HIGH TEMPERATURE
PROCESSOR AND MEMORY
IMPLEMENTATION

by
Burak Erbağcı

MASTER THESIS PROJECT

submitted to
School of Engineering
Ecole Polytechnique Fédérale de Lausanne (EPFL)

in fulfillment of the thesis requirements
for the degree of

MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

supervised by
Prof. Yusuf Leblebici
Alessandro Cevrero
Radisav Cojbasic
Microelectronic Systems Laboratory - STI/LSM, EPFL

April 26, 2011
Abstract

Leakage currents, especially *subthreshold* leakage, represent the limiting factor on the high temperature functionality of MOS transistors, as well as digital circuits comprising these elements. A severe degradation of performance is manifested for digital circuits through the decrease of device on-current and the elevation of both device leakage currents and the associated diffusion capacitances. Apart from the performance loss, reliability of such blocks also becomes an issue due to the degradation of digital logic levels and noise margins. Among other digital blocks, memory elements emerge as the most critical components in high temperature operation because of some specific design and implementation issues. Hence, performance of digital circuits employing memory blocks are dictated by such elements.

This thesis attempts to address high temperature issues in transistor and circuit levels, specifically in memory elements; and to design and implement reliable memories, which are required to be functional under severe thermal environment conditions (> 200°C), for electronic units in next generation all-electric aircrafts as part of *CREAM* project. *SOI technology* is targeted in this project due to significantly reduced leakage currents and improved device performance in comparison with bulk silicon technologies. We have recently taped-out a test chip including different memory implementations, such as a *technology compiled macro* and *standard cell based* memories, as well as various test structures to characterize the high temperature performance of target technology. Test results of this chip will be exploited to design and implement a robust high temperature memory.
Acknowledgements

First, I would like to express my deepest gratitude to my supervisor, Prof. Yusuf Leblebici, for his guidance and support from the first moment I took a step to EPFL. Without his support, knowledge and enthusiasm, none of these works would have been possible and I have a lot of pleasure to work with him.

I would like to thank Alessandro Cevrero and Radisav Cojbasic for their supervision, useful suggestions and friendly advices throughout this research. Their optimism and sympathy always encouraged me, through thick and thin. We had intense discussions and they were always ready to explain and discuss various concepts. Same way, I would like to thank Alain Vachoux for his everyday support, his help with the tools. I would like to thank Armin Tajalli for his useful comments and advices through the last stages of my work. Finally, I would like to thank all LSM family for the enjoyable laboratory environment.

I would like to specially thank Tugce Yazicigil. She is always there for me, like a beloved sister. She is the one who has been motivating me with all her patience and endless support from the first moment we came here. With all my sincerity, I cannot imagine a life in Lausanne without her.

I would like to express my deepest gratitude to Emrah Tas. From the first moment we met, he is like a brother to me. His endless support, patience and sincerity made these stressful and rough days bearable. He, one of two-civilized people, was always there for me for encouragement.

The word thanks cannot do enough justice to express my gratitude to my best friends Mert Bariskin, T. Zafer Ozcan, H. Can Yener and Cagla Cakir. They have been taking care of me even though there are thousands of kilometers between us.

I would like to thank my two brothers, A. Galip Bayrak, and Eymen Kurdoğlu for their support and enthusiasm. My special thanks of course to Gozen Koklu. She is the one who always has the answers. Her support as an elder sister encouraged me
during the thesis. I would like to thank my three kanka, Anil Yuce, Pinar Kaynak and Cigdem Cakmakli for their enthusiasm, joy and support.

I owe my deepest gratitude to my beloved friends, here in Lausanne and back in Turkey, who have constantly filled my life with joy and given me reasons to go on in times of despair. I would certainly have a tremendously hard time finishing my studies at EPFL, if I did not have you by my side. Many thanks to everyone for everything we shared, but especially to Nilay Dagtekin, Kerem Seyid, Eren Halici, Onur Kazanc, Onur Ozen, Zeynep Temel, Gozde Tuncer, Gulperi Ozsema and finally, Arda Gunay.

Last and foremost, I would like to express my deepest gratitude and my endless love to my family. I would never reach to this point without their love, endless support and interest in my career. My work is dedicated to my family.
# Contents

1 Introduction ......................................................... 1
   1.1 Aims and Motivation ........................................... 1
   1.2 Thesis Overview .............................................. 3

2 Background ......................................................... 5
   2.1 The MOSFET Transistor ......................................... 5
      2.1.1 The Device Physics and Operation ....................... 5
      2.1.2 Transistor Level High Temperature Effects ............ 17
   2.2 Overview of SOI CMOS Technology ............................ 21
      2.2.1 Why SOI? .............................................. 22
      2.2.2 SOI Transistors ....................................... 23
      2.2.3 SOI Drawbacks ....................................... 25
      2.2.4 SOI High Temperature Considerations .................. 27
   2.3 Target Technology ............................................ 29
   2.4 Summary and Discussion ...................................... 31
## 3 High Temperature Effects in Circuit Level

3.1 Degradation of Logic Levels, Noise Margin and Performance ........................................... 33

3.2 Failure of Memory Cells ........................................................................................................... 52

3.3 Summary and Discussion ........................................................................................................... 61

## 4 System Overview

4.1 Overview .................................................................................................................................. 62

4.1.1 General Description .................................................................................................................. 62

4.1.2 Features .................................................................................................................................... 62

4.2 Application Specifications .......................................................................................................... 65

4.3 Core Unit .................................................................................................................................... 65

4.3.1 CPU Description ....................................................................................................................... 65

4.3.2 System Bus Architecture ......................................................................................................... 71

4.3.3 Peripherals Description ............................................................................................................ 74

4.4 Power-On-Reset ......................................................................................................................... 79

4.4.1 Block Description ..................................................................................................................... 79

4.4.2 I/O Description ......................................................................................................................... 79

4.5 Reset Manager ............................................................................................................................ 80

4.5.1 Block Description ..................................................................................................................... 80

4.6 Boot Loader ................................................................................................................................ 81

4.6.1 Block Description ..................................................................................................................... 81

4.6.2 Register Definition .................................................................................................................... 82
CONTENTS

4.6.3 I/O Description ................................................. 83

4.7 Clock Manager .................................................. 83
  4.7.1 Block Description ........................................... 84
  4.7.2 Register Definition .......................................... 85
  4.7.3 I/O Description .............................................. 86

4.8 Interrupt Controller ............................................ 87
  4.8.1 Block Description ........................................... 87
  4.8.2 Register Definition .......................................... 89

4.9 Watchdog Timer .................................................. 90
  4.9.1 Block Description ........................................... 91
  4.9.2 Register Definition .......................................... 94
  4.9.3 I/O Description .............................................. 95

4.10 General Purpose Input/Output Port ............................ 95
  4.10.1 Block Description ........................................... 96
  4.10.2 I/O Description .............................................. 99

4.11 Serial Communication Interface ............................... 99
  4.11.1 Block Description ........................................... 100
  4.11.2 Register Definition .......................................... 101
  4.11.3 I/O Description .............................................. 102

4.12 Serial Peripheral Interface Master ........................... 102
  4.12.1 Block Description ........................................... 103
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.12.2</td>
<td>Register Definition</td>
<td>104</td>
</tr>
<tr>
<td>4.12.3</td>
<td>I/O Definition</td>
<td>106</td>
</tr>
<tr>
<td>4.13</td>
<td>Package</td>
<td>106</td>
</tr>
<tr>
<td>4.14</td>
<td>Circuit Area and Floorplan</td>
<td>106</td>
</tr>
</tbody>
</table>

### 5 Memory Blocks

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>Technology compiled IBM MM</td>
<td>107</td>
</tr>
<tr>
<td>5.1.1</td>
<td>SRAM Memory Cell and Array</td>
<td>108</td>
</tr>
<tr>
<td>5.1.2</td>
<td>Array Functional Operation</td>
<td>110</td>
</tr>
<tr>
<td>5.2</td>
<td>Standard Cell Based Memories</td>
<td>113</td>
</tr>
<tr>
<td>5.2.1</td>
<td>FF based SCM</td>
<td>117</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Latch based SCM</td>
<td>122</td>
</tr>
<tr>
<td>5.3</td>
<td>Summary and Discussion</td>
<td>126</td>
</tr>
</tbody>
</table>

### 6 Test Structures

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Memory Testing</td>
<td>128</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Memory Fault Models</td>
<td>129</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Memory Test Algorithms</td>
<td>131</td>
</tr>
<tr>
<td>6.1.3</td>
<td>Testing Memory Blocks</td>
<td>132</td>
</tr>
<tr>
<td>6.2</td>
<td>The Other Test Structures</td>
<td>142</td>
</tr>
<tr>
<td>6.3</td>
<td>Summary and Discussion</td>
<td>143</td>
</tr>
</tbody>
</table>

### 7 The Test Chip

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>The Test Chip</td>
<td>144</td>
</tr>
</tbody>
</table>
List of Figures

2.1 The MOS transistor .................................................. 6

2.2 NMOS transistor for positive $V_{GS}$, showing depletion region and induced channel ............................................. 7

2.3 NMOS transistor with bias voltages ................................. 9

2.4 NMOS transistor under pinch-off condition ......................... 11

2.5 Leakage current components of MOS transistors ............... 13

2.6 $I_G$ versus $V_{GS}$ for various oxide thicknesses .................. 14

2.7 Relationship between $V_T$ and the subthreshold current ........ 15

2.8 (a) MOS device capacitances, (b) decomposition of S/D junction capacitance into bottom-plate and sidewall components .......... 16

2.9 Temperature dependence of subthreshold current ............... 19

2.10 Impact of technology scaling on subthreshold current as a function of temperature ................................................. 19

2.11 (a) Cross-section of an SOI NMOS transistor, (b) An SOI transistor with Shallow Trench Isolation ................................. 21

2.12 (a) Fully-Depleted, (b) Partially-Depleted SOI NMOS transistor ... 23

2.13 Illustration of Kink effect on a PD-SOI device .................. 25
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.14</td>
<td>Illustration of Parasitic Bipolar transistor</td>
<td>26</td>
</tr>
<tr>
<td>2.15</td>
<td>A cross section in IBM 180 nm SOI process with 4AM option</td>
<td>30</td>
</tr>
<tr>
<td>2.16</td>
<td>Layout of (a) dgnfet_bc, (b) dgnfet_lc, and (c) dgnfet transistors</td>
<td>31</td>
</tr>
<tr>
<td>2.17</td>
<td>Antenna diode in SOI technology</td>
<td>31</td>
</tr>
<tr>
<td>3.1</td>
<td>Definition of noise margins $NM_L$ and $NM_H$</td>
<td>34</td>
</tr>
<tr>
<td>3.2</td>
<td>Butterfly curve used for estimating NM</td>
<td>35</td>
</tr>
<tr>
<td>3.3</td>
<td>An inverter gate (a) schematic (b) VTC</td>
<td>35</td>
</tr>
<tr>
<td>3.4</td>
<td>Chain of inverters</td>
<td>36</td>
</tr>
<tr>
<td>3.5</td>
<td>Simulation results of an inverter (corner = tt) (a) VTC, (b) the corresponding NMs</td>
<td>38</td>
</tr>
<tr>
<td>3.6</td>
<td>Ratio of device on-currents (IN/IP) with respect to temperature (corner = tt)</td>
<td>39</td>
</tr>
<tr>
<td>3.7</td>
<td>Simulation results of an inverter (corner = ss) (a) VTC, (b) the corresponding NMs</td>
<td>41</td>
</tr>
<tr>
<td>3.8</td>
<td>Simulation results of an inverter (corner = sf) (a) VTC, (b) the corresponding NMs</td>
<td>42</td>
</tr>
<tr>
<td>3.9</td>
<td>Simulation results of an inverter (corner = ff) (a) VTC, (b) the corresponding NMs</td>
<td>43</td>
</tr>
<tr>
<td>3.10</td>
<td>Simulation results of an inverter (corner = fs) (a) VTC, (b) the corresponding NMs</td>
<td>44</td>
</tr>
<tr>
<td>3.11</td>
<td>Intrinsic inverter delay with respect to temperature and process variation</td>
<td>45</td>
</tr>
<tr>
<td>3.12</td>
<td>Schematic of a two-input NAND gate</td>
<td>46</td>
</tr>
<tr>
<td>3.13</td>
<td>Illustration of stack effect</td>
<td>46</td>
</tr>
<tr>
<td>3.14</td>
<td>Schematic of a three-input NAND gate</td>
<td>47</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td></td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td></td>
</tr>
<tr>
<td>3.15</td>
<td>Simulation results of a NAND2 gate (A: High), VTC and the corresponding NMIs across different corners: tt, fs, and sf</td>
<td>49</td>
</tr>
<tr>
<td>3.16</td>
<td>Simulation results of a NAND2 gate (B: High), VTC and the corresponding NMIs across different corners: tt, fs, and sf</td>
<td>50</td>
</tr>
<tr>
<td>3.17</td>
<td>Simulation results of a NAND2 gate (both inputs switch), VTC and the corresponding NMIs across different corners: tt, fs, and sf</td>
<td>51</td>
</tr>
<tr>
<td>3.18</td>
<td>Intrinsic NAND2 delay with respect to temperature and process variation</td>
<td>52</td>
</tr>
<tr>
<td>3.19</td>
<td>A basic bistable element (a) and its VTC (b)</td>
<td>53</td>
</tr>
<tr>
<td>3.20</td>
<td>An SRAM memory cell schematic</td>
<td>54</td>
</tr>
<tr>
<td>3.21</td>
<td>Voltage levels in the SRAM cell at the beginning of the read operation</td>
<td>55</td>
</tr>
<tr>
<td>3.22</td>
<td>Voltage levels in the SRAM cell at the beginning of the write operation</td>
<td>56</td>
</tr>
<tr>
<td>3.23</td>
<td>Leakage current in SRAM cells</td>
<td>57</td>
</tr>
<tr>
<td>3.24</td>
<td>Simulation result of an SRAM memory cell butterfly curve as a function of temperature, across tt process corner</td>
<td>58</td>
</tr>
<tr>
<td>3.25</td>
<td>Simulation result of an SRAM memory cell butterfly curve as a function of temperature, across ff process corner</td>
<td>59</td>
</tr>
<tr>
<td>3.26</td>
<td>Simulation result of an SRAM memory cell butterfly curve as a function of temperature, across ss process corner</td>
<td>59</td>
</tr>
<tr>
<td>3.27</td>
<td>Simulation result of an SRAM memory cell butterfly curve as a function of temperature, across sf process corner</td>
<td>60</td>
</tr>
<tr>
<td>3.28</td>
<td>Simulation result of an SRAM memory cell butterfly curve as a function of temperature, across fs process corner</td>
<td>60</td>
</tr>
<tr>
<td>4.1</td>
<td>Circuit block diagram</td>
<td>64</td>
</tr>
<tr>
<td>4.2</td>
<td>e200z6 block diagram</td>
<td>67</td>
</tr>
<tr>
<td>4.3</td>
<td>e200z6 unified cache</td>
<td>70</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

4.4 AHB/APB bridge and interfaces . . . . . . . . . . . . . . . . . . . . . . . 72
4.5 AMBA-Lite system block diagram . . . . . . . . . . . . . . . . . . . . . 73
4.6 Memory layout . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 75
4.7 e200z6 interrupts handling . . . . . . . . . . . . . . . . . . . . . . . . . 77
4.8 POR block diagram . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 80
4.9 Reset manager block diagram . . . . . . . . . . . . . . . . . . . . . . . 81
4.10 Boot-Loader block diagram . . . . . . . . . . . . . . . . . . . . . . . . 82
4.11 Clock manager block diagram . . . . . . . . . . . . . . . . . . . . . . . 84
4.12 Interrupt controller block diagram . . . . . . . . . . . . . . . . . . . . 88
4.13 AS3222 SoC reset management . . . . . . . . . . . . . . . . . . . . . . 92
4.14 Watchdog timer block diagram . . . . . . . . . . . . . . . . . . . . . . 93
4.15 General Purpose Input/Output pin implementation . . . . . . . . . . . 97
4.16 General Purpose Input/Output port block diagram . . . . . . . . . . . 97
4.17 SCI Master/Slave block diagram . . . . . . . . . . . . . . . . . . . . . . 100
4.18 SPI Master block diagram . . . . . . . . . . . . . . . . . . . . . . . . . 103

5.1 Schematic of IBM MM cell . . . . . . . . . . . . . . . . . . . . . . . . . . 109
5.2 Layout of IBM MM cell . . . . . . . . . . . . . . . . . . . . . . . . . . . 109
5.3 IBM MM - 8K x 8 array symbolic representation . . . . . . . . . . . . . 111
5.4 IBM MM array clocked read operation . . . . . . . . . . . . . . . . . . . 111
5.5 IBM MM array clocked write operation . . . . . . . . . . . . . . . . . . . 112
5.6 Area of the write address decoder versus the number of words . . . . . 114
5.7 Area of the write address decoder versus the number of words . . . . . . 115
5.8 Binary Tree MUX and One-Hot Encoded MUX: area versus number of rows ................................................................. 116
5.9 Binary Tree MUX and One-Hot Encoded MUX: delay versus number of rows ................................................................. 116
5.10 A generic FF based SCM .................................................. 118
5.11 SDFF cell layouts with different driving strengths, A, E, H, and K . 120
5.12 FF based SCM: memory size versus area ............................. 121
5.13 FF based SCM: memory size versus access and setup times ........ 121
5.14 A generic Latch based SCM .......................................... 123
5.15 LATSR cell layouts with different driving strengths, E, H, and K . 124
5.16 Two non-overlapping clock scheme ................................. 125
5.17 Latch based SCM: memory size versus area ....................... 125
5.18 Latch based SCM: memory size versus access and setup times .... 126
6.1 A typical memory architecture ......................................... 129
6.2 Logic values to detect pattern sensitive faults in a given cell ........ 130
6.3 A typical BIST architecture ............................................ 133
6.4 BIST block diagram ..................................................... 135
6.5 SCI state diagram .......................................................... 137
6.6 SCI shift register and data direction .................................. 138
6.7 Core state diagram ....................................................... 139
6.8 Illustration of handshake protocol between SCI and the core ....... 140
7.1 Test chip block diagram ........................................... 145
7.2 Timing diagram during the programming phase for IBM MM + BIST . 151
7.3 Timing diagram during the testing phase for IBM MM + BIST ....... 151
7.4 Timing diagram during the programming phase for FF based SCM +
BIST ........................................................................ 153
7.5 Timing diagram during the testing phase for FF based SCM + BIST . 153
7.6 Two non-overlapping clocks used during the simulations .......... 154

