is represented in binary, is as follows:

1. Obtain the number of bits, \( n \), left to the binary
   point, and ending with a 1.
2. Subtract 1 from \( n \).
3. Move the binary point \((n-1)\) positions to the left.
4. The approximate \( L_Y(Y) \) is given by
   \[
   (n - 1) \uparrow \left( \text{Bits in } Y \text{ excluding the MSB} \right)
   \]
   Binary
   point

The procedure for obtaining \( L_Y^{-1}(z) \) is as follows:

1. Obtain \( n \); the truncated value of \( z \).

---

**Figure 3.3: Linear Interpolation i.e.**

\[
\begin{align*}
A & \left\{ \begin{array}{c}
2^n \\
2^{n+1}
\end{array} \right\} \\
B & \left\{ \begin{array}{c}
Y \\
\log_2 2^{n+1}
\end{array} \right\} \\
C & \left\{ \begin{array}{c}
L_Y(Y) \\
\log_2 2^n
\end{array} \right\} \\
D & \left\{ \begin{array}{c}
0 \\
E \\
F
\end{array} \right\} \\
E & \left\{ \begin{array}{c}
L_Y^{-1}(z) \\
2^n
\end{array} \right\} \\
F & \left\{ \begin{array}{c}
2^{n+1}
\end{array} \right\}
\end{align*}
\]
2. \( L_2^{-1}(z) = 1 \underbrace{xxx\ldots x}_{2^{-1}} \underbrace{\ldots \ldots \ldots \ldots \ldots x}_{\text{remaining bits}} \) 

The first \( n \) bits after the binary point in (z).

Since a division of two real numbers can also be carried out in logarithmic form, i.e., \( Y/X = \text{Anti-log}_R \left( \text{Log}_R Y - \text{Log}_R X \right) \). Let

\[
L_R(y) = \log_R 2^n + \left\lfloor \frac{y}{2^n} - 1 \right\rfloor \log_R 2 \quad 2^n \leq y \leq 2^{n+1}
\]

and

\[
L_R(x) = \log_R 2^m + \left\lfloor \frac{x}{2^m} - 1 \right\rfloor \log_R 2 \quad 2^m \leq x \leq 2^{m+1}
\]

where

\( L_R(y) \) = Approximate logarithmic function of \( Y \)

\( L_R(x) \) = Approximate logarithmic function of \( X \)

The error between \( L_R(Y) - L_R(X) \) and \( \text{Log}_R(Y) - \text{Log}_R(X) \) is given by

\[
\log_R y - \log_R x + \epsilon
\]

\[
= \log_R 2^n - \log_R 2^m + \left\lfloor \frac{y}{2^n} - 1 \right\rfloor \log_R 2 - \left\lfloor \frac{x}{2^m} - 1 \right\rfloor \log_R 2
\]

and the fractional error is given as

\[
\epsilon_{f1} = 1 - \frac{L_R(y) - L_R(x)}{\log_R y - \log_R x}
\]
therefore

\[
\varepsilon f_1 = \frac{\left( m - n + \frac{x}{2^m} - \frac{y}{2^n} \right) \log_R 2}{\log_R y - \log_R x} + 1
\]  \hspace{1cm} 3.9

It can be seen from Equation (3.9) that the fractional error is also independent of the logarithmic base \( R \).

Setting \( R = 2 \), we get

\[
m - n + \frac{x}{2^m} = \frac{y}{2^n}
\]

\[
\varepsilon f_1 = \frac{\log_2 y - \log_2 x}{\log_2 y - \log_2 x} + 1
\]  \hspace{1cm} 3.10

Now, if we let

\[ Ar = \frac{Y}{X} \]  \hspace{1cm} 3.11

and

\[ Al = \text{Anti-log} \left( L_R (Y) - L_R (X) \right) \]  \hspace{1cm} 3.12

Equation (3.9) can be rewritten as:

\[
\varepsilon f_1 = \frac{\log_R Ar - \log_R Al}{\log_R Ar}
\]  \hspace{1cm} 3.13

Therefore

\[ \frac{Ar}{Al} = \frac{\varepsilon f_1}{Ar} \]  \hspace{1cm} 3.14

and Equation (3.14) can be simplified as:

\[ Al = Ar (1 - \varepsilon f_1) \]  \hspace{1cm} 3.15

If we assign a new function \( Er \) as
Er = Ar / Al                                           3.16

substitute Equation (3.15) into (3.16), we obtain

\[ Er = \frac{\epsilon f_1}{A_r} \] 3.17

In Equation (3.4)

\[ \left( L_2 \right)^{-1} \left[ L_2 (Y) - L_2 (X) \right] \]

\[ = \left( L_2 (Y) - L_2 (X) + 1 - \text{Int} \left[ L_2 (Y) - L_2 (X) \right] \right) \]

\[ \times \left( L_2 (Y) - L_2 (X) \right)^{2} \]

Let

\[ L_2 (Y) - L_2 (X) = C \]

\[ \text{Int} \left[ L_2 (Y) - L_2 (X) \right] = K \]

Equation (3.18) can be written as

\[ \left( L_2 \right)^{-1} (C) = (C + 1 - K)^2 \] 3.19

and assign \( Er \) as

\[ Er = \frac{\left( L_2 \right)^{-1} (C)}{2^C} = \frac{2^C (C + 1 - K)}{2^C} \] 3.20

From Equations (3.17) and (3.20), it can be seen that the total error introduced by the fast log divider is

\[ \epsilon_{y/x} = \left| 1 - \frac{Er}{Err} \right| \] 3.21
The following examples further clarify the above derivations:

**Example 1: Division**

\[
\begin{align*}
\text{Divide} & \left( \frac{5 \times 770}{56} \right)_{10} \\
(5770)_{10} &= (1101001110010)_2 \\
(56)_{10} &= (111000)_2 \\
\text{According to the above procedure} \\
L_2(5770)_{10} &= (1100.101001110010)_2 \\
L_2(56)_{10} &= (101.11000)_2 \\
L_2(5770)_{10} - L_2(56)_{10} &= (110.111001110010)_2 = z
\end{align*}
\]

\[
L_2^{-1}(z) = \frac{5770}{56} \approx (1.111001.110010)_2 = 121.78125
\]

(Correct Ans. = 120.89286)