A.1 NMOS device $I_D$ versus $V_{DS}$, as a function of temperature ($V_{GS}$=1.75
V), across different corners: tt, ss, ff, sf, and fs ...................... 159
A.2 NMOS device $I_D$ versus $V_{DS}$, as a function of temperature ($V_{GS}$=2.5 V),
across different corners: tt, ss, ff, sf, and fs ......................... 160
A.3 NMOS device on-current with respect to temperature, across different
corners: tt, ss, ff, sf, and fs ........................................ 161
A.4 PMOS device $I_D$ versus $V_{SD}$, as a function of temperature ($V_{SG}$=1.75
V), across different corners: tt, ss, ff, sf, and fs ..................... 162
A.5 PMOS device $I_D$ versus $V_{SD}$, as a function of temperature ($V_{SG}$=2.5 V),
across different corners: tt, ss, ff, sf, and fs ......................... 163
A.6 PMOS device on-current with respect to temperature, across different
corners: tt, ss, ff, sf, and fs ........................................ 164
A.7 The ratio of NMOS and PMOS device on-currents with respect to tem-
perature, across different corners: ss, ff, sf, and fs ............... 165
A.8 Simulation results of a NAND3 gate (A, B: High), VTC and the corre-
sponding NMs across different corners: tt, sf, and fs .............. 166
A.9 Simulation results of a NAND3 gate (A, C: High), VTC and the corre-
sponding NMs across different corners: tt, sf, and fs .............. 167
A.10 Simulation results of a NAND3 gate (B, C: High), VTC and the corresponding NMs across different corners: tt, sf, and fs  

A.11 Simulation results of a NAND3 gate (all inputs connected together), VTC and the corresponding NMs across different corners: tt, sf, and fs  

A.12 Intrinsic NAND3 delay with respect to the temperature and the process variation  

B.1 Layout of IBM MM  

B.2 Layout of the Test Chip  

B.3 Layout of the FF based SCMs  

B.4 Layout of the Latch based SCMs
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Oxide-related parasitic device capacitances</td>
<td>17</td>
</tr>
<tr>
<td>4.1</td>
<td>AS3222 SoC memory map</td>
<td>74</td>
</tr>
<tr>
<td>4.2</td>
<td>POR interface description</td>
<td>79</td>
</tr>
<tr>
<td>4.3</td>
<td>Boot-Loader register map</td>
<td>82</td>
</tr>
<tr>
<td>4.4</td>
<td>bl_status definition</td>
<td>83</td>
</tr>
<tr>
<td>4.5</td>
<td>Boot-Loader interface description</td>
<td>83</td>
</tr>
<tr>
<td>4.6</td>
<td>Clock manager register map</td>
<td>85</td>
</tr>
<tr>
<td>4.7</td>
<td>clkm_config definition</td>
<td>85</td>
</tr>
<tr>
<td>4.8</td>
<td>clkm_cnta definition</td>
<td>85</td>
</tr>
<tr>
<td>4.9</td>
<td>clkm_cntb definition</td>
<td>85</td>
</tr>
<tr>
<td>4.10</td>
<td>clkm_cntc definition</td>
<td>86</td>
</tr>
<tr>
<td>4.11</td>
<td>clkm_cntd definition</td>
<td>86</td>
</tr>
<tr>
<td>4.12</td>
<td>clkm_cntcpu definition</td>
<td>86</td>
</tr>
<tr>
<td>4.13</td>
<td>Clock manager interface description</td>
<td>86</td>
</tr>
<tr>
<td>4.14</td>
<td>Interrupt controller register map.(TBD)</td>
<td>89</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-------</td>
<td>--------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4.15</td>
<td>irq_inten definition.(TBD)</td>
<td>89</td>
</tr>
<tr>
<td>4.16</td>
<td>Interrupt controller interface description</td>
<td>90</td>
</tr>
<tr>
<td>4.17</td>
<td>Watchdog timer register map</td>
<td>94</td>
</tr>
<tr>
<td>4.18</td>
<td>wdt_cr definition</td>
<td>94</td>
</tr>
<tr>
<td>4.19</td>
<td>Watchdog timer interface description</td>
<td>95</td>
</tr>
<tr>
<td>4.20</td>
<td>General Purpose Input/Output port register map</td>
<td>98</td>
</tr>
<tr>
<td>4.21</td>
<td>gpio_config_reg1 definition.(TBD)</td>
<td>98</td>
</tr>
<tr>
<td>4.22</td>
<td>General Purpose Input/Output port interface description</td>
<td>99</td>
</tr>
<tr>
<td>4.23</td>
<td>SCI Master/Slave register map</td>
<td>101</td>
</tr>
<tr>
<td>4.24</td>
<td>sci_config definition</td>
<td>101</td>
</tr>
<tr>
<td>4.25</td>
<td>sci_command definition</td>
<td>101</td>
</tr>
<tr>
<td>4.26</td>
<td>sci_dnum definition</td>
<td>101</td>
</tr>
<tr>
<td>4.27</td>
<td>SCI Master/Slave interface description</td>
<td>102</td>
</tr>
<tr>
<td>4.28</td>
<td>SPI Master register map</td>
<td>104</td>
</tr>
<tr>
<td>4.29</td>
<td>spi_config definition</td>
<td>104</td>
</tr>
<tr>
<td>4.30</td>
<td>spi_command definition</td>
<td>104</td>
</tr>
<tr>
<td>4.31</td>
<td>spi_dnum definition</td>
<td>104</td>
</tr>
<tr>
<td>4.32</td>
<td>spi_phase definition</td>
<td>105</td>
</tr>
<tr>
<td>4.33</td>
<td>spi_clkdiv definition</td>
<td>105</td>
</tr>
<tr>
<td>4.34</td>
<td>SPI Master interface description</td>
<td>106</td>
</tr>
<tr>
<td>5.1</td>
<td>IBM MM - 8K x 8 array pin description</td>
<td>110</td>
</tr>
</tbody>
</table>
5.2 Worst case SRAM1SF timing at 2.25 V . . . . . . . . . . . . . . . . . . 112
5.3 FF based SCMs: area, access and setup time for different memory sizes 121
5.4 Latch based SCMs: area, access and setup time for different memory sizes 125

6.1 March-based test algorithms . . . . . . . . . . . . . . . . . . . . . . . 132
6.2 Fault detection for March-based test algorithms . . . . . . . . . . . . 132
6.3 User-Accessible Setup (Test Configuration) Registers . . . . . . . . 136
6.4 Configurations to program UASRs . . . . . . . . . . . . . . . . . . . . 136
6.5 The serial output frame details . . . . . . . . . . . . . . . . . . . . . . 141

7.1 Prometheus Digital-part Pin description . . . . . . . . . . . . . . . . . 149
Chapter 1

Introduction

1.1 Aims and Motivation

Considerable progress has been made in the last three decades towards characteriz-
ing silicon Complementary Metal Oxide Semiconductor (CMOS) technologies for high
temperature operation [1]. Analytical and experimental studies have lead to the in-
corporation of high temperature effects into MOS field effect transistor (FET) models.
Improvements in the modeling and design techniques of integrated circuits (ICs) oper-
ating under severe thermal environmental conditions have prompted renewed interest
on the part of a number of industries with respect to very specific applications [2].
Among these concerns are the automobile, the earth and space probes, geothermal ex-
ploration, the nuclear reactor, and the avionics [1,3,4]. In such applications, packaged
devices and circuits must typically be able to operate reliably at extreme temperatures
up to 300 °C.

Despite these improvements and the growing demand for high temperature ICs,
still, the existing CMOS device models such as BSIM and EKV [5, 6] are valid up
to a certain temperature (typically up to military range, 125 °C). Since the transistor
behavior dramatically deviates at extreme temperatures (200 - 300 °C), these models do
not reflect the correct device operation. In contrast to its insignificance for commercial
electronics which are not required to operate at these high temperatures, this, on the
other hand, is a significant drawback for afore-mentioned applications [7]. With limited
models, and tools to predict high temperature behavior of CMOS devices, designing
reliable ICs operating under severe thermal environmental conditions remains to be a
challenge.
CHAPTER 1. INTRODUCTION

Specifically, the avionics which comprises electronic systems and devices for aeronautics and astronautics relies on high temperature electronics for automotive applications such as engine and brake control systems or under-hood electronics [3]. These systems must typically be able to operate reliably for at least 50,000 hours at ambient temperatures up to 225 °C [8]. Furthermore, the conventional hydraulic systems used in flight control actuators, landing gear actuators and propulsion inverters must also operate at these temperatures. The concept introduced by the CREAM Compact Reliable Electronic integrated to Actuator and Motor project is to replace as many hydraulic power sources and complicated circuit of high-pressure hydraulic lines as possible by Electro-Mechanical Actuators (EMA). The goal is to reach new high performance and reliability capabilities of EMA in harsh thermal environmental conditions ready to use in all-electric aircraft (AEA). Immediate benefits derived from the wider application of electrical power and electronics in actuation include higher performances and reliability, benefits of overall weight reduction, easier maintainability, reducing operating costs (including reduced fuel burn) and enhanced safety. For this global objective, it is targeted to develop an advanced, smart, miniaturized and reliable electronic technological platform integrating new compact technologies, advanced components and assembly methods, and new methodology able to substantially improve the drive and control electronic modules and the EMA motors in order to:

- To provide high power density and compact characteristics of electronics modules integrated in actuators or motors (reduction by a factor 2 of the electronic volume and mass)
- To provide advanced new concept of thermal management of the electronic platform allowing higher performances and reliability
- To provide high temperature and compact motor for actuators (reduction of 30% of the motor volume and mass)
- To integrate the new electronic and motor platform in actuator housing and a very severe thermal environment (above 200 °C) providing performing thermal management

Accepting severe thermal environment (> 200 °C) while maintaining a high reliability of the electronic module and motor also necessitates the design of reliable electronic control units which consist of processor cores, on-chip memory blocks and the necessary peripherals. The performance of these circuits are severely degraded due to the high temperature effects on the transistor characteristics. Indeed, their reliability becomes a critical issue. Given the limited capability of the device models and the tools to design digital circuits as well as the reliability issues arisen at high temperatures, implementation of control units becomes a challenge. Memory modules, on the other
CHAPTER 1. INTRODUCTION

hand, such as on-chip Static Random Access Memories (SRAMs) for processor cache units are more prone to the failures compared to other digital blocks, which make them the major bottlenecks for the reliable operation of the electronic control units. In other words, at high temperatures, the performance of such blocks are dictated by the memory modules. Therefore, the work done for this thesis focuses on the design of reliable SRAMs expected to be functional above 200 °C. The main objectives of the thesis may be listed as:

- Analyzing the high temperature effects on both the transistor and the circuit levels, particularly on the memory cells
- Assessment of the suitability and the potential of Silicon-on-Insulator (SOI) technology for high temperature applications, particularly analysis of the target technology, IBM 180 nm SOI CMOS, in terms of the robustness and the reliability
- Feasibility study for the standard-cell based SRAM macros (SCM): performance versus area trade-off analysis for various memory sizes
- Test chip preparation and analysis of the failures for different memory implementations at elevated temperatures
- Exploration of the techniques to design reliable SRAM modules for the high temperature operation

1.2 Thesis Overview

Following this brief introduction part, the rest of the thesis is formed as follows:

- **Chapter 2 Background** Overviews the basic MOSFET physics and the device operation, presents the analysis of the high temperature effects on the transistor level. It also reviews the SOI CMOS technology and highlights the major benefits of the SOI-MOSFETs in comparison with bulk transistors. Partially Depleted (PD) and Fully Depleted (FD) SOI transistors are also presented. Next, the potential and the suitability of SOI CMOS technology at high temperatures are discussed and the key features of the target technology, IBM SOI 180 nm process, are highlighted.

- **Chapter 3 High Temperature Effects on the Circuit Level** Presents the analysis of high temperature effects on the circuit level, particularly on the memory cells, and demonstrates the results of simulations performed
• **Chapter 4 System Overview** Describes the *Skeleton* system architecture, which consists of a core processor and the memory modules.

• **Chapter 5 Memory Modules** Presents the study on SCMs and describes the implementation of different memory modules including *flip-flop* and *latch* based SCMs and a technology compiled IBM SRAM macro.

• **Chapter 6 Test Structures** Discusses various memory testing algorithms, introduces BIST implementation, and other test structures for the assessment of IBM 180 nm SOI process technology.

• **Chapter 7 Test Chip** Provides the test chip details and testing methodology to be adopted, also gives proposals for the future work.

• **Chapter 8 Conclusions** Summarizes the thesis, and concludes the report.
Chapter 2

Background

This chapter overviews the basic MOSFET physics and the device operation. It describes high temperature effects in transistor level. Then, the discussion is shifted to SOI CMOS technology. The technology characteristics, advantages and drawbacks compared to conventional silicon (Bulk) CMOS technologies are explained. The chapter concludes with the discussion of the potential and suitability of SOI CMOS technology for high temperature applications and key features of the target technology, IBM 180 nm SOI CMOS.

2.1 The MOSFET Transistor

2.1.1 The Device Physics and Operation

The operation of a MOS transistor depends on the principle of controlling the electrical current flow between source and drain terminals with an applied voltage to the gate terminal. When the voltage is larger than a given value (the threshold, on, voltage $V_T$), a conducting channel is formed between source and drain ohmic contacts. The charge carriers of the conducting channel constitute an inversion charge and electrical current flows between the two contacts. Since the conductivity of the channel is modulated by the gate voltage; the larger the voltage difference between gate and source, the higher the conductance of channel and the larger current [9]. On the contrary, no such channel is formed for the applied voltages under $V_T$ and the device is considered off. Cross section of a MOS transistor is illustrated in Figure 2.1.
CHAPTER 2. BACKGROUND

Two types of MOS transistors can be identified. NMOS transistor consists of $n^+$ drain and source regions embedded in a $p$-type substrate. The current is carried by electrons moving through an $n$-type channel between source and drain. This is in contrast to the $pn$-junction diode, where the current is carried by both holes and electrons [9]. MOS devices can also be made by using an $n$-type substrate and $p^+$ drain and source regions. In such a transistor (PMOS), current is carried by holes moving through a $p$-type channel. Bulk CMOS technologies have both devices present to provide low power consumption along with high speed [10].

On the other hand, SOI CMOS technologies offer lower power consumption and higher performance at dramatically increased integration densities [10]. In fact, SOI structures do not vary much from bulk CMOS. The major difference is the insertion of the insulation layer beneath the devices. Therefore, once this is accomplished, the identical process steps can be used to fabricate such devices [11]. Besides, SOI and bulk CMOS devices have the same working principles. Hence, the models derived exclusively for bulk CMOS devices can be utilized for the analysis of SOI counterparts, with some modifications to take some different SOI characteristics into account. SOI CMOS technology will be described in detail in the following sections. Concerning high temperature, we will see that it has significant advantages that can be exploited for the proper operation of digital circuits at elevated temperatures.

For the rest of this chapter, the discussion is limited to NMOS devices. However,
the same concepts and principles also apply for PMOS devices.

Cross section of an NMOS device is shown in Figure 2.2. The fourth terminal, body (B), only serves to bias the substrate and modulate device characteristics and parameters. It is generally connected to the ground and supply voltage in NMOS and PMOS devices, respectively [9]. In order to illustrate the device operation in more detail, first, consider that all terminals are connected to ground and $V_{GS} = 0$. The drain and source are connected by back-to-back pn-junctions (substrate-source and substrate-drain). Under the mentioned conditions, both junctions have 0 V bias, which results in an extremely high resistance between drain and source [9]. Device is in cut-off mode. Now, if a positive voltage is applied to the gate ($V_{GS} > 0$), the capacitor formed by the gate and substrate plates with the gate oxide as dielectric will cause positive and negative charges to accumulate on the gate electrode and the substrate side, respectively. The latter manifests itself initially by repelling mobile holes. Hence a depletion region is formed below the gate. The depletion width ($W_d$) and the space charge per unit area ($Q_d$) is given by

$$W_d = \sqrt{\frac{2\varepsilon_{Si}\phi}{qN_A}}$$

$$Q_d = \sqrt{2qN_A\varepsilon_{Si}\phi}$$

where $N_A$ is the substrate doping concentration, $q$ is the unit (electron in this case) charge and $\phi$ is the voltage across the depletion layer (this is, the potential at the

Figure 2.2: NMOS transistor for positive $V_{GS}$, showing depletion region and induced channel
oxide-silicon boundary), and \( \varepsilon_{\text{ox}} \) is the Si permittivity [9]. As \( V_{GS} \) increases, \( \phi \) reaches a critical value at some point, and the surface inverts to n-type material. This point marks the onset of a phenomenon known as \textit{strong inversion}, which occurs at a voltage equal to twice the Fermi Potential [12] \( (\phi_F) \) given as

\[
\phi_F = \frac{kT}{q} \ln \frac{N_A}{n_i}
\]

where \( k \) denotes the Boltzmann constant, \( T \) denotes the junction temperature \( (K) \) and \( n_i \) denotes the intrinsic carrier concentration of silicon, which is also a function of temperature.

\[
n_i = \sqrt{N_C N_V e^{-\frac{E_G}{kT}}}
\]

In Eq. (2.4), \( E_G \) denotes the silicon bandgap which is the intrinsic Fermi level, and \( N_C \) and \( N_V \) denote the effective density of states of the conduction and the valence bands, respectively.

Further increases in the gate voltage produce no further changes in the depletion layer, but result in additional electrons in the thin inversion layer directly under the oxide. These are drawn into the inversion layer from the heavily doped n+ source region, which results in the formation of a continuous n-type channel between the source and drain [9]. The conductivity of this region is modulated by \( V_{GS} \). Once the inversion layer is formed, the charge stored in the depletion region \( (Q_{B0}) \) is fixed and equals

\[
Q_{B0} = \sqrt{2qN_A \varepsilon_{Si} [2\phi_F]}
\]

Now if the substrate, body, is biased at a different voltage level than the source, which is at ground potential, then the depletion region charge density can be expressed as a function of the source-to-substrate voltage \( (V_{SB} > 0 \text{ for NMOS devices}) \) [13]. In this case, the surface potential required for strong inversion increases.

\[
Q_B = \sqrt{2qN_A \varepsilon_{Si} [(-2\phi_F + V_{SB})]}
\]

The value of \( V_{GS} \) where strong inversion occurs is called \( V_T \). It is a function of several components, most of which are material constants, such as the difference in work function between gate and substrate material, the oxide thickness, \( \phi_F \), the charge of impurities trapped at the surface between channel and gate oxide, the dosage of ions implanted for threshold adjustment [9]. It also depends on some physical phenomena observed in MOS transistors such as: short and narrow channel effects, drain-induced barrier lowering (DBIL), non-uniform doping effect (in both lateral and vertical directions), and channel-length modulation [5]. All these effects are taken into account by the existing device models. However, this complicates the expression for \( V_T \). Instead, a simple, yet accurate expression for the analysis is preferred. Ignoring these effects,
and assuming that the channel doping is constant and the channel length and width are large enough [5], $V_T$ is given by

$$
V_T = V_{FB} + \phi_S + \gamma \sqrt{\phi_S + V_{SB}} = V_{T0} + \gamma \sqrt{\phi_S + V_{SB}} - \sqrt{\phi_S}
$$

(2.7)

where $V_{FB}$ is the flat band voltage, $\phi_S$ equals to $2\phi_F$, $V_{T0}$ is the threshold voltage at zero substrate bias, and $\gamma$ is the body bias coefficient given by

$$
C_{ox} = \frac{\varepsilon_{ox}}{t_{ox}}
$$

(2.9)

$$
\gamma = \frac{\sqrt{2\varepsilon_S q N_D}}{C_{ox}}
$$

(2.10)

where $C_{ox}$ is gate oxide capacitance per unit area, $\varepsilon_{ox}$ is the oxide permittivity, and $t_{ox}$ is the thickness of the oxide. Next, assume now that $V_{GS} > V_T$, and a small voltage $V_{DS}$ is applied between drain and source. This causes a drain current ($I_D$) to flow from drain to source, as illustrated in Figure 2.3.

![NMOS transistor with bias voltages](image)

**Figure 2.3:** NMOS transistor with bias voltages

At a point $x$ along the channel, the voltage is $V(x)$, and the gate-to-channel voltage at that point equals $V_{GS} - V(x)$ [9]. Assuming this voltage exceeds $V_T$ all along the channel, the induced charge per unit area ($Q_i$) at point $x$ can be computed by using

$$
Q_i(x) = -C_{ox}[V_{GS} - V(x) - V_T]
$$

(2.11)
CHAPTER 2. BACKGROUND

\( I_D \) is given as the product of the drift velocity of carriers, electrons in this case, \( \nu_n \) and the available charge \([9]\). Due to charge conservation principle, it is a constant over the length of the channel.

\[
I_D = -\nu_n(x)Q_i(x)W
\]

(2.12)

where \( W \) is the width of the channel in a direction perpendicular to the current flow. The electron velocity is related to the electric field through a parameter called the mobility \( \mu_n \) (expressed as \( m^2/V \cdot s \)). \( \mu_n \) is a complex function of crystal structure and local electrical field, and in general, an empirical value is used:

\[
\nu_n = -\mu_n \xi(x) = \mu_n \frac{dV}{dx}
\]

(2.13)

where \( \xi_i(x) \) is local electric field at point \( x \) \([9]\). Combining Eq.(2.11) through Eq.(2.13) and integrating over the length of the channel \( L \) yields the voltage-current (\( I-V \)) relation of the MOS transistor \([13]\):

\[
I_D = k'_n W L \left[ (V_{GS} - V_T)V_{DS} - \frac{V_{DS}^2}{2} \right]
\]

(2.14)

Here, \( k'_n \) is the process transconductance parameter and given by

\[
k'_n = \mu_n C_{ox}
\]

(2.15)

For smaller values of \( V_{DS} \), quadratic factor in Eq.(2.14) can be ignored, and a linear dependence between \( V_{DS} \) and \( I_D \) is observed. Thus, the operation region where Eq.(2.14) holds is called the linear region. A continuous conductive channel between source and drain is observed in this region \([9]\). As the value of \( V_{DS} \) further increases, the assumption that the channel voltage is larger than the threshold voltage all along the channel holds until \( V_{GS} - V(x) < V_T \). At this point, the induced charge is zero, and the conducting channel disappears or is pinched off in the vicinity of the drain region. This is illustrated in Figure 2.4.

For \( V_{DS} \geq V_{GS} - V_T \), the transistor is in the saturation region and Eq.(2.14) is no longer valid. The voltage difference over the induced channel (from the pinch-off point to the source) remains fixed at \( V_{GS} - V_T \), and consequently, \( I_D \) remains constant, or saturates \([9]\). Hence, replacing \( V_{DS} \) by \( V_{GS} - V_T \) in Eq.(2.14) gives \( I_D \) for the saturation mode of operation.

\[
I_D = k'_n W L \left( (V_{GS} - V_T)^2 \right)
\]

(2.16)

Eq.(2.16) shows the squared dependency of \( I_D \) with respect to \( V_{GS} \). It also suggests that \( I_D \) is no longer a function of \( V_{DS} \), which is not entirely correct. The effective length of conductive channel is actually modulated by \( V_{DS} \): increasing \( V_{DS} \) causes the depletion region at the drain junction to grow, which in turn reduces the effective channel length \([9]\). This phenomenon is referred as channel length modulation (CLM).
As can be observed from Eq.(2.16), the current is inversely proportional to channel length. Taking this effect into account yields a more accurate description of $I_D$

$$I_D = I'_D(1 + \lambda V_{DS})$$  \hspace{1cm} (2.17)

In Eq.(2.17), $I'_D$ is the current derived without considering CLM effect, and $\lambda$ is an empirical parameter to account for CLM effect [9,13]. In general, it is proportional to the inverse channel length. The transistors with shorter channel lengths have larger fraction of their channel as the drain-junction depletion region, and CLM effect is more pronounced [9,13].

Another important point that needs to be considered is the velocity saturation because it has a profound effect on the device operation. Eq.(2.13) states that the velocity of carriers is proportional to the electrical field, independent of the value of that field, in other words carrier mobility is constant. However, at high (horizontal) field strengths, the carriers fail to follow this linear model. In fact, when the electrical field along the channel reaches a critical value $\xi_c$, the velocity of carriers saturates ($\approx 10^5$ m/s) due to scattering effects (collisions suffered by the carriers) [9]. In short channel devices, this effect is more pronounced. The electric field across the channel increases rapidly and becomes significantly high in short time [12,14]. The effect of
velocity saturation can be modeled as
\[ \nu_n = \frac{\mu_n \xi}{1 + \xi / \xi_c} \quad \text{for} \quad \xi \leq \xi_c \quad (2.18) \]
\[ \nu_n = \nu_{sat} \quad \text{for} \quad \xi \geq \xi_c \quad (2.19) \]

\( \xi_c \) depends upon the doping levels and also the vertical electric field applied. The continuity requirement between the linear and saturation regions dictates that \( \xi_c = \frac{2 \nu_{sat}}{\mu_n} \) [9]. Reevaluating Eq.(2.12) and (2.13) with the revised velocity formula, \( I_D \) expression in linear region becomes
\[ I_D = \frac{\mu_n C_{ox}}{1 + (V_{DS} / \xi_c L)} \left( \frac{W}{L} \right) \left[ (V_{GS} - V_T) V_{DS} - \frac{V_{DS}^2}{2} \right] \quad (2.20) \]
\[ I_D = k'_n \left( \frac{W}{L} \right) \left[ (V_{GS} - V_T) V_{DS} - \frac{V_{DS}^2}{2} \right] \kappa(V_{DS}) \quad (2.21) \]

The \( \kappa(V_{DS}) \) factor indicates the degree of velocity saturation and is defined as
\[ \kappa(V) = \frac{1}{1 + (V/\xi_c L)} \quad (2.22) \]

In Eq.(2.20), the term \( V_{DS}/L \) can be interpreted as the average field in the channel. For long-channel devices (large values of \( L \) or small values of \( V_{DS} \), \( \kappa \) approaches to 1, and Eq.(2.20) simplifies to Eq.(2.14). On the other hand, for short-channel devices, \( \kappa \) is less than 1, meaning that \( I_D \) is less than what would normally be expected [9]. Similarly, \( I_D \) expression for saturation region becomes
\[ I_D = \kappa(V_{DSAT}) \frac{k'_n W}{L} \left[ (V_{GS} - V_T) V_{DSAT} - \frac{V_{DSAT}^2}{2} \right] (1 + \lambda V_{DS}) \quad (2.23) \]

where \( V_{DSAT} \) denotes the value of \( V_{DS} \) in which the device enters saturation, and equals \( \kappa(V_{GS} - V_T) [V_{GS} - V_T] \). For short-channel devices and for large enough values of \( V_{GS} - V_T \), \( \kappa(V_{GS} - V_T) \) is substantially less than 1, thus \( V_{DSAT} < V_{GS} - V_T \), which implies that the device enters saturation before \( V_{DS} \) reaches \( V_{GS} - V_T \) [9, 13]. Hence, short-channel devices experience an extended saturation region.

Up to this point, it has been assumed that the transistor turns off abruptly as \( V_{GS} \) drops below \( V_T \), which means the device is not conducting. In reality, for \( V_{GS} \approx V_T \), a weak inversion layer still exists and some drain current flows due mainly to diffusion between the source and the channel. Even for \( V_{GS} < V_T \), that is to say the transistor is in weak inversion, \( I_D \) is finite and it exhibits an exponential dependence on \( V_{GS} \) [15]. Called subthreshold conduction, this leakage current can be expressed as
\[ I_D = I_{S0} \left( 1 - e^{-\frac{V_{GS}}{kT/q}} \right) e^{\frac{V_{GS} - V_T - V_{off}}{nkT/q}} \quad (2.24) \]
\[ I_{S0} = \mu_n \frac{W}{L} \sqrt{\frac{q \varepsilon_s N_{ch}}{2 \phi_s}} (kT/q)^2 \quad (2.25) \]
where $V_{\text{off}}$ is the offset voltage which determines $I_D$ at $V_{DS} = 0$, $I_S$ and $n$ are empirical parameters, with $n \geq 1$, and typically ranging around 1.5 [5].

Indeed, there are also different contributors to the leakage current, $I_{\text{OFF}}$, when a MOS transistor in the supply voltage ($V_{DD}$) and ground ($GND$) path is in cut-off state [16,17]. Depending on their physical origins, they can be classified as tunneling current ($I_G$), gate-induced drain leakage ($I_{\text{GIDL}}$) current, reverse-biased pn junction current, and bulk punch-through current ($I_P$).

As seen in Figure 2.5, $I_G$ originates from the flow electrons across the thin gate oxide between the gate and the substrate due to the high electric field in the gate oxide. On the other hand, $I_{\text{GIDL}}$ flows from the drain to the substrate because of the tunneling of electrons from the valance to conduction band in the transition zone of the drain-substrate junction below the gate-to-drain overlap region where a high electric field exists [16,18]. The leakage currents associated with the reverse-biased pn junctions are due to various mechanisms such as diffusion and thermal generation in the depletion region of junctions. $I_P$ flows from the source to the drain due to lateral bipolar transistor formed by the source (emitter), bulk (base), and the drain (collector).
In general, CMOS technologies have a dominant off-state leakage mechanism and some secondary off-state leakage mechanisms. These mechanisms have evolved due to the introduction of new technology nodes with smaller feature sizes [16, 19]. For submicron technologies below 0.5 \( \mu m \), the dominant mechanism is the subthreshold leakage current. With the technology scaling, the effect of subthreshold current is more pronounced due to the lowering of \( V_T \) (primarily due to increased DIBL effect) and the increased short-channel effects [18,20–22]. In other words, short channel MOS transistors are hard to turn-off. Reverse-biased pn junction leakage and \( I_{GIDL} \) have been reported as a secondary mechanism, and similarly they increase significantly as the channel length is decreased [16,20]. On the other hand, for deep submicron technologies below 100 nm, gate tunneling currents become more of an issue. The compulsory decrease of gate dielectric thickness to improve the performance and to reduce the short-channel effects increase the gate leakage current exponentially because of the tunneling [19,20,23]. Figure 2.6 shows \( I_G \) versus \( V_{GS} \) for various oxide thicknesses.

![Figure 2.6: \( I_G \) versus \( V_{GS} \) for various oxide thicknesses, the source is assumed to be grounded [20]](image)

However, beyond the 65 nm node, this problem is partially eliminated with some High-K dielectric alternatives to replace silicon dioxide [24, 25]. These alternatives provide the same capacitance as silicon dioxide but with a much higher thickness; the gate leakage is decreased.

Among the afore-mentioned leakage current components, subthreshold current is of particular importance because it is the main contributor to \( I_{OFF} \) [16,20]. The presence of subthreshold current results in the deviation from the ideal switch-like behavior that is preferred to assume for MOS transistors. In most digital applications, this is highly undesirable. From the operational standpoint of the digital logic and memory blocks, which relies on the switching behavior of MOS transistors, the current drop is desired to be as fast as possible once \( V_{GS} \) falls below \( V_T \) [12]. Thus, the (inverse) rate
of decline of the current with respect to $V_{GS}$ below $V_T$ is a quality measure of a device. It is quantified by the slope factor $S$, which measures the amount of $V_{GS}$ that needs to be reduced for subthreshold drain current to drop by a factor of 10, is given by [9]

$$S = n \left( \frac{kT}{q} \right) \ln 10 \quad (2.26)$$

At room temperature ($300 \, K$), the minimum theoretical value for $S$ is around 60 mV/dec, and for present CMOS technologies, it takes values in the range of 80-90 mV/dec [16]. It is further degraded by the temperature, as evident from Eq.(2.26). Since the subthreshold current inversely proportional to $V_T$ (Eq.(2.24)), it can be minimized by increasing $V_T$. As shown in Figure 2.7, for a slope factor of 85 mV/decade, the subthreshold current (used interchangeably with $I_{OFF}$) decreases by ten times if $V_T$ is increased by 85 mV. On the other hand, the increase of $V_T$ also translates into the reduction of transistor on current $I_{ON}$ (Eq.(2.20) and (2.23)), which means the decrease of device performance. The trade-off between $I_{ON}$ and $I_{OFF}$ has a significant impact on the device operation. Thus, $I_{ON}/I_{OFF}$ ratio for a given technology node is a performance metric and it should be as high as possible.

Figure 2.7: Relationship between $V_T$ and the subthreshold current [20]

So far, the steady-state behavior of the MOS transistor has been described and the key device parameters has been discussed. Next, the associated parasitic device capac-
Chapter 2. Background

Parasitic capacitances will be considered. They determine the dynamic behavior of MOS transistor. Thus, they have a significant impact on the operation of digital circuits [13].

![MOS device capacitances]

There are two types of parasitic capacitances associated with the MOS transistor: oxide-related capacitances and junction capacitances. As shown in Figure 2.8(a), three oxide related capacitances, $C_{GS}$, $C_{GD}$ and $C_{GC}$ arise as a result of the overlapping ($L_D$) between the gate and the source, between the gate and drain, and between the gate and the channel, respectively. While the first two is voltage independent, the latter depend on the bias conditions. $C_{GC}$ can be decomposed into three components; $C_{gs}$, $C_{gd}$ and $C_{gb}$ depending on the mode of operation [13]. It should be noted that $C_{gs}$ is actually the gate-to-channel capacitance seen between the gate and the source terminals; $C_{gd}$ is the gate-to-channel capacitance seen between the gate and the drain terminals; and similarly, $C_{gb}$ is the gate-to-channel capacitance seen between the gate and the bulk [13]. All oxide-related capacitances are given in the following table.

On the other hand, two junction capacitances $C_{sb}$ and $C_{db}$ are associated with the depletion charge surrounding the source-substrate and the drain-substrate junctions. Source and drain junction capacitances can be decomposed into bottom-plate and sidewall components as illustrated in Figure 2.8(b). They are highly nonlinear and strongly dependent on the bias conditions of the corresponding junctions, as well as the temperature [26]. Hence, the calculation of the associated junction capacitances is
complex and will not be described here. The detailed explanation can be found in [26].

The basic device operation is described in this section. The room temperature conditions are assumed during the analysis. However, it is also possible to incorporate high temperature effects into existing models. While for some device parameters this task is relatively straightforward, for many others it is complex and a detailed elaboration is required. In the following, high temperature effects in transistor level will be analyzed in detail. The temperature dependence of the key device parameters has a significant impact on the operation of digital circuits consisting of MOS transistors at elevated temperatures. Hence, understanding them is a key point to predict the operation of digital logic and memory blocks at high temperatures.

2.1.2 Transistor Level High Temperature Effects

MOS transistor characteristics are strongly dependent on temperature. At high temperatures, this dependence is manifested as a significant degradation of device performance [1, 2, 7, 27]. One of the main parameters responsible for this is the variation of $V_T$. As reported in [1, 7], $V_T$ exhibits an inverse linear dependence on temperature. To probe this further, recall the expression for $V_T$, Eq.(2.7). Then, the temperature dependence can be determined analytically by inserting the expressions for $V_{FB}$, $\gamma$ (Eq.(2.9)), and taking its derivative with respect to $T$, $\partial V_T / \partial T$.

$$\frac{\partial V_T}{\partial T} = -\frac{1}{2q} \frac{\partial E_G}{\partial T} + \left( 1 + \frac{\sqrt{4\varepsilon_S q N_A / \phi_F}}{C_{ox}} \right) \frac{\partial \phi_F}{\partial T}$$  \hspace{1cm} (2.27)

The term $\frac{\partial \phi_F}{\partial T}$ arises from the temperature dependence of $n_i$, which can be evaluated by using Eq.(2.4) and (2.3):
CHAPTER 2. BACKGROUND

\[ \frac{\partial \phi_F}{\partial T} = -\frac{k}{q} \ln \left( \frac{\sqrt{N_C N_V}}{N_A} \right) - \frac{kT}{q \sqrt{N_C N_V}} \frac{\partial \sqrt{N_C N_V}}{\partial T} + \frac{1}{2q} \frac{\partial E_G}{\partial T} \]  

(2.28)

Since both \( N_C \) and \( N_V \) are proportional to \( T^{3/2} \) [1, 7, 12], \( \frac{\partial \sqrt{N_C N_V}}{\partial T} \) equals to \( \frac{3}{2} \frac{\sqrt{N_C N_V}}{T} \), and substituting Eq.(2.28) into Eq. gives

\[ \frac{\partial V_T}{\partial T} = -(2m - 1) \frac{k}{q} \left[ \ln \left( \frac{\sqrt{N_C N_V}}{N_A} \right) + \frac{3}{2} \right] + m - 1 \frac{\partial E_G}{\partial T} \]  

(2.29)

where \( m \) is equal to \( 1 + \frac{\sqrt{\varepsilon \varepsilon_{\text{Si}} q N_A / \phi_F}}{C_{\text{ox}}} \), which is clearly positive, and for silicon \( \frac{\partial E_G}{\partial T} \approx 2.7 \times 10^{-4} \text{ eV/K} \) and \( \sqrt{N_C N_V} \approx 3 \times 10^{19} \text{ cm}^{-3} \) [12]. Negative \( \frac{\partial V_T}{\partial T} \) implies an inverse linear relationship between \( V_T \) and \( T \) as expected.