**Example 2: Error Analysis**

From Eq. (3.10)

\[\epsilon f_1 = \frac{5 - 12 + \frac{56}{32} - \frac{6770}{56}}{\log_2 6770 - \log_2 56} + 1\]

From Eq. (3.11)

\[Ar = \frac{6770}{56} = 120.8928571\]

From Eq. (3.15)

\[Al = 120.8928571 - 2.1327 \times 10^{-3}\]

\[= 119.662895\]
From Eq. (3.17)
\[
Er = (120.8928571) - 2.1327E-3
= 0.989826
\]

From Eq. (3.20)
\[
Err = \frac{6}{2} \frac{2 (6.90283 + 1 - 6)}{6.90283}
= 1.01770271
\]

and finally, from Eq. (3.21)
\[
\varepsilon_{y/x} = |1 - (0.989826)(1.01770271)|
= 7.348 \times 10^{-3}
\]

From Example 1, we found that
\[
L_2 \begin{bmatrix} 6770 \\ 56 \end{bmatrix}_{10} = 121.78125
\]
and the correct answer for 6770/56 is 120.89286, therefore,
\[
\varepsilon_{y/x} = \left| \frac{121.78125 - 120.89286}{120.89286} \right|
= 7.348 \times 10^{-3}
\]
which agrees with the value obtained by Eq. (3.21).
3.4 Hardware Architecture for the Fast Log. Divider

It is clear that a dedicated hardware implementation of an algorithm is faster than a software implementation. It is also true that different hardware approaches to a specific algorithm will provide different efficiency in silicon area, interconnection complexity and speed.

In next section, a new VLSI architecture for approximate logarithmic division will be introduced. This architecture is believed to operate at a higher speed than the conventional divider.
3.4.1. Architecture 1

Figure 3.4: Architecture 1: 4-Bit Divider

Figure 3.4 shows the proposed architecture for a 4-bit fast log divider. Both dividend and divisor are decoded using a special purpose decoder (decoder_1) shown in Figure
3.5, and whose truth table is given in Table 3.1. The decoded output determines the number of bits of dividend and divisor to be shifted left. A logic '1' on the no-shift bit means a shift is not performed, a logic '1' on the shift-1 bit means a one-bit shift to the left is performed ... etc. The decoded output of decoder_2, Figure 3.6 and Table 3.2, are used to

<table>
<thead>
<tr>
<th>A₃ A₂ A₁</th>
<th>NO-Shift</th>
<th>Shift-1</th>
<th>Shift-2</th>
<th>Shift-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0 0 1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0 1 0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 1 1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1 0 0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1 0 1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1 1 0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1 1 1</td>
<td>1</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Note: NO-Shift = A₃
Shift-1 = \overline{A₃} A₂ = A₃ + A₂
Shift-2 = A₁ \overline{A₂} A₃ = A₁ + A₂ + A₃
Shift-3 = \overline{A₁} A₂ \overline{A₃} = A₁ + A₂ + A₃
(Shift = Shift to the left)

Table 3.1: TT for Decoder_1
determine the most significant bits before the binary point of \( L(A) = L(A A A A) \) and \( L(B) = (B B B B B) \). The output from the two \( 3 \times 4 \) barrel shifters \((32,62)\) represents the fractional parts of \( L(A) \) and \( L(B) \). The decoded divisor is then subtracted from the decoded dividend using the 3-bit and 2-bit subtractors respectively. The output from the 2-bit subtractor, which represents the integer part of \( L(A/B) \), is fed to decoder-3 (Table 3.3 and Figure 3.7), and decoder-4 (Table 3.4 and Figure 3.8). The output from the 3-bit subtractor, which represents the fractional part of \( L(A/B) \), is fed to the \( 4 \times 4 \) barrel shifters as follows:

- \( i/p \) to barrel shifter 1 = S S S 0
- \( i/p \) to barrel shifter 2 = 1 S S S

where S S S is the 3-bit output of the subtractor.

The output from the two \( 4 \times 4 \) barrel shifters represents the integer and fractional part of \( A/B \).
Figure 3.5: Schematic of Decoder-1

Figure 3.6: Schematic of Decoder-2
<table>
<thead>
<tr>
<th>$A_3$</th>
<th>$A_2$</th>
<th>$A_1$</th>
<th>OUT 1</th>
<th>OUT 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**TABLE 2:** TT for decoder $\overline{12}$

Note: Out 1 = $A_3 + A_2$
Out 0 = $A_3 + \overline{A_2} \cdot A_1$
<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_0$</th>
<th>NO-Shift</th>
<th>Shift-1</th>
<th>Shift-2</th>
<th>Shift-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**TABLE 3:** TT for decoder-3

*Note: Shift = shift to the right.*

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_0$</th>
<th>NO-Shift</th>
<th>Shift-1</th>
<th>Shift-2</th>
<th>Shift-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**TABLE 4:** TT for decoder-4

*Note: Shift = shift to the left.*
Figure 3.7: Schematic of Decoder-3

Figure 3.8: Schematic of Decoder-4
The following example clarifies the above description.

Example: Architecture 1

Let: dividend = A₃ A₂ A₁ A₀ = 1111
  3 2 1 0
Divisor = B₃ B₂ B₁ B₀ = 0101
  3 2 1 0
Output of decoder 1 of dividend = 1000 i.e. no shift
i/p to 3 x 4 barrel shifter of dividend = 111
o/p of 3 x 4 barrel shifter of dividend = 111
o/p of decoder 2 of dividend = 11
  i.e. L₂ (1111) = 11.111²
Output of decoder 1 of divisor = 0100
  i.e. one bit shift to the left.
i/p to 3 x 4 barrel shifter of divisor = 101
o/p of 3 x 4 barrel shifter of divisor = 010
o/p of decoder 2 of divisor = 10
  i.e. L₂ (0100) = 10.010²
Therefore, L₂ (1111/0101) = 01.101²
i/p to decoder 3 and decoder 4 = 01
o/p of decoder 3 = 0010
  i.e. 2 bits shift to the right.
o/p of decoder 4 = 0100
  i.e. 1 bit shift to the left.
i/p to 4 x 4 barrel shifter (#2) = 1101
i/p to 4 x 4 barrel shifter (#1) = 1010
Therefore, output = 0011.0100
  i.e. (1111/0101)² = (11.01)²

It is noted that when the number of bits of the dividend and the divisor are increased, the hardware required
for the barrel shifters and the delay for the decoder also increase. In our image threshold selection architecture, two 24-bit by 16-bit divisions are required. In order to save silicon area, an alternate design is proposed next.