Another important issue is the variation of \( \mu_n \) due to the temperature. At higher temperatures, \( \mu_n \) tends to decrease because of increased phonon scattering [12,14]. Its temperature dependence is reported as

\[ \mu_n(T) \approx \mu_n(T_0) \left( \frac{T}{T_0} \right)^{-\kappa} \]  

(2.30)

where \( T_0 \) is the room temperature and \( \kappa \) is a technology dependent constant varies from 1.2 - 2.0 [14].

The temperature dependences of \( V_T \) and \( \mu_n \) have two significant impacts. One of them is the decrease of \( I_D \) at high temperatures. As seen in Eq.(2.20) and (2.23), for both linear and saturation mode of operations, a temperature increase tends to increase \( I_D \) through the variation of \( V_T \), and to decrease it through the variation of \( \mu_n \). However, the decrease due to \( \mu_n (\alpha T^\kappa) \) dominates the linear increase by \( V_T \). Thus, the net effect is the decrease of drain current at elevated temperatures. On the other hand, a temperature increase tends to increase subthreshold current exponentially through the variation of \( V_T \), and to decrease it through the variation of \( \mu_n \). Since the decrease of \( V_T \) dominates, the subthreshold current increases exponentially at elevated temperatures (Figure 2.9-2.10) [14,20,28]. Actually, this temperature dependence is expected since the subthreshold current is a diffusion mode of transport which is strongly dependent on temperature [20]. In addition, it is further increased by the variation of subthreshold slope (Eq.(2.26)) through the increase of thermal voltage \( (kT/q) \) at high temperatures.
Subthreshold current becomes more critical as the technology scales. It should be also noted that the temperature dependence of subthreshold current also varies with device scaling. As shown in Figure 2.9-2.10, subthreshold current is significantly increased for smaller channel lengths. The temperature dependence of this current is also reduced for scaled transistor [28].

Gate leakage current is also found to be increased at high temperatures [16,20]. The temperature dependence of gate leakage is not as strong as the subthreshold current. For example, at high temperatures (105 °C), the tunneling current is reported as ten times lower than the subthreshold leakage [16]. At elevated temperatures the gate leakage tends to increase due to the increase of the energy of the electrons. On the other hand, the increased scattering of electrons by the lattice vibrations counteracts.
Hence, the net effect is a weak dependence of gate leakage on temperature [16, 20].

The temperature dependence of leakage currents originated from the reverse-biased pn junction diodes are also another important point that needs to be considered. As explained before, these currents exist as a result of diffusion and thermal generation mechanisms in the depletion region, which depend strongly on temperature through $n_i$. For low to moderate temperatures, these currents are dominated by generation mechanisms and increases at a rate proportional to $n_i$. At high temperatures, they are determined primarily by diffusion and increase more rapidly at a rate proportional to $n_i^2$ [16].

The parasitic junction capacitances, $C_{sb}$ and $C_{db}$; and the saturation velocity, $\nu_n$ are also affected by the temperature. A linear model is reported in [5] for the temperature dependence of junction capacitances. At higher temperatures, their values increase which translates into a reduction in device performance. On the other hand, $\nu_n$ decreases linearly at elevated temperatures [5]. The implication of these effects is further degradation of device performance through both the decrease in saturation-mode current and the deterioration of dynamic behavior.

Consequently, at high temperatures, leakage currents start becoming comparable in magnitude to $I_{ON}$, which means a significant degradation in $I_{ON}/I_{OFF}$ ratio. This has also some implications in circuit level that will be elaborated in detail in the next chapter.
2.2 Overview of SOI CMOS Technology

SOI circuits consist of ICs made from single-device islands dielectrically isolated from the substrate and each other [11]. Each island contains a single isolated transistor. Typically, sapphire ($\text{Al}_2\text{O}_3$) and silicon dioxide ($\text{SiO}_2$) can be used as insulating materials. However, today, most of SOI CMOS technologies use $\text{SiO}_2$, since it offers higher manufacturability and superior performance compared to $\text{Al}_2\text{O}_3$ [11].

![Figure 2.11: (a) Cross-section of an SOI NMOS transistor, (b) An SOI transistor with Shallow Trench Isolation](image)

As shown in Figure 2.11(a), SOI wafers consist of three layers; a thin Si surface layer where the transistors are formed, an underlying layer of $\text{SiO}_2$, referred as the buried oxide (BOX), which provides the vertical isolation, and a bulk silicon wafer at the bottom. In addition, the lateral isolation, shallow trench isolation (STI), can also be provided by a trench oxide (Figure 2.11(b)). Since it extends all the way through the silicon device layer atop the BOX, dielectric material continuously surrounds each transistor on all sides. Therefore, complete electrical isolation is provided for each transistor with respect to all other devices, and the substrate, on the wafer [29]. The process technologies used to fabricate SOI devices are described in [29].
CHAPTER 2. BACKGROUND

2.2.1 Why SOI?

SOI technology has significant benefits compared to bulk technologies. First of all, the complete isolation of n- and p- well structures eliminates latch-up problem, which is basically the formation of parasitic bipolar transistors by a PMOS/NMOS device pair in close proximity [11]. Excessive amount of current flowing as a result of the activation of these parasitic devices may give permanent damage to both device and supply interconnects. Therefore, Latch-up is an important concern and threatens product reliability in bulk technologies. In SOI technology, PMOS and NMOS transistors can be spaced closer than the transistors built on bulk silicon wafers without the latch-up concern, which translates into the increased integration density for such devices [11,29]. Thanks to the complete isolation of each transistor from the substrate, SOI transistors are also much less vulnerable to signal noise from background cosmic ray particles which translates into decreased soft-error sensitivity [11,29].

SOI offers a way to greatly reduce leakage currents by reducing or eliminating both within a single device and between devices well below achievable in a standard bulk CMOS process [4,30]. The continuous oxide barrier prevents the junction leakage currents flowing between the well-regions and the substrate. With the removal of unused silicon between transistors, the possibility of leakage current between devices within a well (intra-well) and from well-to-well (inter-well) is also eliminated. The ability to make the source/drain junction areas of SOI transistors smaller by approximately two decades, also decreases the depletion regions of these junctions. As a result, less junction leakage current arising from the generation of carriers in these regions is produced and $I_{OFF}$ is reduced to a comparable degree [4,29,30]. Reduced source/drain junction areas of SOI transistors also results in the significant reduction of the associated parasitic junction capacitances.

The short channel effects associated to loss of control of the channel by the gate are also reduced for SOI devices [11,29]. This allows lower $V_T$ operation for SOI devices in comparison with bulk devices. The better control of $V_T$ for shorter transistor lengths also enhances further device scaling.

Subthreshold slope of transistors built on SOI wafers is much steeper than that of transistors in bulk silicon wafers. A quasi ideal subthreshold slope of 62 mV/decade has been observed in SOI transistors [31]. Therefore, $V_T$ can be reduced without any significant increase in the subthreshold leakage current, which also implies the usage of lower power-supply-voltages [29]. Improved switching characteristics along with lower $V_T$ allows drain current to significantly increase for such devices. Higher $I_D$ combined with significantly reduced parasitic junction capacitances improve speed performance of SOI transistors. The speed performance of SOI ICs is enhanced by 20-35% compared to the same circuits fabricated on bulk silicon if they are operated at
the same voltages [11,29]. Moreover, reduced leakage currents, improved subthreshold slope and turn-off characteristics make SOI process also very suitable for low power applications. There is a 35-70% reduction in power consumption in SOI ICs compared to bulk silicon ICs, if they are operated to give the same speed performance [11,29].

### 2.2.2 SOI Transistors

MOS transistors built on SOI-substrates are categorized in a number of ways. The first grouping differentiates between **thick-SOI MOS transistors** that are built on SOI-substrates whose silicon layer is thicker than 1 µm, and **thin-SOI MOS transistors** in which the silicon layer thickness is less than 1µm [29]. Thick-SOI is specifically used for high-voltage and high-power devices such as power-IC switches. However, the principal SOI material being presently used for VLSI CMOS logic and memory applications is thin-SOI [29]. Within the category of thin-SOI devices, there are **partially-depleted SOI** (PD-SOI) and **fully-depleted SOI** (FD-SOI) devices. Depending on the thickness of the silicon device layer of the SOI structure, SOI devices operate in one of these modes.

![Figure 2.12: (a) Fully-Depleted, (b) Partially-Depleted SOI NMOS transistor](image)

Figure 2.12: (a) Fully-Depleted, (b) Partially-Depleted SOI NMOS transistor

The cross-section shown in Figure 2.12(b) corresponds to a PD-SOI transistor. When the maximum gate-bias is applied to such devices, the depth of the resultant depletion region in the channel is shallower than the thickness of the silicon layer [29].
In other words, there is a quasi-neutral region which is partially depleted of mobile charge carriers under the channel. Therefore, the body of such devices is intrinsically floating unless a contact to some potential is explicitly provided [29]. If the body is connected to the ground, PD-SOI devices behave very much like bulk transistors. Otherwise, floating-body effects arise, leading to potentially detrimental consequences such as kink effect, reduction in breakdown voltage and the causing of hysteresis and instability during dynamic operations [29]. Also note that such body contact increases device area, thus, reduces density.

In FD-SOI transistors, the depth of the depletion region in the channel extends entirely through the thickness of the silicon layer, even when there is no-bias applied to the gate (Figure 2.12(a)). In addition, the junction-depth formed in the source/drain regions is also as deep as the silicon layer. There is no pn-junction at the source-BOX and drain-BOX interface. Thus, there is no depletion-region at the interface to contribute to the source-body and drain-body capacitances. This also implies that there is less overall junction-surface area, which results in reduced junction leakage currents. In FD-SOI transistors, the depletion-charge is thus constant and cannot extend further into the body as the gate-bias is increased. As a result, the applied gate-bias voltage results in more inversion layer charge for the same gate-bias as in a bulk transistor. This has several benefits such as the increase of the drain current and improved subthreshold slope compared to both PD-SOI and bulk devices [29].

Despite the significant advantages over their PD-SOI counterparts, FD-SOI transistors are not yet widely implemented. This is mainly because of the requirement that the silicon layer thickness must always remain well below the depletion region width in the channel for all modes of device operation, from cut-off to saturation. This dictates the use of a low $V_T$ and controlling the silicon layer thickness to within 5-10 nm, which in turn impose severe limitations on the manufacturability of FD-SOI devices [29]. On the other hand, PD-SOI devices alleviate the constraint on $V_T$ and its sensitivity, allowing the channel doping profile to be tailored to meet any desired $V_T$ value, thus easing the manufacturing problem [11,29]. Furthermore, in FD-SOI transistors, some part of the gate charge that is not balanced by charge in the depletion region will result in the accumulation of additional charge in the substrate beneath the BOX layer. This induced charge will act as a back-gate, and it will have the effect of trying to establish an inversion layer along the interface between silicon surface layer and the BOX [29]. As a result, a coupling exists between the gate and the back-gate, and electrical characteristics of the channel along the back-gate vary with the gate-bias. In fact, when silicon layer of an FD-SOI device becomes too thin, the gate eventually can no adequately longer turn-off the device [29,32]. Hence, in such cases FD-SOI devices should be fabricated with a dual gate structure and a thinned body region, which adds extra complexity to the fabrication process of such devices [11,29,32].
2.2.3 SOI Drawbacks

The advantages stated in the previous section are based on the assumption that each single transistor is isolated with trench oxide. However, if more than one transistor sit on the same island, the leakage between the transistors still exists. Depending on the junction potentials, this may have significant impacts as described before.

In PD-SOI devices, floating body effects arise due to the depletion region under the gate when the transistor operated in inversion does not extend all the way to the buried oxide interface [11,29]. Instead, a quasi-neutral region exits. If this body region is left electrically floating, PD-SOI transistor behaves slightly different in comparison with bulk devices. Particularly, the kink effect manifests itself as a kink in the I-V characteristics of PD-SOI devices [11, 29]. The dashed lines in Figure 2.13 illustrate this effect.

The kink effect mechanism can be explained as follows. When the drain potential is large enough, the accelerated channel electrons can acquire sufficient energy in the high electric field zone near the drain to create electron-hole pairs by impact ionization [29]. Drain voltages typically get this high when the transistor is in saturation mode of operation. The electrons generated by this effect move into the drain, increasing the drain current; and the generated holes migrate to the lowest potential, floating body, where they rise the potential with respect to source [29]. Hence, increase in the body potential has the same effect as biasing the substrate; $V_T$ is reduced. The decrease in $V_T$ causes an increase in the drain current which shows up as a kink in $I-V$ characteristics. It also degrades the subthreshold characteristics of PD-SOI transistors.

Although kink effect represents an annoyance in the output characteristics of PD-SOI
devices, it actually leads to the fabrication of faster digital circuits [11,29].

The floating body may also produce transient effects in the drain current of PD-SOI transistors. An overshoot and undershoot in the drain current occur when the transistor is turned on and off, respectively, until the equilibrium is established by the recombination of the excess carriers generated by the impact ionization [29].

![Image of Parasitic Bipolar Transistor]

Figure 2.14: Illustration of Parasitic Bipolar transistor

The effect of parasitic bipolar transistor is illustrated in Figure 2.14. This effect occurs when the source, body and drain of SOI transistors act as the emitter, base and collector of parasitic transistor in which the base current consists of carriers produced by the impact ionization [33,34]. This effect is more likely to occur for FD-SOI devices since body region is more depleted which means the emitter injection efficiency for the parasitic bipolar is higher. This effect manifests itself as the reduction of the breakdown voltage between the source and the drain, abnormally steep subthreshold characteristics beyond the theoretical limit, a larger $I_{OFF}$ and a smaller $V_T$ [33,34].

Another floating body effect arises for the technology nodes beyond 0.13 µm. As a result of very thin gate oxide thickness required for these generations of devices, a significant amount of gate-to-body tunneling current flows [29]. The charge introduced into the floating body modulates the body voltage and causes a history effect, which can become excessive on smaller technology nodes [35]. That is, previous transition at the input will impact the switching delay due to the modulation of source-body junction voltage. However, this effect can be minimized by appropriate device design [29].

SOI transistors are thermally insulated from the substrate by poorly heat-conducting buried-oxide layer. The thermal conductivity of the buried oxide is about two orders
of magnitude lower than that of silicon [29]. Thus, removal of excess heat generated within the SOI transistors is less efficient than in bulk technologies. This may increase the device-temperature substantially in SOI technologies in comparison with the bulk technologies [29]. Although, the heat transfer rate is lower in an SOI device, there still exists a number of paths to transport heat out. These paths include vertically through the buried oxide layer and laterally through the silicon island into the contacts and metallization [11, 29].

The increased operating temperature within an SOI transistor can impact the device performance. As explained before, the mobility of carriers will be reduced with the increasing temperature. That is, as the drain voltage increase, the drain current will also increase. This will increase the power dissipated in the device and the temperature will rise. Hence, the mobility of carriers in the channel will be degraded. This effect will manifest itself in the I-V characteristics of the device in the form of a drop in the drain current with increasing voltage [11, 29]. However, in CMOS circuits, large drain currents occur only during the switching for a very brief period of time ( < 1 ns), otherwise all devices operate in cut-off (standby) mode [29]. Therefore, for digital logic circuits that exhibit short transition times, self heating generally will not cause any problems. This is particularly a concern for circuits with high duty cycle and slow slew rate such as clock distribution tree and I/O drivers [11].

On the other hand, for scaled SOI devices self heating effect is more pronounced. This is because for such technology nodes, both the channel length and silicon film thickness are much smaller than the phonon mean free path for silicon (≈ 300 nm at room temperature), and the thermal conductivity is severely degraded due to phonon boundary scattering [11].

### 2.2.4 SOI High Temperature Considerations

Transistor level high temperature effects are described for bulk devices in Chapter 2.1.2. Due to the similarity of their structures and working principles, most of these phenomena also apply for SOI devices. For example, \( \mu_n \), \( \nu_{sat} \) and \( I_D \) vary with temperature in SOI devices like in bulk devices [30]. Some other parameters, on the other hand, vary in a different way. This is because of the fact that SOI technology has some unique properties that can be exploited for better high temperature performance than that of bulk technology. SOI devices are reported to operate at 50 - 100 °C higher temperature than bulk devices [4]. Now, let us analyze these properties in detail.

As explained before, the major issue regarding the device operation at high temperature is the leakage current, \( I_{OFF} \). It represents the limiting factor on the high
temperature functionality of MOS transistors. Most of the leakage current mechanisms are strongly dependent on temperature. For example, the subthreshold current is significantly increased due to decrease of $V_T$ and degradation of slope factor at high temperatures. Since it is the dominant leakage mechanism, this directly translates into the strong dependence of $I_{OFF}$ on temperature for bulk devices. However, this dependence is relatively weak for SOI devices. $V_T$ of SOI transistors (especially FD-SOI devices) is known to be 2-3 times less sensitive to temperature than that of bulk devices [4, 30, 36, 37]. Therefore, $I_{OFF}$ is less sensitive and a lower percentage change is observed at elevated temperatures.

Due to floating body effects described in the previous section, body-source and body-drain junctions are intentionally increased. In addition to this leakage, increased junction temperature naturally increases the junction leakage further. This reduces the body potential in cases when the source is lower in voltage than the drain. Thus, lower body potential results in higher threshold voltages, countering the conventional thermal subthreshold voltage reduction endemic to bulk technologies [11]. This results in less temperature dependence of $I_{OFF}$ particularly for PD-SOI devices. Since, floating body voltage equilibrium results in less short channel effects and temperature sensitivity as bulk, PD-SOI devices can inherently operate with lower $V_T$. This, on the other hand, also implies that with default $V_T$ uniformly lower, the subthreshold current is already elevated. In other words, while they decrease the variation of $I_{OFF}$ due to temperature, floating body effects also degrade the slope factor via the increase of subthreshold current at elevated temperatures.

The complete isolation of transistors significantly reduces (or eliminates as in FD-SOI case) the junction leakage currents both within a single device and between devices in an IC. Considering the temperature dependence of these junction currents, this means a substantial reduction in $I_{OFF}$ in SOI than in bulk transistors [4, 30, 36]. The reduction of $I_{OFF}$ in comparison with bulk devices, also implies less degradation of subthreshold slope at high temperatures. For FD-SOI devices, a rather ideal slope factor, $2.3 \frac{kT}{q}$, is reported at high temperatures [36]. However above 200 °C, subthreshold slope exponentially increases with temperature due to higher leakage [36].

One should keep in mind that for the case in which each transistor is not completely isolated from its surrounding, the leakage current between transistors still exists. Depending on the junction potentials, these junction leakage currents may become an issue regarding the SOI device operation at elevated temperatures.