Figure 3.9: Architecture 2: Divider For A Large Number of Bits
3.4.2. Architecture 2

Figure 3.9 shows the proposed divider architecture for a large number of bits. The parallel loading shifters shown in Figure 3.9 are used to load in the incoming values or shift the values available in the latches on position to the left. When in the shifting mode, the shifted output controls the transmission gate. The down counter, counts down by 1 for every 0 shifted and stops when a 1 is detected. Counter 1 and counter 2 are programmed at an initial value of 1111 = \(16 - 1\). Figure 3.9 shows the numerical values at each stage using the example provided in previous section. The anti-log section consists of a 4-bit parallel loading up-counter [68] and a 16-bit parallel loading shifter. If we assume that the output from the 16-bit subtractor is 'A' and the output from the 4-bit subtractor is 'B', then the parallel loading shifter will shift 'A' \(16 - B\) bits to the right and 2 is added to the result. This can be implemented by a 4-bit parallel loading up-counter. Shifter #3 performs a one bit shift to the right for every count-up of the 4-bit up-counter and stops when counter overflow is detected. It is noted that since the LSB of 'A' is not contributing to the final answer, the LSB is discarded prior to feeding shifter #3.

3.4.3 Special Architecture: Architecture 3

It is interesting to note that the dividers required in the proposed threshold selection architecture have a special feature. That is, the outputs of the dividers are less than 256 in all cases. Because of this feature,
architecture 2 proposed in the previous section can be further simplified. Figure 3.10 shows the new architecture that is suitable for our threshold selection algorithm. It can be seen from Figure 3.10 that the shifter #3 and the 4-bit up-counter of Figure 3.9 have been replaced by an 8 x 8 barrel shifter and a 3 to 8 decoder. This architecture not only is simpler, but is also faster than architecture 2.

Figure 3.10: Architecture 3: Special Purpose Divider
3.5 Software Simulation For Fast Log Divider

A program is written in Fortran to simulate all the possible values of dividend and divisor for

\[ 0 < \text{dividend} < 2^{16} - 1 \]

and

\[ 0 < \text{divisor} < 2^8 - 1 \]

The worst case error percentage and absolute difference in pixel values are plotted against the gray level in Figure 3.11. It is found that the worst case error percentage in the range of \( 2^n \) to \( 2^{n-1} \), where \( n \) is an integer, drops as \( n \) increases. Appendix 1 shows the source codes of the written program.

3.6 Software Simulation of Threshold Selection Architecture

In order to verify the proposed architecture in Figure 3.1, a software program is written in Microsoft FORTRAN [51] and tested on various images. The images used are digitized at a resolution of 255 X 255 pixels using Video Van Gogh [52]. The source codes are listed in Appendix 2.
3.7 Examples

The effectiveness of the proposed design is demonstrated through 2 examples:

Example 1: Figure 3.12 shows the image of a bridge sampled at 128 x 128 pixels. Its histogram and thresholded image are shown in Figure 3.13 and 3.14 respectively. The number of iterations required is 4 and the arbitrary initial value is 128.

Figure 3.12: The Original Image of a Bridge

Example 2: This threshold is also applied to microscopic surface flaws detection [33]. Figure 3.15 shows the image (512 x 512 x 8-bits) of a surface obtained by a CCD camera through a microscope and an optical system (magnification = 500 times). The histogram of the image shown in Figure 3.16 indicates uni-modal distribution and the threshold image is
Figure 3.13: The Histogram of The Bridge Image

Figure 3.14: The Image After Thresholding
shown in Figure 3.17. The gray level above the threshold are assigned to dark (0), and the gray levels below the threshold value are assigned bright (255). It can be observed from Figure 3.17 that the background clutter is the roughness which has approximately the same gray level range as the surface flaw. However, to make flaw detection fully automatic, it is necessary to eliminate the background clutter. This can be done by repeating the above algorithm in the gray level range of 0 to the previous optimum threshold level, until a new optimum threshold level is found. In this case, the residual background clutter is removed after two iterations. Figures 3.18 and 3.19 show the thresholded image using the second and third optimum threshold level.

Figure 3.15 : An Image With Surface Flaws
Figure 3.16: Histogram Distribution of Figure 3.15.

Figure 3.17: Thresholded Image (1st Optimum Threshold Level)
Figure 3.18: Thresholded Image (2nd Optimum Threshold Level)

Figure 3.19: Thresholded Image (3rd Optimum Threshold Level)
3.8 Summary

A new architecture for digital image selection was presented in this chapter. The proposed architecture requires little hardware and it is suitable for VLSI implementation. To further increase the speed of the proposed architecture, few architectures for fast log divider were introduced in this chapter. The effectiveness of the proposed designs were demonstrated through various examples.
CHAPTER 4

SUBSYSTEM DESIGN AND CIRCUIT SIMULATION

4.1 Introduction

In the last two decades, integrated circuit technology has evolved from primitive circuits consisting of a few transistors interconnected to implement a basic gate, to the present levels of Very Large Scale Integration (VLSI) with integration of complete subsystems consisting of tens of thousands of gates. With the tremendous advances in integrated circuit technology, some designers have assessed system architecture as being non-critical. This is erroneous because an effective architecture development requires technological constraints as well as the internal device structure. The rapid advances in VLSI fabrication not only increase the number of gates per chip, but also increase the speed of each individual devices. Technology in chip fabricating, methods of chip partitioning, approaches in subsystem designing, and techniques of chip layout are just some of the many factors that will determine the performance of a VLSI system. The most important parameters that the designer has to consider are speed, area, noise margin, and power dissipation. In the previous chapter, we have proposed an area efficient image threshold selection architecture, developed a fast divider architecture, and partitioned the large system into some smaller subsystems. In this chapter, the design of each subsystem will be investigated.

Figure 4.1 shows a more detailed schematic of Figure
3.2. It is noted that the 24-bit adders in Figure 3.2 have

Figure 4.1: Block Diagram of The Threshold Selection Architecture
been replaced by the 16-bit counters and 8-bit adders to reduce the silicon area. The lower left corner of Figure 4.1 shows the blocks of the external image buffers and the address counter of the image. In this figure the whole architecture has been partitioned into subsystems (i.e.: counters, adders, comparators, dividers, etc.). In Chapter 3, it was explained that the divider itself is large enough to be partitioned into smaller second level subsystems. The block diagram of the divider is available in Figure 3.10 of Chapter 3.