At high temperatures, a reduction in floating body effects is reported in [30, 36]. This can be explained by several mechanisms: as the temperature increases, impact ionization is reduced at the drain, excess minority carrier concentration in the device body is reduced through increased recombination, and the source junction bias is re-
duced owing to an increased saturation current. Consequently, the effects of parasitic bipolar, kink, transient and history are reduced [30,36]. As explained before, reduction of kink effect also decreases the drain current in saturation mode. Self heating effect is also reduced by 30-40% due to lower mobility and drain current [36].

Another important issue regarding the SOI device performance is the diffusion capacitances. Since they are significantly reduced (or eliminated as in FD-SOI case), SOI devices outperform their bulk counterparts at elevated temperatures [11,29,36].

2.3 Target Technology

In this project, the target technology is IBM SOI 180 nm CMOS process (csoi7rf). The key technology features are:

- 180 nm lithography
- Twin-well CMOS technology on high-resistivity SOI substrate (∼1000 Ω-cm)
- 2.5V Thick Oxide CMOS (I/O): Gate oxide thickness = 5.2 nm
  - Body contacted (bc or lc), Floating body, Stacked
- Shallow Trench Isolation
- Blanket Isolation Trench feature (BTQ)
- Low resistance Co Salicided N+ and P+ polysilicon and diffusions
- Multiple Aluminum and Copper (Cu) global metal interconnection options:
  - relAM 3 to 4 levels of: M1(Cu), M2, MT, and AM
  - relDM 4 to 5 levels of: M1(Cu), M2, MT, E1(Cu), and MA
  - relLD 5 levels of: M1(Cu), MT, E1(Cu), EZ(Cu), and LD
- Stud contact (CA) and wiring level vias V1, V2, FT, E2, and VV
- Planarized passivation and interlevel dielectrics
- Wirebond or Controlled Collapse Chip Connect (C4) solder-bump terminals.

Four-level metalizing option with AM as the last metal (4AM) is selected for this technology. A cross section of this option is illustrated in Figure 2.15. Note that the figures are not drawn to scale. The standard metal levels are M1, MT, AM. Metal
level M1 is copper. Metal levels M2, MT and AM are Aluminum. PD transistors are completely isolated from the substrate and each other, which is of great importance regarding high temperature operation (Chapter 2.2.1 and 2.2.4). For added isolation and reduced parasitic capacitance the devices can be placed in TQ trench (BTQ). Compared to bulk CMOS devices, SOI devices have higher drive currents, less junction capacitance, reduced short channel effects, better subthreshold slope and no latch up. Also note that all diffusions and polysilicon are silicided for low resistivity unless the silicide formation is intentionally blocked to form a resistor.

Figure 2.15: A cross section in IBM 180 nm SOI process with 4AM option [38]

In order to prevent the body of an SOI transistor from floating to an unknown voltage, a body contact must be used. The body contact significantly increases the area of the transistor, so the use of this type of contact is restricted to devices in which floating body effects cannot be tolerated. Two such transistor types are denoted as \( \_bc \) and \( \_lc \). The difference is that in \( \_lc \) transistors body is shorted to source, whereas \( \_bc \) transistors make it possible to connect body terminal explicitly to any other potential than the source. Floating body transistors are also available, referred as \( dgnfet \) and \( dgpfet \) for NMOS and PMOS devices, respectively. All transistor types with the smallest dimensions are illustrated in the following figure. Note the area difference between the floating body and body contacted transistors.

Another important point is that the equivalent to protect diodes in bulk technologies is a diode-connected transistor which shunts current around the FET (Figure 2.17). This configuration should be used in order to protect transistor gates especially when
CHAPTER 2. BACKGROUND

2.4 Summary and Discussion

In this chapter, the basic MOS transistor physics has been reviewed. The high temperature effects in transistor level have been analyzed. The temperature dependences of key device parameters such as $\mu$, $V_T$ and parasitic diffusion capacitances dictate the performance. At high temperatures, the transistor characteristics are severely degraded. Specifically, the exponential increase of subthreshold current as well as the reduction of device on current translates into poor device performance.

Next, SOI CMOS technology has been described and its major benefits compared to bulk technologies have been explained. The potential and the suitability of SOI CMOS
technology at high temperatures have been discussed. The target technology has been defined and its key features have been highlighted. Based on this discussion, SOI transistors are expected to outperform their bulk counterparts at high temperatures due to significantly reduced leakage currents and device capacitances, as well as improved device on current.
Chapter 3

High Temperature Effects in Circuit Level

At high temperatures, the device performance is significantly reduced due to decrease of $I_{ON}$ and the increase of parasitic device capacitances. With $I_{OFF}$ increasing and being comparable to $I_{ON}$, logic levels and noise margin are severely degraded. At some temperature, there exists no more clear distinction between the transistor on and off states, which makes it impossible to switch between these two states. Therefore, digital logic and memory blocks will not function as intended at elevated temperatures. The mechanisms that are responsible for the degradation of $I_{ON} / I_{OFF}$ ratio have been described in Chapter 2. Now, their effects on the circuit level will be analyzed in detail.

3.1 Degradation of Logic Levels, Noise Margin and Performance

Logic levels in digital circuits are not represented by quantized voltage values, but rather by voltage ranges corresponding to logic-0 and logic-1 levels [13]. The minimum and maximum input voltage that can be interpreted as logic-1 (high) and logic-0 (low) are defined as $V_{IH}$ and $V_{IL}$, respectively. Similarly, the maximum and minimum output voltage that can be produced when the output level is logic-1 and logic-0 are $V_{OH}$ and $V_{OL}$, respectively. Correct interpretation of input voltages will result in one of these voltage levels at the output (Figure 3.1). The ability of a logic gate to interpret an input signal within a voltage range corresponding to either logic-1 or logic-0 levels allows digital circuits to tolerate some internal and external signal perturbations. In
other words, unless the signal perturbation at the input completely degrades the logic levels (i.e. logic-1 and logic-0 levels remain to be distinguishable), the logic gates will continue functioning under some signal perturbations. Called noise margin, the toleration of digital circuits to signal perturbations is defined as

\[ NM_L = V_{IL} - V_{OL} \]  
\[ NM_H = V_{OH} - V_{IH} \]  

where \( NM_L \) and \( NM_H \) is for the low and high signal levels. The graphical illustration of \( NM_L \) and \( NM_H \) are given in Figure 3.1. Here, the shaded areas indicate the valid regions of the input and the output voltages and NMs are indicated as the amount of variation in the signal levels that can be allowed for ensuring correct logic values at the output.

\[ \text{Figure 3.1: Definition of noise margins } NM_L \text{ and } NM_H \]

As seen in Figure 3.1, \( NM \) is basically a measure of robustness of a logic gate against external (or internal) perturbations such as noise and variation. The analytical calculation of NMs is complex and will not be provided. A detailed analysis can be found in [39]. However, it can be approximated graphically by drawing and mirroring voltage-transfer-characteristics (VTC) of a logic gate (butterfly curve), and finding the maximum possible square between them [40]. Since there two \( NM_L \) and \( NM_H \) defined, the corresponding side lengths of such squares give NMs. Butterfly curves are particularly important for the design of memory cells as we will see later. For an inverter gate, this method is illustrated in Figure 3.2.

Here, \( NM_L \) is shown but \( NM_H \) can be obtained similarly. For the logic gates with two or more inputs, VTC with respect to each input should be drawn and the corresponding NMs should be calculated separately. By this way, \( NM \) of such gates can be determined since it is dictated by the minimum \( NM_L \) or \( NM_H \) among all possible values (i.e. for each input).
Consider a simple inverter with VTC illustrated below, which is a regenerative logic gate in the sense that it produces the output logic level from one of the rails, \( V_{DD} \) or \( GND \) depending on the input logic level. Also note regenerative term will be used for such circuits throughout this thesis.

\[
\frac{\partial V_{OUT}}{\partial V_{IN}} = -1 \quad (3.3)
\]
When logic-1 (logic-0) is applied to the input, logic-0 (logic-1) is produced at the output from the GND ($V_{DD}$) rail. In this case, NMOS (PMOS) transistor, denoted as M1 (M2), is conducting, and PMOS (NMOS) transistor is off with a small $I_{OFF}$ (5-6 orders of magnitude lower). As the temperature increases, the leakage of M2 (M1) increases and $I_{ON}$ of M1 (M2) decreases. Namely, it will be more difficult for M1 (M2) to drive logic-0 (logic-1) at the output since M2 (M1) will try to counteract with an elevated $I_{OFF}$. That implies a shift of $V_{IH}$ ($V_{IL}$); namely the switching characteristics of the inverter is degraded. If the temperature keeps increasing, eventually $I_{OFF}$ becomes comparable to $I_{ON}$ [30], and the devices can not be turned-off anymore. In such a case, there will be no reasonable logic levels produced at the output. This will translate into the degradation of both $V_{OH}$ and $V_{OL}$.

![Figure 3.4: Chain of inverters](image)

Now, imagine inverter A is driving another inverter stage, B (Figure 3.4). If the degradation of $V_{OL}$ ($V_{OH}$) at the output of inverter A results in a voltage level that is larger than $V_{IL}$ ($V_{IH}$) of inverter B, the following stage will not be able to distinguish the input voltage level; it fails to produce logic-1 (logic-0) at the output. Therefore, $V_{IL}$ ($V_{IH}$) is the maximum (minimum) allowable voltage at the input of inverter B to ensure a logic-1 (logic-0) at the output. The reduction of $I_{ON}/I_{OFF}$ ratio and degradation of logic levels at high temperatures also cause the erosion of NM [20,30,41]. In [41], NM is approximated as

$$NM \approx \frac{kT}{q}(n \ln \gamma - 1)$$

(3.4)

Here $n$ is given as the subthreshold slope (S), $\gamma$ is given as $\frac{I_{peak}}{I_{leak}}$ where $I_{peak}$ and $I_{leak}$ denotes the peak current and the leakage current ($\approx$ subthreshold current) drawn from the supply during the transition at the output of a logic gate, respectively. Both currents depend on $V_{DD}$ and transistor dimensions. Meanwhile, $I_{leak}$ also depends on both transition time at the input of the corresponding gate and $I_{ON}$. However, due to ln term, the decrease of NM with respect to temperature can not be inferred directly from Eq.(3.4). Depending on the amount of decrease of $I_{ON}/I_{OFF}$, as well as the dimensions of the devices involved in the output transition, NM is modified, accordingly. Nevertheless, one can still conclude that significant reduction of $\gamma$ at high temperatures cause degradation of NM. Thus, the reliability of digital logic circuits is expected to be reduced at high temperatures. In other words, the deterioration of NM may prevent digital logic circuits from functioning properly since it is the tolerance of digital blocks to signal variations. This, on the other hand, is particularly important for memory elements as will be discussed later.
The simulated VTC of an inverter gate as a function of temperature is shown in Figure 3.5(a). Note that the dashed lines for 175 °C and 225 °C are extrapolated (non-guaranteed) results because BSIMSOI model [42] used during the simulations are valid up to 125 °C. Nevertheless, they still indicate further deterioration of the inverter switching characteristics, but rather in a very optimistic way. Here, both NMOS and PMOS devices are simulated on typical (tt) corners. The point VIN = VOUT on the VTC curve is defined as the inverter threshold (switching threshold, $V_{th}$) voltage. When the input voltage is equal to $V_{th}$, however, the output voltage can actually attain any value between $V_{th}-V_{T,n}$ and $V_{th}-V_{T,p}$ depending on the size of NMOS and PMOS transistors, where $V_{T,n}$ and $V_{T,p}$ are the threshold voltages for NMOS and PMOS devices, respectively [13].
Figure 3.5: Simulation results of an inverter (corner = tt) (a) VTC, (b) the corresponding NMs
CHAPTER 3. HIGH TEMPERATURE EFFECTS IN CIRCUIT LEVEL

Assuming a symmetric inverter gate at the room temperature, $V_{th} = V_{DD}/2$, the slope of VTC decreases with temperature; $V_{IH}$ and $V_{IL}$ increases (Figure 3.5(a)). Hence, the right shift of VTC curve implies the degradation of both NMOS and PMOS devices. In other words, switching performance of the gate is degraded since a sharp VTC transition is desired for digital logic blocks. Furthermore, the symmetry of VTC is destroyed in favor of PMOS, meaning that NMOS on-current degrades more in comparison with PMOS. Also note the decrease of $NM_H$ while $NM_L$ increases (Figure 3.5(b)).

![IN/IP vs Temperature, as a function of VGS = VSG (corner = tt)](image)

Figure 3.6: Ratio of device on-currents (IN/IP) with respect to temperature (corner = tt)

In Figure 3.6, NMOS and PMOS transistors (lc) with the same dimensions are simulated (W=500 $\mu$m, L=320 $\mu$m). The ratio of device on-currents (IN/IP) drops as the temperature increases. Thus, NMOS transistor degrades more. In addition, for smaller $V_{GS}$ (or $V_{SG}$ for PMOS), the ratio is higher. This is because $V_T$ for PMOS ($\approx 620$ mV) is larger than for the NMOS ($\approx 400$ mV) and PMOS transistor starts to turn-off before the NMOS. See Appendix A.1 for detailed NMOS and PMOS I-V simulation results.

On the other hand, it is also important to include process variation during the temperature analysis because it affects VTC, as well as NM. The simulation results across different corners; ff, ss, sf, fs are provided in the following. Note that sf stands for slow NMOS and fast PMOS process, and fs stands for fast NMOS and slow PMOS case. The rest is defined similarly. As seen from Figure 3.7(a)-3.10(a), the shift of VTC varies with process variation. Especially for sf and fs (Figure 3.8(b) and 3.10(b)) where the variation between NMOS and PMOS transistors are the maximum, the intersection point of $NM_H$ and $NM_L$ varies significantly, which means the degradation of NM occurs at different temperatures. At high temperatures, the degradation of NM is maximum for the sf corner since the intersection occurs at lower temperature in comparison with other corners. This is also expected because the degradation of
NMOS transistor is already larger and with a slow process, it will be much weaker against the PMOS transistor. Note that for fs corner, NM of the gate which is dictated by \( NM_L \) for low temperatures seems to be improved. This is because \( NM_L \) increases as \( NM_H \) decreases. At high temperatures, \( NM_H \) determines the overall NM of the gate and its value is larger than \( NM_L \) at low temperatures. Given the large amount of degradation of NMOS transistor in comparison with PMOS, this is understandable since fast NMOS and slow PMOS process will compensate this effect. Although, NM seems to be increased, the slope of VTC (the device performance) is still degraded as shown in Figure 3.10(a). Another important remark is that since the location of \( V_{th} \) depends on the device dimensions, the shift of VTC, as well as the degradation of \( NM_H \) and \( NM_L \) varies accordingly as the temperature increases.
Figure 3.7: Simulation results of an inverter (corner = ss) (a) VTC, (b) the corresponding NMs
Figure 3.8: Simulation results of an inverter (corner = sf) (a) VTC, (b) the corresponding NMs
Figure 3.9: Simulation results of an inverter (corner = ff) (a) VTC, (b) the corresponding NMs
CHAPTER 3. HIGH TEMPERATURE EFFECTS IN CIRCUIT LEVEL

Figure 3.10: Simulation results of an inverter (corner = fs) (a) VTC, (b) the corresponding NMs
The gate delay also increases with temperature as the parasitic diffusion capacitances increase while $I_{ON}$ degrades. In Figure 3.11, the intrinsic gate delay is simulated across different process corners. The delay is the largest for ss case where both transistors are slow, as expected.

![Figure 3.11: Intrinsic inverter delay with respect to temperature and process variation](image)

Next, consider a two input NAND gate (NAND2) as illustrated in Figure 3.12. Unlike the inverter gate, which has a single VTC curve, NAND2 gate has three switching scenarios. Since logic-0 is the controlling value (output is logic 1 if one input is logic-0 regardless of the other input), the output switches when either one of the inputs is logic-1 and the other makes a transition or both of the inputs make the same transition as in the inverter gate. In the latter, NAND2 acts a non-symmetric inverter in favor of PMOS side because stacked NMOS transistors will be slower compared to parallel PMOS transistors. This manifests itself as a shift of the location of $V_{th}$ which is determined by the transistor sizes (e.g. the stronger NMOS, the more symmetric VTC curve).

Imagine a logic-1 is applied to the input A (B), and the input B (A) makes a transition from logic-0 to logic-1. In this case the output will switch from logic-1 to logic-0. The corresponding VTC will be similar to of an inverter gate (Figure 3.3), but with a non-symmetric behavior. Moreover, since $M_{P2}$ is in cut-off, there exists a subthreshold conduction during the entire transition. In other words, with $M_{P2}$ leaking, it counteracts NMOS transistors which discharges the output capacitance; but the corresponding subthreshold current ($I_{sub}$) is negligible compared to device on-currents. It becomes an issue as the temperature increases and in this case, VTC is
expected to be significantly degraded.

Similarly, NMOS transistors will be in cut-off when a logic-0 is applied to their inputs. $MN_1$, $MN_2$ or both will be leaking depending on the input pattern. However, in this case, the leakage current is expected to be less due to *stacking-effect*. Specifically, when both of the devices are in cut-off, the barrier height is modulated to be higher for the two-stack due to smaller drain to source voltage resulting in reduced leakage current [20]. The concept of stack effect is illustrated in Figure 3.13.

---

**Figure 3.12:** Schematic of a two-input NAND gate

**Figure 3.13:** Illustration of *stack effect*
These effects on the other hand, can be more pronounced for multiple-input gates. Consider, for example, a NAND gate with three inputs as illustrated in Figure 3.14. Assume that the inputs A and B are at logic-1, and C makes a transition. In this case, $M_{P1}$ and $M_{P2}$ will be off during the transition at the output. Depending on the temperature, the corresponding subthreshold leakage currents may significantly degrade the device performance, as well as the logic levels. Thus, the total subthreshold current at the output is expected to increase as the number of parallel transistors in cut-off mode increases. On the other hand, the leakage current is expected to be less for the series transistors in cut off mode because of the increased stack effect. The simulation results for a NAND3 gate are provided in Appendix A.2.

![Figure 3.14: Schematic of a three-input NAND gate](image)

In the following, NAND2 gate is simulated and the corresponding VTC curves (when the input A or B equals to logic-1, and both inputs makes a transition) as a function of temperature along with different corners (tt, fs and sf) are obtained (Figure 3.15-3.17). Note that since the degradation of NM and VTC is more pronounced in sf and fs corners, only these corners along with tt is simulated. However, the similar behavior is observed in ff and ss corners as in the inverter gate.