4.2 Combinational Adder

The adder is one of the most important elements in our proposed architectures. In this section, the wide range of circuit options that are available when dealing at the transistor level of an adder will be examined.

If \( A \) and \( B \) are the adder inputs, \( C \) is the carry input, \( \text{SUM} \) is the sum output, and \( \text{Carry} \) is the carry output, the sum and carry can be defined as:

\[
\text{SUM} = ABC + ABC + \overline{ABC} + \overline{ABC}
\]

\[
\text{CARRY} = AB + AC + BC
\]

\[
= AB + C(A + B)
\]

The simplest approach to designing an adder is to implement gates to yield the required majority logic functions. Figure 4.2 shows the gate schematic of the above functions [6,53].

When two multi-bit binary numbers are to be added, the two methods that are available are bit serial and bit parallel.
Figure 4.2: Logic Diagram of a Full Adder

4.2.1 Serial Adder

An adder is serial when a single full-adder is successively presented with pairs of corresponding input bits in the least significant and working upwards in significance, storing each individual sum in a register.

Figure 4.3 shows a block diagram of a serial adder. Here, two shift registers contain the numbers to be added and a third shift register contains the sum. By shifting the sum into A while the bits of A are shifted out, it is possible to use one register for storing both the augend and the sum bits. A single full-adder performs the addition bit-by-bit and a carry latch retains the carry data for the subsequent addition.
The disadvantages of the serial adder are:

1. It is necessary for the registers to be of length \( M + 1 \) for addition of two binary words of length \( M \).

2. The speed of the adder is slow when the number of bits to be added are large.

### 4.2.2 Parallel Adder

An adder is parallel when a full-adder exists for each pair of corresponding bits in the numbers. Here all sums are computed simultaneously, the carry from the less significant additions being passed forward along the adder. Figure 4.4 shows the block diagram of a parallel adder. Parallel adders are usually faster than serial adders but require more logic. A parallel adder can either be a ripple-
carry adder or a carry look-ahead adder.

Figure 4.4: Parallel Adder

4.3 Carry-Ripple Adder

The time required to perform addition in the ripple-carry adder is the time required for the propagation of the carries through to the last stage. Although carries will not propagate through all stages in every addition, the time allocated for the additive operation must be at least equal to the longest carry propagate time, plus addition time in the last full-adder. Many structures of the adder are available in the literature, and they can be summarized as follows:
4.3.1 Static Adder [6,54]

Figure 4.5 shows the schematic of a static full-adder [6,54]. It is noted that the static logic requires equal numbers of p-transistors and n-transistors, therefore the number of devices involved is large. As far as the area is concerned, it is advisable to study alternate designs.

Figure 4.5: Fully Static Adder
4.3.2 Dynamic Adder \([15,55,53]\)

Because dynamic CMOS gates use only one of the two networks which form a static gate, a tremendous saving in the layout area is expected due to the simplicity of the interconnections for complex blocks of gates which have several inputs. Figure 4.6 shows CMOS dynamic logic implementations of an one-bit-adder \([15]\). Figure 4.7 shows a domino full-adder \([55]\). Figure 4.8 shows the schematic of a pseudo NMOS full-adder \([53]\).Figures 4.9 and 4.10 show the NORA adder \([14]\) and Zipper CMOS adder \([19]\) respectively.

**Figure 4.6: Dynamic Adder**
Figure 4.7: Domino Adder

Figure 4.8: Pseudo NMOS Full Adder
Figure 4.9: NORA Adder

4.10: Zipper CMOS Adder
4.3.3 Transmission Gate And Carry-Save Adders

[56, 59, 60]

The carry-save adder was first introduced in 1968 by Burton and Woaks [56] and was used in a multiplier design. Since then, much work have been carried out in this area [57, 58]. Today, most of the carry save adders are implemented using transmission gate logic [59, 60].

In transmission gate logic, the transistors are used to control the flow of a logic variable from input to output of the combination logic. Figure 4.11 shows a carry save adder implemented in transmission gate logic [60].

![Figure 4.11: Carry Save Adder](image)

The main disadvantage of a serial adder is its slow operating speed. This is due to the fact that the signal has to flow through various stages before the final output
can be obtained.

4.4 Carry Look-ahead Adders

For a carry look-ahead adder, we defined generated signal as $G_i$, propagated signal as $P_i$ and the sum as $S_i$, where

$$G_i = A_i \cdot B_i$$
$$P_i = A_i \oplus B_i$$
$$C_i = G_i + P_i C_i$$

and $$S_i = C_{i-1} \oplus P_i$$

By using carry look-ahead adders, the linear growth of adder carry delay with the size of the input word may be improved by calculating the carries to each stage in parallel.

Figure 4.12: Four-Bit Carry Look-Ahead Circuits By Hewlett Packard
Figure 4.13: Optimum Layout Via Functional Cell Realization

There are many approaches available in the literature for carry-look-ahead adders [61, 62, 63]. Figures 4.12 and 4.13 show the approach carried out by Hewlett-Packard Corp [64] and its optimum layout suggested by Uehera and VanCleemput [65], respectively.

In this thesis, adders based on the Manchester carry look-ahead approach are used. This approach increases the efficiency of the domino-carry chain [66]. The elemental circuit is shown in Figure 4.14.

Figure 4.14: Manchester Carry Chain.
The operation proceeds as follows: When CLOCK (CLK) is low, the output node is precharged by the p pull-out transistor. When CLOCK (CLK) goes high, the n pull-down transistor turns on. If carry generate \((A \cdot B)\) is true, then the output node discharges. If carry propagate \((A + B)\) is true, then a previous carry may be coupled to the output node, conditionally discharging it [6]. Figure 4.15 shows the schematic of a four-bit Manchester carry chain.

![Diagram of a four-bit Manchester carry chain](image)

Figure 4.15: Four-Bit Manchester Carry Chain

4.4.1 Speed Up The Worst-Case Delay

Even though the Manchester adder is claimed [62] to be faster than most of the other adders, its worst-case delay can be further improved. From Figure 4.15, we can see that when C is high, and all propagation signals are true, the output is pulled down through six series transistors. This
worst-case propagation time can be improved by bypassing the four stages if all carry propagate signal are true. The simplified diagram is shown in Figure 4.16. In the case of all carry propagates being true, the dynamic AND gate will turn on the carry bypass signal. Note that although the circuits have similar circuitry, the node capacitance at intermediate nodes in the look-ahead gate is approximately 1/2 that of the Manchester chain. Thus this arrangement should improve the overall speed of the adder.

Figure 4.16: Improved Manchester Look-Ahead Circuitry

It was discussed in section 4.4 that the technique that is useful for speeding up the carry propagation is carry look-ahead. This involves computing carries for groups of bit positions rather than just one at a
time. A CMOS implementation that utilizes the efficiency of the Manchester scheme to achieve group carry look-ahead is shown in Figure 4.17. The size of the grouping can be optimized for the particular application, but it has been reported that a group size of four is generally optimum [62], especially if carry buffers are used in the look-ahead generation path.

Figure 4.17: Schematic of Manchester Carry Look-Ahead Adder
4.4.2 Carry Select Adder [54,67]

It was mentioned in the last section that the optimum group size of a Manchester adder is four. A carry select style is introduced to an eight bit adder to further speed up the adder. Figure 4.18 shows a carry select adder. It can be seen that Adder-1, Adder-2, and Adder-3 are operated simultaneously with the exception that Adder-2 has a carry input of one and Adder-3 has a carry input of zero. The actual carry output of Adder-1 will select one of the two 'sum' outputs with a simple and fast multiplexer. Here we can actually obtain an eight-bit operation at approximately the speed of a four-bit adder.

![Carry Select Adder Diagram](image)

Figure 4.18: Carry Select Adder

4.4.3 Functional Simulation Of The Manchester Adder

Figure 4.19 shows the schematic and the simulated output of a four-bit carry generate and carry propagate.
Figure 4.19: Schematic and Simulated Results of the Carry-Propagate and Carry-Generate Stages
Figure 4.20: Schematic and Simulated Results of the SUM Stage
Figure 4.20 shows the schematic and the simulated waveform of a four-bit sum stage. The simulation of the whole adder is illustrated in Figure 4.21.

4.5 Subtractor

It was illustrated in Chapter 3 that the fast logarithmic divider requires two binary subtractors. To perform the subtraction of two numbers, the adder scheme previously discussed can be used. Here the subtraction takes the form:

\[ A - B = A + (-B) \]

and can be performed by adding the 2's complement of the subtrahend. The 2's complement can be formed by inverting the subtrahend (1's complement) and adding 1 to the least significant bit. The extra 1 can be added at the carry of the LSB of the adder. Figure 4.22 shows the block diagram of a 4-bit subtractor.

4.5.1 VLSI Layouts of Adders and Subtractors

Figure 4.23 shows the layout of an 1-bit carry generate and carry propagate stage of an carry look-ahead adder. The simulated results of this stage is shown in Figure 4.24 and Figure 4.25 respectively. It can be seen that the delay time of the carry generate stage is about 5 ns and the delay time of the carry propagate is about 4 ns.

Figure 4.26 shows the sum stage of the adder. Figure 4.27 shows the SPICE simulation of this stage and a delay time of 4 ns is observed. Figure 4.28 shows a 4-bit manchester carry chain stage and Figure 4.29 shows an 8-bit
Figure 4.21: Simulated Results of a 4-Bit Look-Ahead Adder
Manchester carry look-ahead carry select adder. This adder will operate at approximately the speed of a 4-bit adder. Figure 4.30 shows an 8-bit Manchester carry look-ahead adder. This adder occupied less area than the previous described adder, but operate at a slower speed.

Binary subtraction for numbers in 2's compliment format can be simply performed using the adder schemes previously discussed. The subtraction can be performed by taking the 1's complement (inverse) of the subtrahend, and adding 1 to the least significant bit, follow by addition.

Figure 4.22: Schematic of a 4-bit Subtractor
Figure 4.23: Layout of a Carry-Generate and Carry-Propagate Circuitry.

Figure 4.24: SPICE Simulation of the Carry-Generate Stage
Figure 4.25: SPICE Simulation of the Carry-Propagate Stage

Figure 4.26: SUM Stage of The Adder
Figure 4.27: SPICE Simulation of The SUM Stage

Figure 4.28: Layout of The 4-bit Manchester Carry Chain
Figure 4.29: Layout of the 8-bit Manchester Carry Look-Ahead Carry Select Adder
Figure 4.30: Layout of the 8-bit Carry Look-Ahead Adder

4.6 Fast Comparator

Designers who set up to create high-performance cost-effective VLSI chips must minimize the power, delay and area of CMOS ICs. But traditional logic design, with its black-box representation of Boolean function does not shrink the area.


Systematically designed pass-transistor networks can
reduce complex functions to highly regular structures that operate more quickly than conventional MOS logic, fill only one third as much space, and consume only one eighth as much power [24].

Figure 4.31 shows the basic structure of a CMOS comparator and its simulated outputs. More than one comparator can be cascaded to generate a multi-bit comparator. Figure 4.32 shows the schematic and the simulated waveforms of a 4-bit comparator. Figure 4.33 shows the VLSI layout of a 2-bit comparator. This layout can be cascaded to form a larger comparator.
ICAP -- U 381p1  T. A. N. 2215
Select ICAP command

History NCOMP1 currently ends at 20
colors saved.
colors saved.

Figure 4.31: CMOS Comparator
Figure 4.32: Schematic and Simulated Results of a 4-Bit Comparator
4.7 Multiplexer

Multiplexers are widely used to select between a number of inputs. Figure 4.34 shows a two way multiplexer with transmission gate. The multiplexer that uses transmission gates has substantially fewer transistors than comparable circuits with multi-input gates. The limitation of a transmission gate multiplexer is that it is not able to drive a large load. Figure 4.35 shows the VLSI layout of a multiplexer.