First, a constant logic-1 is applied to the input A and the input B is swept. Then, a constant logic-1 is applied to the input B and the input A is swept. Finally, both inputs are swept together in order to obtain the inverter equivalent of NAND2 gate.

As the simulation results indicate, the slope of VTC decreases with temperature; both $V_{IH}$ and $V_{IL}$ increases for all cases. This implies the degradation of both NMOS
and PMOS devices. The right shift of VTC curves with temperature is also expected since NMOS degrades more in comparison with PMOS. The intersection point of $N M_H$ and $N M_L$ varies significantly with respect to both the input which makes the transition and the process corner, which means the degradation of NM occurs at different temperatures.

During the transition of the input B (Figure 3.15), $N M_H$ and $N M_L$ intersect at lower temperature compared to the case in which input A switches (Figure 3.16). Hence, the degradation of NM is more pronounced at high temperatures. In addition, for sf process corner, the degradation of NM is larger in comparison with the other simulated process corners, as expected.

On the other hand, NAND2 gate behaves as an inverter when both inputs make the same transition. In this case, NM continuously decreases due to non-symmetrical VTC of the gate for all corners (Figure 3.17).
Figure 3.15: Simulation results of a NAND2 gate (A: High), VTC and the corresponding NMs across different corners: tt, fs, and sf
Figure 3.16: Simulation results of a NAND2 gate (B: High), VTC and the corresponding NMs across different corners: tt, fs, and sf
Figure 3.17: Simulation results of a NAND2 gate (both inputs switch), VTC and the corresponding NMs across different corners: tt, fs, and sf
CHAPTER 3. HIGH TEMPERATURE EFFECTS IN CIRCUIT LEVEL

The gate delay also increases with temperature as the parasitic diffusion capacitances increase while $I_{ON}$ degrades. In Figure 3.18, the intrinsic gate delay is simulated across different process corners. The falling delay is larger than the rising delay due to slower discharge path through the NMOS transistors. Besides, the switching delay from signal B connected to bottom transistor ($MN_2$) is larger than the delay from signal A connected to the top transistor $MN_1$. Hence, the intrinsic gate delay is determined by the high-to-low transition at the input B. The delay is the highest for sf case and the lowest for fs among the simulated corners, as expected.

![NAND2 Delay vs Temperature](image)

Figure 3.18: Intrinsic NAND2 delay with respect to temperature and process variation

3.2 Failure of Memory Cells

In the previous section, the high temperature effects on the regenerative logic gates are discussed. Degradation of VTC, NM and device performance at high temperatures are analyzed in detail. It was seen that these circuits will still be able to restore the correct output logic level as long as the input and the output levels are within the allowed range. On the other hand, bistable circuits which realize the memory function (also referred as cell) are more susceptible to high temperature effects. Although, these circuits consist of basic regenerative logic gates such as the inverter or the NAND gate, the degradation of device characteristics has critical implications on these circuits. Now, let us analyze these implications in detail.

As the name implies, bistable circuits have two stable states or operation modes,
logic-0 and logic-1, each of which can be attained under certain input and output conditions [13]. In digital design, bistable circuits are of particular interest since all static memories such as latch, and flip-flop circuits, registers and memory-cells consist of such elements.

The basic bistable element consists of two identical cross-coupled inverter gates as illustrated in Figure 3.19(a). Here, the output voltage of the first inverter (M1) is equal to the input voltage of the second inverter (M2), i.e., $V_{\text{OUT1}} = V_{\text{IN}2}$ and the output voltage of M2 is equal to the input voltage of M1, i.e., $V_{\text{OUT2}} = V_{\text{IN}1}$. Since the input and output voltages of M2 correspond to the output and input voltages of M1, respectively, it is possible to plot the VTC of M1 and M2 on the same axis pairs, as shown in Figure 3.19(b).

![Figure 3.19: A basic bistable element (a) and its VTC (b) [9]](image)

The circuit has only three possible operating points (A, B and C) since the two VTCs intersect at three points. Given the gain of the inverters (the slope of the respective VTC curves) in the transient region is larger than unity, C is a metastable operating point. Consequently, even if the circuit is biased at this point initially, any deviation from this bias point, possibly caused by noise, will be amplified, causing the operation point to move one of the stable operating points (A or B) in which the loop gain is smaller than unity. Thus, C is concluded to be an unstable operation point, meaning the circuit has two stable operating points.

Bistable circuits serve as a memory, storing either a logic-1 or a logic-0 (corresponding to positions A and B). In order to change the stored value, we must be able to bring the circuit from state A to B and vice versa. This can be achieved by making A (or B) temporarily unstable by increasing the loop gain to a value larger than unity. This is generally done by applying a trigger pulse at $V_{\text{IN}1}$ or $V_{\text{IN}2}$. Assume, for example, the
CHAPTER 3. HIGH TEMPERATURE EFFECTS IN CIRCUIT LEVEL

54

A bistable circuit stores a logic-1 \((V_{IN1} = 0 \text{ V}, V_{OUT1} = V_{DD})\) in the beginning. Forcing \(V_{IN1}\) to logic-1 causes both inverters to be on simultaneously for a short time and the circuit reaches to the unstable state in which the loop gain is larger than unity. At this point, the effect of the trigger pulse is regenerated by the positive feedback and the circuit moves to the other state \((B, \text{ in this case})\). The width of the trigger pulse needs to be a little larger than the total propagation delay around the loop, which is the twice the average propagation delay of the inverters \([9]\). A more detailed explanation regarding the operation of bistable elements can be found in \([9,13]\).

Next consider an SRAM memory cell consisting of a bistable element and two access transistors as illustrated in Figure 3.20. Bistable circuit functions as the data storage element and two access transistors are used to both overwrite and to read the memory content. The access transistors are activated by a word line (WL) signal when the corresponding cell is accessed for read or write operation, connecting the cell to the complementary bit-line columns (BL or \(\overline{BL}\)).

![Figure 3.20: An SRAM memory cell schematic](image)

The SRAM cell should be sized as small as possible to achieve high memory densities. Reliable operation of the cell, on the other hand, dictates some sizing constraints. In order to be able to link the high temperature effects to memory cells, let us briefly explain the operation of an SRAM cell. Consider the data-read operation first, assuming that a logic-0 is stored in the cell as illustrated in Figure 3.21. Here, the transistors \(M_2\) and \(M_5\) are turned off, while the transistors \(M_1\) and \(M_6\) are conducting. Hence, the internal node voltages \(V_1\) and \(V_2\) are equal to 0 and \(V_{DD}\), respectively, before the access transistors \(M_3\) and \(M_4\) are turned on. In addition, BL and \(\overline{BL}\) are precharged to \(V_{DD}\).

Once the cell is accessed and \(M_3\) and \(M_4\) are activated, the voltage level of \(\overline{BL}\) will
not show any significant variation since no current will flow through $M_4$. On the other hand, $M_1$ and $M_3$ will conduct a nonzero current and the voltage level of BL will begin to drop slightly. It is important to note that the amount of the corresponding voltage drop ($\Delta V$) is limited to a few hundred millivolts during the read phase because the BL capacitance $C_{BL}$ is typically very large [13]. This small voltage drop is detected and amplified as a stored logic-0 by the sense amplifier. Since there are many BL and $\bar{BL}$ pairs within an SRAM depending on the memory organization, the sense amplifier is shared by multiple bit lines. Hence, the connection of the sense amplifier to a BL and $\bar{BL}$ pair is controlled by a column select (CS) line. A more detailed explanation regarding the sense amplifiers in SRAM memories can be found in [13]. While $C_{BL}$ is being discharged by $M_1$ and $M_3$, $V_1$ will increase from the initial value of 0 V. Especially if $\frac{W}{L}$ ratio of the access transistor $M_3$ is large compared to the $\frac{W}{L}$ of $M_1$, $V_1$ may exceed the threshold voltage of $M_2$, forcing an unintended change of the stored state [13]. Hence, $M_1$ is generally sized larger than $M_3$ to guarantee that $M_2$ remains turned off during the read phase (3.5). A symmetrical condition also dictates the aspect ratios of $M_2$ and $M_4$ [13].

$$V_{1,max} \leq V_{T2}, V_{2,max} \leq V_{T1} \quad (3.5)$$

Now consider the write logic-0 operation, assuming that a logic-1 is stored in the SRAM cell initially (Figure 3.22). In this case, the transistors $M_1$ and $M_6$ are turned off, while the transistors $M_2$ and $M_5$ are conducting. Thus, the internal node voltages $V_1$ and $V_2$ are equal to $V_{DD}$ and 0, respectively, before the access transistors $M_3$ and $M_4$ are turned on. In the beginning of the write phase, BL is discharged to $\approx 0$ V and $\bar{BL}$ is precharged to $V_{DD}$. 

Figure 3.21: Voltage levels in the SRAM cell at the beginning of the read operation.
Once the cell is accessed and $M_3$ and $M_4$ are activated, we expect that $V_2$ remains below the threshold voltage of $M_1$ since $M_2$ and $M_4$ are sized according to condition (3.5). Consequently, $V_2$ would not be sufficient to turn on $M_1$. Therefore, in order to change the stored state, i.e., to force $V_1$ to 0 V and $V_2$ to $V_{DD}$, $V_1$ must be reduced below the threshold voltage of $M_2$. The transistors $M_3$ and $M_5$ are sized accordingly, to meet the condition (3.6). Note that a symmetrical condition also dictates the aspect ratios of $M_4$ and $M_6$.

$$V_{T2} \leq V_1, V_{T1} \leq V_2$$

(3.6)

In order to hold its data properly, the cross-coupled inverters must maintain bistable operating points. The best measure of the ability of these inverters to maintain their state is $NM$, which is the maximum amount of voltage that can be introduced at the outputs of the two inverters such that the cell retains its data [20]. In other words, $NM$ in the memory context basically quantifies the amount of voltage noise required at the internal nodes of a bistable element (cross-coupled inverters in this case) to flip the cells contents. Thus, they are required to have a large enough $NM$ to make sure that the circuit will be able to preserve its content properly [41,43].

The high temperature effects on the inverter $VTC$ and $NM$ are analyzed in the previous section. It was seen that the reduction of device on current, as well as the significant increase of leakage current, manifests itself as the degradation of both gate switching performance (the slope of $VTC$) and the reliability ($NM$, and logic levels). Both of the inverters within the cell will suffer from the high temperature effects. The closed loop operation to store the data is expected to make these effects more critical for the circuit reliability. In other words, the degradation of $VTC$ and $NM$ in one of the inverters will be cumulatively increased by the other inverter due to the positive feedback, which makes them more prone to the failure at high temperatures.
Consider the major leakage components in SRAM cells as illustrated in Figure 3.23. The leakage current in the SRAM cells mainly consists of subthreshold leakage currents flowing through turned-off NMOS and PMOS transistors and tunneling current across the thin gate oxide [13]. Assume a logic-1 is stored and the cell is not accessed, i.e., $M_3$ and $M_4$ is turned off. Given the exponential increase of the subthreshold current at high temperatures, $I_{nsub1}$ tends to decrease the voltage at node 1 ($V_1$). Besides, despite its weak temperature dependence, $I_{G2}$ will also have a similar effect on $V_1$. If $V_1$ is reduced below the threshold voltage of $M_2$, the cell content will flip. In the meantime, the voltage at node 2 ($V_2$) tends to increase as a result of the combined effect of $I_{psub6}$, $I_{nsub4}$ and $I_{G5}$, which also facilitates the possible cell failure.

![Figure 3.23: Leakage current in SRAM cells](image)

Also note that the effect of leakage current is more critical for SRAM cells in comparison with the other static memory elements such as latch, and flip-flop circuits. This is mainly because of the sizing constraints imposed by (3.5, 3.6). Very small SRAM cell sizes, as well as very close spacing of the cells, to achieve high memory densities also translates into higher susceptibility to the effects of leakage current. In addition, the operation of sense amplifiers, which is based on detecting of small voltage differences between the bit lines, is also expected to be disturbed due to the increase of leakage current. Consequently, the cell content may be interpreted inaccurately even though the correct value is preserved.

In the following, an IBM SRAM memory cell VTC is simulated as a function of temperature along with different corners (tt, ff, ss, sf and fs) and the corresponding
butterfly curves are obtained. As seen by the simulations (Figure 3.24 - 3.28), VTC of the memory cell is non-symmetric in favor of NMOS transistor due the sizing constraint (3.5), meaning that $V_{th}$ is much smaller than $V_{DD}/2$. Therefore, $NM_L$ is much smaller than $NM_H$. In all corners, a degradation of VTC slope is observed. NM is approximated as the side of the largest embedded square in the corresponding butterfly curve. According to this, the maximum degradation of NM with respect to temperature is measured as $\approx 36$ mV, $45$ mV, $28$ mV, $29$ mV, and $65$ mV in tt, ss, ff, sf, and fs corners, respectively. Among all simulated process corners, the maximum degradation is observed in fs corner as expected. This is because $V_{th}$ is already below $V_{DD}/2$ due to the sizing constraint, and the faster NMOS transistor and the slower PMOS transistor, as well as NMOS transistor degrading more, result in the maximum change of NM with respect to temperature. Also note that NM of the memory cell is already much smaller than that of the regenerative logic gates, i.e., the inverter and the NAND gates. Therefore, memory cells are expected to be more prone to the failure at elevated temperatures.

Figure 3.24: Simulation result of an SRAM memory cell butterfly curve as a function of temperature, across tt process corner
CHAPTER 3. HIGH TEMPERATURE EFFECTS IN CIRCUIT LEVEL

Figure 3.25: Simulation result of an SRAM memory cell butterfly curve as a function of temperature, across ff process corner

Figure 3.26: Simulation result of an SRAM memory cell butterfly curve as a function of temperature, across ss process corner
Figure 3.27: Simulation result of an SRAM memory cell butterfly curve as a function of temperature, across sf process corner

Figure 3.28: Simulation result of an SRAM memory cell butterfly curve as a function of temperature, across fs process corner
3.3 Summary and Discussion

In this chapter, the high temperature effects in the circuit level have been described in light of the grounds established in Chapter 2. The critical parameters for the reliable circuit operation at high temperatures have been highlighted.

First, the high temperature operation of regenerative logic circuits has been elaborated. An inverter, a two- and three-input NAND gates has been used during the analysis. In addition, the simulations have been performed across different corners in order to see the effect of process variation on the high temperature effects.

The high temperature effects on bistable circuits have been analyzed. A basic cross-coupled inverter gates have been used during the analysis. Understanding the behavior of bistable elements are of great importance since static memories consist of these elements. Next, possible failure modes have been described for SRAM cells and the corresponding butterfly curves across various corners as a function of temperature have been obtained. An IBM SRAM memory cell has been used in the simulations. The leakage currents in SRAM cells have also been analyzed in detail.

The high temperature effects on bistable elements are expected to be more critical compared to regenerative logic gates due to the closed loop operation, i.e., internal positive feedback mechanism. Moreover, among bistable elements, SRAM cells are expected be more susceptible to high temperature effects compared to other memory elements such as flip-flop or latch, due to some design constraints.
Chapter 4

System Overview

This chapter overviews the system architecture of the electronic control unit (Skeleton) to be implemented for CREAM project [44].

4.1 Overview

4.1.1 General Description

The AS3222 SoC, which is a 32-bit core processor based on the AMBA system, is preferred for control systems applications, such as motor control in high-temperature environment (200°C). In addition, it contains peripherals such as On Chip Memory, Serial Interfaces, Watchdog Timer, Reset and Clock Managers. The AS3222 SoC block diagram is illustrated in Figure 4.1.

4.1.2 Features

- 100 MHz operation at 200°C:
  - Alternative frequency/temperature operation: 25 MHz at 200°C.
  - Alternative frequency/temperature operation: 100 MHz at 125°C.
- 32-bit PowerPC Core (e200z6).
- Bus Interface Unit (AMBA 2.0 v6).
• Embedded Memory (RAM & ROM).
  
  – 64-Kbyte RAM in case that technology supports high temperature operation memories.
  
  – 32-Kbyte RAM in case that the RAM memory is full-custom made.
  
  – 32-Kbyte ROM for boot.

• External Memory Interface Parallel Bus Port.

• Memory Management Unit.

• 32-Kbyte Cache Memory.
  
  – ECC maybe implemented depending on the used technology.

• Four SPI Interfaces.

• Two RS422 SCI Interfaces.

• 16-bits GPIO for test purpose 8bits Input/8bits Output.

• Reset Manager.

• Two external interrupt sources for PWM and VIT.

• Clock Manager containing configurable clock divider.

• JTAG Debug Interface (Nexus 3):
  
  – PowerTRACE II programmer/debugger compliant.
  
  – Mictor connector compliant. Voltage levels LVTTL compliant (3.3V).

• Voltage levels LVTTL compliant (3.3V).

• Output clock drive compliant to the Actel APA1000 clock specifications.

• No SEU correction implemented could be a subject for future projects:
  
  – ECC maybe implemented depending on the used technology.
Figure 4.1: Circuit block diagram
4.2 Application Specifications

This circuit is a part of a European research project. The main goal of this project called CREAM is to demonstrate the feasibility of the replacement of all electrical hydrostatic actuators by electromechanical actuators in the aircrafts. This concept is preferred to get rid of as many hydraulic power sources and complicated circuit of high-pressure hydraulic lines as possible for the All Electrical Aircraft (AEA).

This circuit is expected to work at a high temperature environment (target 200 °C). It is aimed to control motors in aircrafts, and the device will be located close to the motor to save extra wires.

The AS3222 system includes a high performance 32 bits core that is used to control an external device, SEFORA, which is in charge of the motor control. In addition, this system contains several serial interfaces used for the communication with the SEFORA device and also for the communication with the external control and maintenance interface. A parallel bus is used for the communication with an external high temperature EEPROM and a high temperature ADC.

4.3 Core Unit

4.3.1 CPU Description

The e200 processor family is formed by CPU cores that implement low-cost versions of the PowerPC Book E architecture. Low-cost solutions rather than maximum performance is required by deeply embedded control applications and e200z6 processors are preferred to provide these low-cost solutions.

The e200z6 is a single-issue, 32-bit Book E-compliant design with 64-bit general-purpose registers (GPRs) [45,46].

Real-time integer and single-precision, embedded numeric operations are supported by a signal processing extension (SPE) APU and embedded vector and scalar floating-point APUs. In addition, GPRs are used to provide this support.

Single-precision vector instructions are supported by the vector SPFP APUs while single-precision floating-point operations are supported by the scalar SPFP APUs using the lower 32 bits of GPRs. Floating point values are accumulated as single-precision
values in true 32-bit by the SPFP APUs which is not the case for a PowerPC ISA. Single-precision format is used with FPRs instead of a 64-bit double-precision format.