Figure 4.33: Layout of a 2-bit Comparator
Figure 4.34: Two Way Multiplexer

Figure 4.35: VLSI layout of a Multiplexer
4.8 Synchronous Counter

A counter is a arrangement of an arbitrary number of flip-flops used to generate an output sequence of binary numbers, directly related to the number of activating (clock) pulses observed at the clock input. Figure 4.36 shows the gate schematic of a 4-bit counter. It is noted that all the stages beyond the first stage are identical [68,69]. The first stage consists of 4 inverters, and 2 D-flip-flops. Stage 2 consists of 2 D-flip-flops, 1 multiplexer, 1 nand2 and an inverter. Figure 4.37 shows the design of a static D-flip-flop using static logic and Figure 4.38 shows an alternate design using transmission gate logic.

![Gate Schematic of a 4-bit Up Counter](image)

Figure 4.36: Gate Schematic of a 4-bit Up Counter
Figure 4.37: Static D_Flip-Flop

Figure 4.38: Transmission Gate Static Flip-Flop
The counter circuit is very simple and efficient. Its regularity is well suited to layout for VLSI fabrication. Figure 4.39 and 4.40 show the schematic of stage 1 and stage 2 respectively.

Stage 1 of the counter is laid-out on the silicon and is shown in Figure 4.41. The SPICE simulation of this stage predicts a delay time of 5 ns and is illustrated in Figure 4.42. Figure 4.43 shows the VLSI layout of the stage 2 of the counter. A delay of 7 ns is obtained from the simulation. Figure 4.44 shows the complete design of a 16-bit up counter. It is observed that the stages after the first are the same.

Apart from straight binary up counter, a down counter with parallel loading is required in the proposed fast logarithmic divider. Figure 4.45 shows the gate schematic of a down counter with parallel loading capability.

Figure 4.46 and Figure 4.47 show the stage 1 and the stage 2 of a down counter respectively. This counter has a parallel loading feature where a parallel input can be loaded into the counter. Figure 4.48 shows a 5-bit down counter programmed at the initial count stage of 10111. A 4-bit down counter programmed at 1111 is shown in Figure 4.49.
new block type requested please hit a window

Figure 4.39: Stage 1 of The Counter
Figure 4.40: Stage 2 of The Counter
Figure 4.41: Layout of Stage 1.

Figure 4.42 SPICE Simulation of Stage 1.
Figure 4.43: VLSI Layout of Stage 2.
Figure 4.44. Layout of the 16-Bit Counter
Figure 4.45: Gate Schematic of a 4-Bit Down-Counter
Figure 4.46: Stage 1 of The Down-Counter.

Figure 4.47: Stage 2 of The Down-Counter.
Figure 4.46: Layout of a 5-Bit Down-Counter
4-BIT DOWN COUNTER

Figure 4.49: Layout of a 4-Bit Down-Counter
4.9 Latch

A group of flip-flops sensitive to pulse duration is usually called a latch. Figure 4.50 shows the static master-slave latch \(^2\) \[6,70\] and Figure 4.51 shows the CMOS latch \[21,22\]. The former is used in the counter where static signal is required. The later has an advantage in area and is used in the accumulator. Figure 4.52 and 4.53 show the layouts of both latches.

![Static Master-Slave Latch Diagram](image)

**Figure 4.50: Static Master-Slave Latch**
Figure 4.51: C.MOS Latch

Figure 4.52: Layout of a Static Latch
2

Figure 4.53: Layout of a C 2 Latch
4.10 Barrel Shifter

The barrel shifter is an important element in the fast logarithmic divider structure. Figure 4.54 shows an 8-bit barrel shifter. It can be seen that the interbus switches have their gate inputs connected in a staircase fashion in groups of 8, and that there are now 8 shift control inputs which must be mutually exclusive in the active stage. It is clear that at a single instance only one transistor per row is being turned on. Therefore, an nMOS oriented design is acceptable. Figure 4.54 illustrates the operations of an 8-bit barrel shifter.

Figure 4.55 shows the layout of an 8-bit barrel shifter. The structure of the barrel shifter is highly regular. It can be seen from Figure 4.55 that a complete barrel shifter of any size can be formed by replication of the standard cell. It should be noted that standard cell boundaries must be carefully chosen to allow for butting together side by side and top to bottom to retain the overall topology.
Figure 4.54: Schematic and Simulation of a Barrel Shifter
Figure 4.55: Layout of an 8-bit Barrel Shifter.
4.11 Layouts of The Fast Log Divider and The Threshold Selection Hardware

Figure 4.56 shows the final layout of the proposed fast log divider architecture. The layout of the proposed threshold selection architecture is illustrated in Figure 4.57.

4.12 Summary

The designs of the subsystems are investigated in this chapter. The schematics of the main components are presented and simulated using LOGCAP. Layouts of the main components are also illustrated and simulated using SPICE. Finally, the layouts of the fast log architecture and the whole system are presented.
Figure 4.56: Layout of The Fast Log Divider
Figure 4.57: The Complete Layout of The System
CHAPTER 5
EXTENSIONS AND CONCLUSIONS

5.1 Introduction

Seldom, if ever, has there been such a dramatic change in any field of engineering as have taken place in the field of electrical engineering over the past 20 years. Just in short two decades, integrated circuits have evolved from SSI (Small Scale Integration) to MSI (Medium Scale Integration) to LSI (Large Scale Integration) and now, the advent of VLSI (Very Large Scale Integration) is becoming the technology of the 1980s.

When this project was first undertaken in September, 1985, the designs from the VLSI group at Windsor were fabricated using 5 μm technology. Since then, this technology has been improved twice. The 3 μm single metal technology was introduced in 1986 and now, it is updated to 3 μm double metal technology.

Like any field of engineering, the conclusion of this thesis does not mean the termination of this project. Instead, many new ideas and problems are encountered. It is hoped that this thesis will be of value to other students and will serve as a precedent for new architectures for automatic threshold selection and three new architectures for division. There are still many areas to be improved on and explored in the future.

In this chapter, the limitations and modifications of the design are discussed. Some possible improvements and
extensions as well as the conclusion of the design are also presented.

5.2 Limitation

All the chips fabricated through CMC are limited to an area of approximately 6700 um x 6700 um. This limitation has forced us to sacrifice speed for area in some circuits. An obvious example can be found in the architecture 2 of the divider presented in Chapter 2. In this architecture, the counters and parallel loading shifters are used to replace the much faster but larger barrel shifter (24 x 24) and decoders in the architecture 1 (Figure 3.4).

5.3 Possible Extensions and Improvements

While no single volume could adequately summarize the VLSI field, this section attempts to provide some improvements of the proposed architectures.

5.3.1 The CMOS Latches

Due to the fact that all the architectures presented were carried out in VLSI for the very first time, the master-slave flip-flops in counter designs were constructed by cascading two static flip-flops to obtain the maximum stability. The schematic and layout of the clocked CMOS register (C MOS) shown in Figures 5.1 and 5.2 occupy a smaller area than the static CMOS register that was used because of its regular structure. Note that C MOS latches may have a skew problem. With the careful design of C MOS logic, a smaller area in silicon should be obtained.
Figure 5.1: The Schematic of a CMOS

Figure 5.2: The VLSI Layout of a CMOS Latch
5.3.2 Double Metal CMOS

Recently, the double metal 3 um. technology has been introduced by CMC. Most of the layouts in this thesis were carried out in single metal technology and hence, redesigning the layout in double metal technology will reduce the area occupied.

5.3.3 Parallel Decoding Counter

Figure 4.36 in Chapter 4 shows a simple 4-bit synchronous counter with serial decoding implemented in CMOS. Faster operating speed can be obtained by the use of parallel (synchronous) decoding. An eight-stage counter will need a 7-input gate to decode the carry for its eight stage. Because of the specification degradations, gates having more than four inputs are implemented in two stages as shown in Figure 5.3. This implementation introduces other delays, which being the performance of a multistage synchronous counter with parallel decoding down to the level of that of a synchronous counter with ripple carry [69]. Therefore, the fastest circuit arrangement is by combination of parallel and ripple decoding. It was suggested [69] that the basic four-stage blocks with parallel decoding are used to build larger counters, and ripple carry is used between the blocks.
5.3.4 Circuit Modifications

In the future, when layout area is no more a limitation to the designers, the proposed architectures can be modified to operate at higher speed. Two suggestions are presented as follows:

1. We are interested in using pipelining as a general method for implementing these algorithms in hardware. By pipelining, processing may proceed concurrently with input and output, and consequently overall execution time is minimized.

2. One of the fastest-growing demands for semiconductor devices is memories. The basic ability of the RAM (Random-Access Memories) to store and retrieve data at will, makes RAM a popular system design tool. One advantage using RAM in the proposed threshold selection architecture is to utilize the accessibility of the RAM. RAM can be used to generate the histogram of the input image within the
architecture. By doing so the processing speed can be reduced dramatically.

5.3.5 Other Applications

The fast-log architecture presented in Chapter 3 can be extended to obtain the approximate square-root value of a real number. Figure 6.4 shows a circuit for obtaining the square-root of a real number, and is based on the approximate log algorithm. Since

$$
\log_2 \left( \frac{1}{2} \right) = \frac{1}{2} \log_2 (X) - \frac{1}{2} L_2 (X)
$$

the square-root can be performed by making a one-bit shift to the right on the approximate log, and then taking the approximate inverse log. For example, if $X = 1101$, then $L_2 (X) = 11.101$, $1/2 L_2 (X) = 1.1101$ and therefore $X = 11.101 = 3.625$.

To understand the operation of the circuit shown in Figure 6.1, an example is presented as follows:

A: A A A A = 1011
3 2 1 0

From Table 3. (Chapter 3): the output of Decoder-1 is 1000, i.e., no shift, and hence the output from $3 \times 4$ barrel shifter is 011.

From Table 3.2 (Chapter 3): the output of Decoder-2 is 11, therefore, no shift is performed by the $2 \times 2$ barrel shifter.

Hence, the input to the $2 \times 2$ barrel shifter is 11 and the output is therefore 11.
The input to the 4 x 2 barrel shifter is 1011.

According to Figure 3.4 in Chapter 3, the input is shifted to the left one bit to produce the output 0110.

Therefore, the square-root of 1011 (A A A A) is 11.0110.

It is obvious from the approach presented that this work can be readily extended to obtain the approximate value of the 2n square-root ($\sqrt[2^n]{x}$) of a real number, and for multiplication.

Figure 5.4: Architecture For Square-Root
5.4 Conclusion

A new architecture for the threshold selection algorithm has been developed. This hardware structure requires very few components and it is suitable for VLSI implementation. Subsystem partitioning of the architecture has made the interconnection simple and regular. Simple and regular interconnections lead to low cost implementations, high performance and low overhead for support components. The most important parameters that have been considered in the VLSI design are speed, area, noise margin, and power dissipation.

Basic knowledge in VLSI implementations has been presented in Chapter 1. Chapter 2 covered the automated gate matrix layout technique, which optimizes the silicon area. The circuit layout and its associated capacitances and resistances and the SPICE simulation of the presented layout were also presented in Chapter 2. Chapter 3 introduced a simple algorithm for threshold selection in hardware implementations and three architectures of division using fast approximate logarithmic conversion. These dividers are believed to operate at higher speed than the conventional divider. Although the results obtained for division are not very accurate, they are obtained with a minimum number of operations. These dividers are suitable for our image threshold selection algorithm because speed is an important factor in this architecture, and a low percentage error can be tolerated. Chapter 4 investigated the possibilities of
designing each subsystems. The schematics of the main components were presented and simulated using a functional simulator (LOGCAP). Chapter 4 also presented the layouts and the SPICE simulations of the designs. In this Chapter, some possible improvements, extensions and limitations of the design have been discussed.
BIBLIOGRAPHY

1. G.E. Moore, "Are We Really Ready For VLSI?", Caltech Conf. on VLSI, January 1979, pp 3-14.


44. P. Panda, A. Rosenfeld, "Image Segmentation by Pixel Classification in (Gray Level, Edge Value) Area", IEEE Trans. on Comp., Vol C-27, No. 9, September 1978, pp 875-879.