The vector and scalar SPFP APUs carry out floating-point operations on single-precision operands. These operations provide a simpler exception model than the floating-point instructions defined by the PowerPC ISA. GPRs are preferred rather than FPRs to improve the performance for converting between floating-point, integer and fractional values.

Data in the GPRs are worked on by all arithmetic instructions that execute in the core. They have been extended to 64 bits to be compatible with the vector instructions defined by the SPE and embedded vector floating-point APUs.

The e200z6 includes a 32-Kbyte unified cache and memory management unit (MMU). In addition, a Nexus Class 3+ module is integrated.

Key features of the e200z6:

- Single-issue, 32-bit Book E-compliant core
- In-order execution and retirement
- Precise exception handling
- Branch processing unit (BPU)
- Load/store unit (LSU)
- 64-bit GPR file
- \textit{AMBA}^TM AHB-Lite 64-bit system bus
- MMU with 32-entry fully associative TLB and multiple page-size support
- 32-Kbyte, 8-way set-associative unified cache
- SPE APU supports integer operations by using both halves of the 64-bit GPRs.
- Single-precision scalar floating-point APU
- Single-precision vector floating-point APU
- Nexus Class 3+ real-time development unit
- Power management
– Low-power design, extensive clock gating
– Power-saving modes
– Dynamic power management of caches, MMUs and execution units

Core Structure

Core structure is shown in Figure 4.2.

Figure 4.2: e200z6 block diagram
Memory Management Unit

The key features of the MMU:

- Motorola Book E MMU architecture compliant
- 32-bit effective address is converted to 32-bit real address using a 41-bit interim virtual address.
- 32-entry fully associative translation look-a-side buffer (TLB1) that supports 9 page sizes (4-Kbyte to 256-Mbyte).
- One 8-bit PID register (PID0) for supporting up to 255 translation IDs at any time in the TLB
- No page table format is defined, software is free to use its own page table format.
- Hardware assist for TLB miss exceptions

Sequential instruction fetches or a change in program flow (branches and interrupts) generate instruction accesses while load, store and cache management instructions generate data accesses. MMU converts effective addresses generated by the e200z6 instruction fetch, branch and load/store units to 32-bit real addresses. Then they are used for memory accesses.

The virtual and physical address space is separated into pages by the PowerPC Book E architecture. The e200z6 MMU supports 9 page sizes and a valid entry for the page covering the effective address is required to be in a TLB to provide an efficient real address translation. If there is not any existing TLB entry for the accessed addresses, instruction or data TLB errors occur.

A process ID (PID) value is determined for each of instruction or data effective addresses to create a virtual address for each access. For e200z6 MMU, the PID is 8 bits in length. In addition, at the Book E level, a single PID register is a 32-bit register. The most significant bits are read as 0 and the PID [7..0] interface signals represent the current process ID.

Cache Memory

The L1 cache has the following features:


- 32-Kbyte unified cache design
- Virtually indexed, physically tagged
- 32-byte (8-word) line size
- 32-bit address, 64-bit data
- Pseudo-round-robin replacement algorithm
- Eight-entry store buffer
- One-entry push buffer
- One-entry line fill buffer
- Hit under fill/copy back
- Parity protection

System performance is improved by the cache because it provides low-latency data to the e200z6 instruction and data pipelines. Hence, this decouples the processor performance from system memory performance.

A single bus connected to the cache is used to perform both instruction and data accesses. Virtual addresses which are the addresses from processor to the cache are used to index the cache array. The virtual-to-physical translation is achieved by MMU. It is preferred for cache tag comparison. The access hits in the cache, when there is a match between the physical address and a valid tag entry.

During the read operation, data is provided to the processor by the cache, while during the write operation, data from the processor updates the cache. The cache leads to a bus cycle on the system bus, when the access does not match with a valid cache tag entry or a write access must be written through to memory. The e200z6 cache, which is shown in Figure 4.3, is arranged as 8 ways of 128 sets with each line including 32 bytes (4 double words) of storage.

Debug Interface

Debug functions of the internal debug support in the e200z6 core such as data and instruction breakpoints and program trace modes are used to achieve hardware and software debugging.
Software debug facilities are determined by Book E. Besides these Book E-defined facilities, the e200z6 is more flexible and functional in terms of debug event counters, linked instruction and data breakpoints and finally sequential debug event detection.

There are some restrictions on functionality. Instruction address compares and data address compares do not support compare on physical addresses. In addition, data value compares are not supported.

A real-time debugging with an external Nexus class 2,3 or 4 modules is supported by the e200z6 core. The Nexus3 module is compatible with class 3 of the IEEE-ISTO 5001-2003 standard. The following features are enforced:

- Program trace through branch traces messaging (BTM).
- Data trace by means of data write messaging (DWM) and data read messaging (DRM).
- Ownership trace by means of ownership traces messaging (OTM).
- Run-time access to embedded processor registers and memory map through the JTAG port.
- Watchpoint messaging through the auxiliary pins.
• Watchpoint trigger enable of program and/or data trace messaging.
• Registers for program trace, data trace, ownership trace and watchpoint trigger.
• All features controllable and configurable through the JTAG port.

4.3.2 System Bus Architecture

The system bus is based on the AMBA 2.0 v6 specification and little endian data format is preferred for its operation. Refer to [47, 48] for AMBA details. High speed part of this bus covers the CPU, the memories and the high bandwidth devices and AHP protocol is used while low speed part of this bus covers the lower speed peripherals (SCI, GPIO) and APB protocol is used. AHP to APB bridge is used to manage the communication between the high speed and low speed part of this bus.

There are both AHP and APB signals at the interface. The communication of the AMBA bus is mainly formed by Master-Slave interfaces communication. As shown in Figure 4.4, there is an AHB Slave device and on the other hand, there is an APB Master device. Original AMBA system supports multi Master configuration but in AMBA Lite system, which is represented in Figure 4.5, allows only a single Master device which is core e200z6.
Figure 4.4: AHB/APB bridge and interfaces
Figure 4.5: AMBA-Lite system block diagram
CHAPTER 4. SYSTEM OVERVIEW

Memory Map

The AMBA system supports 32 bits wide Address line and that will result in sufficient design flexibility for AS3222 SoC Memory Map. Every AHB Slave device has been assigned with 0x10000h addresses while APB Subsystem has been assigned with 0x10000h addresses which means every APB Slave has been assigned with 0x1000h addresses. Table 4.1 presents the Memory Map.

<table>
<thead>
<tr>
<th>AMBA</th>
<th>Device</th>
<th>Base Address</th>
<th>Address Ranges</th>
<th>Wait States</th>
</tr>
</thead>
<tbody>
<tr>
<td>APB</td>
<td>APB Subsystem</td>
<td>0x10000</td>
<td>0x10000</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Reset Manager</td>
<td>0x10000</td>
<td>0x1000</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Clock Manager</td>
<td>0x11000</td>
<td>0x1000</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Watchdog Timer</td>
<td>0x12000</td>
<td>0x1000</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>SCI 1</td>
<td>0x13000</td>
<td>0x1000</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>SCI 2</td>
<td>0x14000</td>
<td>0x1000</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>SPI 1</td>
<td>0x15000</td>
<td>0x1000</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>SPI 2</td>
<td>0x16000</td>
<td>0x1000</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>SPI 3</td>
<td>0x17000</td>
<td>0x1000</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>SPI 4</td>
<td>0x18000</td>
<td>0x1000</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>GPIO</td>
<td>0x19000</td>
<td>0x1000</td>
<td>0</td>
</tr>
<tr>
<td>AHB</td>
<td>Boot Loader</td>
<td>0x20000</td>
<td>0x10000</td>
<td>0</td>
</tr>
<tr>
<td>AHB</td>
<td>32-Kbyte RAM</td>
<td>0x30000</td>
<td>0x10000</td>
<td>Customizable</td>
</tr>
<tr>
<td>AHB</td>
<td>Interrupt Controller</td>
<td>0x40000</td>
<td>0x10000</td>
<td>0</td>
</tr>
<tr>
<td>AHB</td>
<td>Parallel Bus Port - EEPROM</td>
<td>0x50000</td>
<td>0x10000</td>
<td>0 to 8</td>
</tr>
<tr>
<td>AHB</td>
<td>Parallel Bus Port - ADC</td>
<td>0x60000</td>
<td>0x10000</td>
<td>0 to 25</td>
</tr>
<tr>
<td>AHB</td>
<td>Parallel Bus Port CS3</td>
<td>0x70000</td>
<td>0x10000</td>
<td>Customizable</td>
</tr>
<tr>
<td>AHB</td>
<td>Parallel Bus Port CS4</td>
<td>0x80000</td>
<td>0x10000</td>
<td>Customizable</td>
</tr>
</tbody>
</table>

Table 4.1: AS3222 SoC memory map

4.3.3 Peripherals Description

RAM

Single port RAM with a size of 32-Kbyte is formed by 8K words of 32 bits and has a one cycle access time. Whole memory separated into 4-Kbyte sectors, every software routine with its own memory can be accessed without disturbing the memories assigned
to the other routines. MMU achieves the new memory sector access. Memory sector is changed by a software routine. MMU controlling registers contain the ”New Victim” address value for the new sector and it is handled by the software routine. As the RAM memory mentioned is single byte addressable, 0x8000h addresses are required to address 32-Kbyte. Memory Map indicates that although it more than required, this memory module has been assigned with 0x10000h. Figure 4.6 exhibits the memory layout for 32-Kbyte. When 64-Kbyte is considered, similar memory layout with different sizes can be used. The afore mentioned device operate at the AHB AMBA system.

![Memory layout](image)

Figure 4.6: Memory layout

Boot ROM

With the usage of Boot Loader device, the AS3222 SoC boots up the system. Boot Loader includes the procedure that is run after the start-up. The records for Boot
Loader procedure are kept in Boot ROM. Depending on the external jumper position, boot up is hardware selectable between EEPROM and SCI program execution. This device operates at the AHB AMBA system.

Interrupt Controller

When there is an interrupt, processor saves its old context and starts execution at preset interrupt handler address and these interrupt mechanisms of the e200z6 core are determined by Book E Architecture. Processor state information is held in the MSR while save/restore registers are used to keep the address at which execution should continue after the interrupt is handled.

Several registers are preferred for save/restore operations. SRR0/SRR1 are preferred for non-critical interrupts, CSRR0/CSRR1 are used for critical interrupts while DSRR0/DSRR1 are preferred for debug interrupts when the debug APU is enabled.

Interrupt vector prefix register (IVPR) and interrupt-specific interrupt vector offset register (IVOR\_n) are used to determine the address where the processor starts executing. Processing of instructions inside the interrupt handler starts in supervisor modes.

rfi, rfci or rfdi are executed to restore their old state from their respective save/restore registers and continue with the instruction fetching after an interruption.

Interrupt request signals with the e200z6 core are represented in Figure 4.7.

Interrupt request signal is assigned by the Watch Dog Timer to the special line IVOR\_12 and it is defined by the Book E Architecture. Interrupt Controller where all other interrupt sources are assigned is basically multiplex eight interrupt request signals and result in an only one interrupt request.

SPI and SCI devices provide all eight interrupt signals for the whole system. First, SPI interrupt is occurred when the SPI communication is finished. Then SPI device provides an interrupt which informs the system that the communication is done. SCI device implements interrupt signals for both transmit and receive modes. Communication is completed, proper interrupt signal is set by the SCI.

CPU configures the Interrupt Controller functionality. This device works at the APB AMBA system.
Watch Dog Timer

When there is an infinite loop or a wrong address access, which are not expected, the Watch Dog function is achieved to prevent the system of freezing. It basically a counter which asserts a reset for the CPU for the count of larger values than the maximum value. It is self-incremented. CPU can clear it any time and also configure it. In addition, it can be enabled and disabled any time. As the interrupt controller, it also works at the APB AMBA system.
CHAPTER 4. SYSTEM OVERVIEW

SPI

Four SPI communication devices in the AS3222 SoC are the SPI Masters and one of them is connected to the Sefora ASIC. They commence and control the communication. Frame size and phase/mode properties are the considerations for configuration of the SPI. This device also operates at the APB AMBA system same as the others.

SCI

RS422 communication protocol is implemented by the Serial Communication Interface which supports sending and receiving the data at the same time. Package size, parity and STOP bit number are configurable by the CPU. It works at the APB AMBA system.

Reset Manager

Reset source is controlled and then they are applied to the CPU and peripherals by the Reset Manager. Power-on reset, external reset (RSTB pin) and Watch Dog reset are the possible reset sources. It is configured by the CPU and operates at the APB AMBA system.

GPIO Port

The General Purpose Input/Output port is used to capture stimuli and to export the results of examined operations which is mainly for development and test purpose. It is configured by the CPU and every single bit can be addressed and configured separately. Finally, it works at the APB AMBA system.

Clock Manager

Peripheral’s clock signals are generated by the Clock Manager. Low frequency clock signals required for low speed peripherals are provided by the Clock Manager which is mainly formed by a clock divider. It can be enabled or disabled at any time. It is configured by the CPU and operates at the APB AMBA system.
4.4 Power-On-Reset

Power-On-Reset (POR) is low voltage detector (LVD) and edge detector with time constant.

4.4.1 Block Description

The POR cell observes both VDD and VDDIO, each with a LVD and an edge detector. LVD, which confirms that the power supply is above a certain threshold voltage, includes a trigg delay to generate a reset pulse after a certain amount of time below threshold voltage. Edge detector preserves a reset of a few microseconds after a sharp power-up. The output signal depends on the VDD power supply. POR block diagram is shown in Figure 4.8.

Key features:

- Very sensitive to process corners.
- Must verify reset sequence in all cases.
- Must verify non-reset steady state after power-up in all cases.

4.4.2 I/O Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>I/O</th>
<th>Description</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supplies</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VDD</td>
<td>P</td>
<td>I/O</td>
<td>Core voltage supply</td>
<td>=3.3V</td>
</tr>
<tr>
<td>VDDIO</td>
<td>P</td>
<td>I/O</td>
<td>IOs voltage supply</td>
<td>=2.5V</td>
</tr>
<tr>
<td>VSS</td>
<td>P</td>
<td>I/O</td>
<td>Negative voltage supply (ground)</td>
<td>=0V</td>
</tr>
<tr>
<td>IOs</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RSTM_PORB</td>
<td>D</td>
<td>O</td>
<td>Combined POR signal (active low)</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.2: POR interface description; Type: D: digital, A: analog P: power in, PO: power out; I/O: I: input, O: output, I/O: input/output.
4.5 Reset Manager

Reset Manager device is used to combine up to three different reset signal sources and obtain a single Reset signal for the Core e200z6.

Main features of the Reset Manager:

- Up to 3 independently generated reset signal sources handled; External Reset Button, Power-On-Reset and Low Power Detection, Watchdog Timer Reset
- Reset source memorized for software purpose
- Core e200z6 uses the generated Reset Signal.
- AMBA APB Slave device.

4.5.1 Block Description

The Reset Manager device, which is shown in Figure 4.9, merges three different reset signal sources and creates a single Reset signal to be used by Core e200z6. Every reset
source could be disabled independently. On the other hand, Reset Manager cannot be disabled.

![Figure 4.9: Reset manager block diagram](image)

### 4.6 Boot Loader

Boot Loader device is used to boot up system of the AS3222 SoC. After the power-up, the procedure that is contained by the Boot Loader is executed.

Main features:

- Boot Loader procedure recorded in 32-Kbyte ROM: Boot Up code, which is based on SAGEM SCI boot up protocol, will be developed by AS/EPFL-LSM.
- Hardware selectable EEPROM/SCI program execution: ”1”-EEPROM, ”0”-SCI.
- AMBA AHB Slave device.

#### 4.6.1 Block Description

After the power-up, the Boot Loader, which determines if the system executes program from EEPROM or SCI, operates as a first device. External jumper position is used to
determine this decision. Boot Loader, whose block diagram is shown in Figure 4.10, operation principle is based on the following algorithm:

- Instructions that are addressing the Boot Loader device is executed by core e200z6 after the Power-On-Reset.
- Core e200z6 loads internal RAM with EEPROM or SCI program which depends on the position of the external jumper.
- Program execution from RAM is started by core e200z6.

![Boot-Loader block diagram](image)

**Figure 4.10: Boot-Loader block diagram**

### 4.6.2 Register Definition

<table>
<thead>
<tr>
<th>Boot Loader Register Map</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address Offset</td>
</tr>
<tr>
<td>0x00</td>
</tr>
</tbody>
</table>

Table 4.3: Boot-Loader register map
CHAPTER 4. SYSTEM OVERVIEW

<table>
<thead>
<tr>
<th>Bit #</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-2</td>
<td>-</td>
<td>R</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>BL_DONE</td>
<td>RW</td>
<td>0</td>
<td>Boot Loader Procedure Executed</td>
</tr>
<tr>
<td>0</td>
<td>BL_IP</td>
<td>R</td>
<td>0</td>
<td>Boot Loader Input Port Value</td>
</tr>
</tbody>
</table>

Table 4.4: bl_status definition

4.6.3 I/O Description

<table>
<thead>
<tr>
<th>Name</th>
<th>I/O</th>
<th>Description</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>HCLK</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>HRESETn</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>HSEL</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>HADDR[31..0]</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>HTRANS[1..0]</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>HSIZE[2..0]</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>HWDATA[63..0]</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>HREADYin</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>HRESP[1..0]</td>
<td>O</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>HRDATA[63..0]</td>
<td>O</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>HREADYout</td>
<td>O</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>BL_IP</td>
<td>I</td>
<td>Boot Loader Input Port</td>
<td>Hardware Jumper</td>
</tr>
</tbody>
</table>

Table 4.5: Boot-Loader interface description

4.7 Clock Manager

Clock Manager device provides clock signals and the AS3222 SoC transmits up to four of these clock signals. Output clock periods are based on a multiplied main clock period. Original frequency could be reduced by the configurable value. This device provides the e200z6 core with clock signal and the main clock signal could be scaled down or bypassed.

Main features:

- The main CPU clock signal provided, scaled down or bypassed.
• 50% duty-cycle.
• Up to 4 independently configurable output clock signals.
• Up to $2^{32}$ scaled down values.
• AMBA APB Slave device.

4.7.1 Block Description

Clock Manager device, which is presented in Figure 4.11, exports up to four different output clock signals. These clock signals are generated in counter fashion way and the generated clock signals have their clock period as a multiple of the main clock period. The main clock signal could be generated in the same way but also it could simply bypassed to e200z6 core. This feature is fully configurable and it is set during the initialization process.