71. C. A. Mead, "VLSI And Technological Innovation", VLSI 81.


87. Omar Wing, "Interval-Graph-Based Circuit Layout", Proceedings ICCAD, 1983, pp. 84-85.


APPENDIX I: FORTRAN PROGRAM FOR AUTOMATIC THRESHOLD SELECTION ALGORITHM
program Threshold Selection Algorithm With Fast Log
INTEGER*4 S2,S1,C2,C1
INTEGER*2 ER,VALUE,MOD,BURST
CHARACTER*20 'FSP'.
VALUE=0
ncn=1
ER=0
FSP='B:bridg2.PIC'
CALL GETDAT(FSP,ER)
IF(ER.EQ.0)GOTO 100
WRITE(*,90)ER
90 FORMAT(I3)
STOP 'ERROR'
100 ITRES=128 =
MOD=0
BURST=9
IF(ER.NE.0)GOTO 80
5 S1=0
S2=0
C1=0
C2=0
DO 15 I=2,251
DO 15 J=16,255
CALL GETPNT(I,J,VALUE,ER)
IF(VALUE.GT.ITRES)THEN
S1=S1+VALUE
C1=C1+1
ELSE
S2=S2+VALUE
C2=C2+1
END IF
15 CONTINUE
EM1 = LDIV(S1,C1)
EM2 = LDIV(S2,C2)
ITEMP=(EM1+EM2)/2
IF(ITEMP.NE.ITRES)THEN
ITRES=ITEMP
ncn=ncp+1
GOTO 5
END IF
DO 25 I=2,251
DO 25 J=16,255
CALL GETPNT(I,J,VALUE,ER)
IF(ER.NE.0)GOTO 80
IF(VALUE.GT.ITEMP)THEN
VALUE=255
CALL SETPNT(I,J,VALUE,ER)
IF(ER.NE.0)GOTO 80
ELSE
VALUE=0
CALL SETPNT(I,J,VALUE,ER)
IF(ER.NE.0)GOTO 80
END IF
25 CONTINUE
write(*,111) itemp,ncn
111 format(i4,i4)
FSP='b:BB.BUF'
CALL SAVDAT(FSP,ER)
IF(ER.NE.0)GOTO 80
CALL VUBDAT(MOD,BURST,ER)
IF(ER.NE.0)GOTO 80
STOP
END

C -----------
C THIS FUNCTION CALCULATES THE
C FAST LOG DIVISION OF TWO GIVEN
C INTEGERS.
C -----------
FUNCTION L DIV(LIN,MIN)
INTEGER*4 LIN,MIN
DIMENSION N(50)
I = 0
LANS = 0
J = 0
L = LIN
M = MIN
5 IF(L.LE.1)GOTO 6
I = I + 1
L = L/2
GOTO 5
6 IF(M.LE.1)GOTO 7
J = J + 1
M = M/2
GOTO 6
7 NB = I - J
LA = LIN - 2 ** I
MA = (MIN - 2 ** J) * 2 ** NB
NA = LA - MA
IF(NA.LT.0)THEN
NB = NB - 1
NA = NA + 2 ** I
ENDIF
15 IF(NA.EQ.0)GOTO 15
NF = 0
FACTOR = (2**I - 1)/(NA)
IF(FACTOR.GT.2)NF = FACTOR/2
15 NA = NA+2**I
II = 0
8 IF(NA.LT.1)GOTO 10
II = II + 1
N(II) MOD(N)
NA = NA/2
GOTO 8
9 DO 9 K = II - 1
LANS = N(K)*2**I+NB-II+LANS
LDIV = LANS
RETURN
END
APPENDIX II: FORTRAN PROGRAM FOR FAST LOG ERROR ANALYSIS
DIMENSION N(50), LWRST(512), WORST(512), DUM(512), bb(512)
1rst = 1
DO 100 I = 1, 512
bb(I) = 0
SUM(I) = I
LWRST(I) = 0
100 WORST(I) = 0
DO 1 JI = 356, 356
MIN = JI
IJ = JI + 511
DO 1 KI = JI, IJ
abc = abc + 1
LIN = KI
J = 0
I = 0
LANS = 0
L = LIN
M = MIN,
WRITE(*,*) lin, min
5 IF(L.LT.1) GOTO 6
I = I + 1
L = L / 2
GOTO 5
6 IF(M.LE.1) GOTO 7
J = J + 1
M = M / 2
GOTO 6
7 NB = I - J
LA = LIN - 2 ** I
MA = (MIN - 2 ** J) * 2 ** NB
NA = LA - MA
IF(NA.LT.0) THEN
NS=NB-1
NA=NA+3**I
END IF
IF(NA.EQ.0)GOTO 15
NF=0
FACTOR=(2**I-1)/(NA)
IF(FACTOR.GT.2)NF=FACTOR/2
15 NA=NA+2**I
II=0
8 IF(NA.LT.1)GOTO 10
II=II+1
N(II)=MOD(NA,2)
NA=NA/2
GOTO 8
10 DO 9 K=II-NB,II
9 LANS=N(K)***(K+NB-II)*LA
LTRUE=LIN/MIN
LDIFF=ABS(LTRUE-LANS)
bb(1diff+1)=bb(1diff+1)+1
IF(LDIFF.GE.LRST)LRST=1diff+1
WRITE(*,*)LIN,MIN,LTRUE,LANS,1diff
IF(LDIFF.GT.LWORST(LTRUE)LWORST(LTRUE)=LDIFF
PERCENT=ABS((LANS-LTRUE)/FLOAT(LTRUE)*100)
IF(PERCENT.GT.WORST(LTRUE))WORST(LTRUE)=PERCENT
IF(LDIFF.GT.10.AND.PERCENT.GT.13.5)WRITE(*,*)
LINE,LMIN,LTRUE,LANS,LDIFF,LWORST
C IF(PERCENT.GT.30)WRITE(*,*)LIN,MIN,LTRUE,LANS,PERCENT
C CONTINUE
C WRITE(*,*)WORST,LWORST
OPEN(UNIT=7,FILE='LOG',STATUS='OLD')
WRITE(7,11)
11 FORMAT('FAST LOG ERRORS ANALYSIS')
WRITE(7,*)
WRITE(7,*)
WRITE(7,12)255
12 FORMAT(5X,13)
WRITE(7,13)
13 FORMAT(6X,'3')
WRITE(7,14)
14 FORMAT(2X,'GRAY LEV',17X,'ABS_DIFF',4X,'PERCENT')
DO 101 I=256,511
WRITE(7,17)FLOAT(I),DUM(I),FLOAT(LWORST(I)),WORST(I)
17 FORMAT(E10.3,3X,E11.3,1X,E11.3,1X,E11.3)
101 CONTINUE
WRITE(7,18)
18 FORMAT('END')
CLOSE(UNIT=7)
DO 34 I=1,LRST
ABCD=ABCD+bb(i)
34 WRITE(*,*)I-1,bb(i)
WRITE(*,*)abcd
STOP
END