Figure 4.11: Clock manager block diagram
4.7.2 Register Definition

Table 4.6: Clock manager register map

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>clkm_config</td>
</tr>
<tr>
<td>0x04</td>
<td>clkm_cnta</td>
</tr>
<tr>
<td>0x08</td>
<td>clkm_cntb</td>
</tr>
<tr>
<td>0x0c</td>
<td>clkm_cntc</td>
</tr>
<tr>
<td>0x10</td>
<td>clkm_cntd</td>
</tr>
<tr>
<td>0x14</td>
<td>clkm_cntcpu</td>
</tr>
</tbody>
</table>

Table 4.7: clkm_config definition

<table>
<thead>
<tr>
<th>Bit #</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-5</td>
<td>-</td>
<td>R</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>7</td>
<td>CLK_INPUT_BP</td>
<td>RW</td>
<td>0</td>
<td>Input Clock Bypassed to CPU</td>
</tr>
<tr>
<td>6</td>
<td>CLK_INPUT_EN</td>
<td>RW</td>
<td>0</td>
<td>Input Scaled Down Clock Enable</td>
</tr>
<tr>
<td>5</td>
<td>CLK_TST_BP</td>
<td>RW</td>
<td>1</td>
<td>Input Test Clock Bypassed to CPU</td>
</tr>
<tr>
<td>4</td>
<td>CLK_D_EN</td>
<td>RW</td>
<td>0</td>
<td>Output Clock D Enable</td>
</tr>
<tr>
<td>3</td>
<td>CLK_C_EN</td>
<td>RW</td>
<td>0</td>
<td>Output Clock C Enable</td>
</tr>
<tr>
<td>2</td>
<td>CLK_B_EN</td>
<td>RW</td>
<td>0</td>
<td>Output Clock B Enable</td>
</tr>
<tr>
<td>1</td>
<td>CLK_A_EN</td>
<td>RW</td>
<td>0</td>
<td>Output Clock A Enable</td>
</tr>
<tr>
<td>0</td>
<td>CLKM_EN</td>
<td>RW</td>
<td>1</td>
<td>Clock Manager Enable</td>
</tr>
</tbody>
</table>

Table 4.8: clkm_cnta definition

<table>
<thead>
<tr>
<th>Bit #</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-0</td>
<td>CNT_A</td>
<td>RW</td>
<td>0x000000000</td>
<td>Clock A Counter Value</td>
</tr>
</tbody>
</table>

Table 4.9: clkm_cntb definition

<table>
<thead>
<tr>
<th>Bit #</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-0</td>
<td>CNT_B</td>
<td>RW</td>
<td>0x000000000</td>
<td>Clock B Counter Value</td>
</tr>
</tbody>
</table>
### clkm_cntc Definition

<table>
<thead>
<tr>
<th>Bit #</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-0</td>
<td>CNT_C</td>
<td>RW</td>
<td>0x00000000</td>
<td>Clock C Counter Value</td>
</tr>
</tbody>
</table>

Table 4.10: clkm_cntc definition

### clkm_cntd Definition

<table>
<thead>
<tr>
<th>Bit #</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-0</td>
<td>CNT_D</td>
<td>RW</td>
<td>0x00000000</td>
<td>Clock D Counter Value</td>
</tr>
</tbody>
</table>

Table 4.11: clkm_cntd definition

### clkm_cntcpu Definition

<table>
<thead>
<tr>
<th>Bit #</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-0</td>
<td>CNT_CPU</td>
<td>RW</td>
<td>0x00000000</td>
<td>CPU Clock Counter Value</td>
</tr>
</tbody>
</table>

Table 4.12: clkm_cntcpu definition

### 4.7.3 I/O Description

<table>
<thead>
<tr>
<th>Name</th>
<th>I/O</th>
<th>Description</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCLK</td>
<td>I</td>
<td>APB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PRESETn</td>
<td>I</td>
<td>APB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PSEL</td>
<td>I</td>
<td>APB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PWRITE</td>
<td>I</td>
<td>APB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PADDR[7..0]</td>
<td>I</td>
<td>APB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PWDATA[31..0]</td>
<td>I</td>
<td>APB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PENABLE</td>
<td>I</td>
<td>APB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>CLK_INPUT</td>
<td>I</td>
<td>Input Clock Signal</td>
<td>Quartz</td>
</tr>
<tr>
<td>CLK_TEST</td>
<td>I</td>
<td>Input Clock Signal</td>
<td>Test Clock Signal</td>
</tr>
<tr>
<td>CLK_A</td>
<td>O</td>
<td>Output Clock Signal</td>
<td>Default Output Sefora</td>
</tr>
<tr>
<td>CLK_B</td>
<td>O</td>
<td>Output Clock Signal</td>
<td></td>
</tr>
<tr>
<td>CLK_C</td>
<td>O</td>
<td>Output Clock Signal</td>
<td></td>
</tr>
<tr>
<td>CLK_D</td>
<td>O</td>
<td>Output Clock Signal</td>
<td></td>
</tr>
<tr>
<td>CLK_CPU</td>
<td>O</td>
<td>e200z6 Clock Signal</td>
<td>System Clock</td>
</tr>
</tbody>
</table>

Table 4.13: Clock manager interface description
4.8 Interrupt Controller

Interrupts generation and handling are controlled by the AS3222 SoC using a DesignWare Intellectual Property (IP) device `dw_ahb_ictl` and it is provided by Synopsys.

Main features:

- Up to 64 normal interrupt sources to a single Interrupt Request (IRQ) signal
- Up to 8 fast interrupt sources to a single Fast Interrupt Request (FIQ) signal
- Vector port interface that allows e200z6 core to sample the vector address related with the current highest priority IRQ without an AMBA bus access.
- Configurable input and output polarity.
- Vectored interrupts generation.
- Masking.

4.8.1 Block Description

The `dw_ahb_ictl` is a configurable, vectored interrupt controller for AMBA based systems. Interrupt Controller block diagram is given in Figure 4.12. It is an AMBA 2.0-compliant Advance High-Speed Bus (AHB) slave device and is part of the DesignWare Synthesizable Components for AMBA 2. 2 to 64 normal interrupt (IRQ) sources, which are supported by the `dw_ahb_ictl`, are processed to obtain a single IRQ interrupt to the processor. 1 to 8 fast interrupt (FIQ) sources, which are supported by the `dw_ahb_ictl`, are processed to produce a single FIQ interrupt to the processor. When AHB bus interface of the `dw_ahb_ictl` is powered down, interrupts are propagated. Interrupts have to stay asserted until they are serviced and this has to be checked by the user. Software interrupts, priority filtering and vector generation are supported by IRQ interrupts. Input and output polarity are configurable. FIQ and IRQ interrupts are similar but FIQ does not contain priority filtering and vector generation.
CHAPTER 4. SYSTEM OVERVIEW

Figure 4.12: Interrupt controller block diagram
### 4.8.2 Register Definition

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>irq_inten</td>
</tr>
<tr>
<td>0x04</td>
<td>irq_intmask</td>
</tr>
</tbody>
</table>

Table 4.14: Interrupt controller register map.(TBD)

<table>
<thead>
<tr>
<th>Bit #</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-8</td>
<td>-</td>
<td>R</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>7</td>
<td>SCI_2_Rx_en</td>
<td>RW</td>
<td>0</td>
<td>Interrupt Source</td>
</tr>
<tr>
<td>6</td>
<td>SCI_2_Tx_en</td>
<td>RW</td>
<td>0</td>
<td>Interrupt Source</td>
</tr>
<tr>
<td>5</td>
<td>SCI_1_Rx_en</td>
<td>RW</td>
<td>0</td>
<td>Interrupt Source</td>
</tr>
<tr>
<td>4</td>
<td>SCI_1_Tx_en</td>
<td>RW</td>
<td>0</td>
<td>Interrupt Source</td>
</tr>
<tr>
<td>3</td>
<td>SPI_4_en</td>
<td>RW</td>
<td>0</td>
<td>Interrupt Source</td>
</tr>
<tr>
<td>2</td>
<td>SPI_3_en</td>
<td>RW</td>
<td>0</td>
<td>Interrupt Source</td>
</tr>
<tr>
<td>1</td>
<td>SPI_2_en</td>
<td>RW</td>
<td>0</td>
<td>Interrupt Source</td>
</tr>
<tr>
<td>0</td>
<td>SPI_1_en</td>
<td>RW</td>
<td>0</td>
<td>Interrupt Source</td>
</tr>
</tbody>
</table>

Table 4.15: irq_inten definition.(TBD)
I/O Description

<table>
<thead>
<tr>
<th>Name</th>
<th>I/O</th>
<th>Description</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>HCLK</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>HRESETn</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>HSEL</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>HADDR[31..0]</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>HTRANS[1..0]</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>HSIZE[2..0]</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>HWDATA[63..0]</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>HREADYin</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>HRESP[1..0]</td>
<td>O</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>HRDATA[63..0]</td>
<td>O</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>HREADYout</td>
<td>O</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>SPI1_IRQ</td>
<td>I</td>
<td>Interrupt Source</td>
<td></td>
</tr>
<tr>
<td>SPI2_IRQ</td>
<td>I</td>
<td>Interrupt Source</td>
<td></td>
</tr>
<tr>
<td>SPI3_IRQ</td>
<td>I</td>
<td>Interrupt Source</td>
<td></td>
</tr>
<tr>
<td>SPI4_IRQ</td>
<td>I</td>
<td>Interrupt Source</td>
<td></td>
</tr>
<tr>
<td>SCI1_Tx_IRQ</td>
<td>I</td>
<td>Interrupt Source</td>
<td></td>
</tr>
<tr>
<td>SCI1_Rx_IRQ</td>
<td>I</td>
<td>Interrupt Source</td>
<td></td>
</tr>
<tr>
<td>SCI2_Tx_IRQ</td>
<td>I</td>
<td>Interrupt Source</td>
<td></td>
</tr>
<tr>
<td>SCI2_Rx_IRQ</td>
<td>I</td>
<td>Interrupt Source</td>
<td></td>
</tr>
<tr>
<td>INT_PWM</td>
<td>I</td>
<td>External Interrupt Source from PWM</td>
<td></td>
</tr>
<tr>
<td>INT_VIT</td>
<td>I</td>
<td>External Interrupt Source from VIT</td>
<td></td>
</tr>
<tr>
<td>IRQ</td>
<td>O</td>
<td>Interrupt Request to CPU active high</td>
<td></td>
</tr>
<tr>
<td>IRQn</td>
<td>O</td>
<td>Interrupt Request to CPU active low</td>
<td>Optional</td>
</tr>
<tr>
<td>FIQ</td>
<td>O</td>
<td>Fast Interrupt Request to CPU active high</td>
<td>Optional</td>
</tr>
<tr>
<td>FIQN</td>
<td>O</td>
<td>Fast Interrupt Request to CPU active low</td>
<td>Optional</td>
</tr>
<tr>
<td>IRQ_ack</td>
<td>I</td>
<td>Interrupt Acknowledged by CPU</td>
<td>Optional</td>
</tr>
<tr>
<td>IRQ_addr</td>
<td>O</td>
<td>Interrupt Source address</td>
<td>Optional</td>
</tr>
<tr>
<td>IRQ_addr_vect</td>
<td>O</td>
<td>Interrupt Source address vector</td>
<td>Optional</td>
</tr>
</tbody>
</table>

Table 4.16: Interrupt controller interface description

4.9 Watchdog Timer

Appropriate program execution and running is controlled by the AS3222 SoC using a DesignWare Intellectual Property (IP) device `dw_apb_wdt` and it is provided by Syn-
Main features:

- Width of configurable Watchdog counters is 16 to 32 bits.
- Counter counts down from a preset value to 0 to point out the occurrence of a time out.
- Counter#1 is driven with the internal clock signal by Clock Manager, while Counter#2 is driven with the external clock signal (Quartz).
- If a timeout occurs the $dw_{apb\_wdt}$ can generate a system reset or first generate an interrupt and if this is not cleared by the service routine during the given time, second timeout occurs and a system reset is generated.
- Watchdog Interrupt and Reset signals are passed to the output pins.
- Programmable timeout period. To reduce the register requirements, the option of hard coding this value is obtainable during configuration.
- Optional dual programmable timeout period. It is preferred when the duration waited for the first kick is not same with the duration required for the subsequent kicks. The option of hard coding these values is also available.
- Programmable and hard coded reset pulse length.
- Unexpected restart of the $dw_{apb\_wdt}$ counter is prevented.
- Unexpected disabling of the $dw_{apb\_wdt}$ is prevented.
- Optional support for Pause mode by using the external pause enable signal.
- Test mode signal to decrease the time required for functional test.

4.9.1 Block Description

System lockup due to the conflicts parts or programs in an SoC is prevented by the $dw_{apb\_wdt}$ which is an APB slave peripheral. This component is programmed and configured according to user-defined options. Figure 4.13 represents an example of $dw_{apb\_wdt}$ peripheral. The generated interrupt is passed to the e200z6 core while generated reset is passed to a Reset Controller. Reset Controller generates the Reset signal for the components in the system which also contains the main e200z6 core Reset signal. The Watchdog Timer is also reset by the Reset Controller in our particular case. Watchdog Timer block diagram is shown in Figure 4.14.
Figure 4.13: AS3222 SoC reset management

Counting

The \texttt{dw\_apb\_wdt} counts from a timeout value and continues decrementing to 0. When the counter reaches to 0, a system reset or interrupt is happened according to the selected output response mode. The user can restart the counter to its initial value and it is programmable by restart register. The process of restarting the Watchdog counter is referred to as \textit{kickingthedog}. The value 0x76 has to be written to the Current Counter Value Register (\texttt{wdt\_crr}) to prevent unexpected restarts.

Watchdog Interrupt

The \texttt{dw\_apb\_wdt} can be programmed to generate an interrupt when there is a timeout. If a 1 is written to the response mode field (RMod, bit 1) of the Watchdog Timer Control register (\texttt{wdt\_cr}), the \texttt{dw\_apb\_wdt} generates an interrupt. Then, a second timeout occurs and it generates a system reset, if it is not cleared during the given
time. If a restart is happened when the watchdog counter reaches to 0, interrupt is not generated. In addition, interrupt is cleared by reading the Watchdog Timer Interrupt Clear register \((wdt\_eoi)\) without any kick. On the other hand, interrupt can also be cleared by a kick (Watchdog counter restart).

Watchdog Reset

If a 0 is written to the output response mode field (RMOD, bit 1) of the Watchdog Timer Control Register \((wdt\_cr)\), a system reset is generated by the \(dw\_apb\_wdt\) when there is a timeout. The Watchdog Timer can be configured to be always enabled on reset of the \(dw\_apb\_wdt\). For this case, it always invalidates whatever has been written to bit 0 of the \(wdt\_cr\) register (the enable field).
4.9.2 Register Definition

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>wdt_cr</td>
</tr>
<tr>
<td>0x04</td>
<td>wdt_torr</td>
</tr>
<tr>
<td>0x08</td>
<td>wdt_ccvr</td>
</tr>
<tr>
<td>0x0c</td>
<td>wdt_crr</td>
</tr>
<tr>
<td>0x10</td>
<td>wdt_stat</td>
</tr>
<tr>
<td>0x14</td>
<td>wdt_eoi</td>
</tr>
</tbody>
</table>

Table 4.17: Watchdog timer register map

<table>
<thead>
<tr>
<th>Bit #</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-5</td>
<td>-</td>
<td>R</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>4-2</td>
<td>RPL</td>
<td>RW</td>
<td>000</td>
<td>Reset Pulse Length</td>
</tr>
<tr>
<td>1</td>
<td>RMOD</td>
<td>RW</td>
<td>0</td>
<td>Response Mode</td>
</tr>
<tr>
<td>0</td>
<td>WDT_EN</td>
<td>RW</td>
<td>0</td>
<td>Watchdog Timer Enable</td>
</tr>
</tbody>
</table>

Table 4.18: wdt_cr definition
4.9.3 I/O Description

<table>
<thead>
<tr>
<th>Name</th>
<th>I/O</th>
<th>Description</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCLK</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PRESETrn</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PSEL</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PWRITE</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PADDR[7..0]</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PWDATA[31..0]</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PENABLE</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>RESETrn</td>
<td>I</td>
<td>Reset Signal from Reset Manager</td>
<td></td>
</tr>
<tr>
<td>WDT_CLK_INT</td>
<td>I</td>
<td>Clock Signal by Clock Manager</td>
<td></td>
</tr>
<tr>
<td>WDT_CLK_EXT</td>
<td>I</td>
<td>Clock Signal by External Quartz</td>
<td>Optional</td>
</tr>
<tr>
<td>WDT_IRQ</td>
<td>O</td>
<td>Interrupt Request to CPU</td>
<td></td>
</tr>
<tr>
<td>WDT_RST</td>
<td>O</td>
<td>Reset Signal to Reset Controller</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.19: Watchdog timer interface description

4.10 General Purpose Input/Output Port

Digital data is exchanged with off-chip devices by the AS3222 SoC using a DesignWare Intellectual Property (IP) device `dw_apb_gpio` and it is provided by Synopsys.

Main features:

- Up to 128 separately configurable signals.
- Up to 4 independently configurable ports (A to D).
- Different data registers and data direction registers for each signal.
- Configurable hardware and software control for each signal.
- Configurable interrupt mode for Port A.
- To debounce interrupts, debounce logic with an external slow clock is configurable.
- Option to generate single or multiple interrupts.
• Configurable reset values on output signals.
• Configurable synchronization of interrupt signals.

4.10.1 Block Description

The output data and direction of external I/Q pads are controlled and the data on external pads can be read back using memory-mapped registers by the dw_apb_gpio. Two methods of generating the default source of the input data, the output data and the control of each signal are software and hardware control.

Software control takes place over the APB bus interface, while hardware control is through the auxiliary hardware control interface. Hardware option of each signal has to be chosen during the configuration time to exist. When the hardware option is built, writing to a control register for the corresponding signal achieves to exchange between the software and hardware modes. In addition, device can be configured that user can switch between software and hardware modes individually for each bit of each signal, if the hardware mode exists. In our particular implementation, only the software control mechanism, which is implemented on Port A, is used.

APB read of the memory-mapped register is used to read the data on the external gpio port, gpio_ext_portx. The data on the gpio_ext_portx control lines or the contents of the gpio_swportx_dr which depends on the value of the gpio_swportx_ddr are provided by an APB read to the gpio_ext_portx register. The GPIO pin implementation and the GPIO port block diagram are given in Figure 4.15 and Figure 4.16 respectively.
CHAPTER 4. SYSTEM OVERVIEW

Figure 4.15: General Purpose Input/Output pin implementation

Figure 4.16: General Purpose Input/Output port block diagram
Register Definition

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>gpio_swporta_dr</td>
</tr>
<tr>
<td>0x04</td>
<td>gpio_swporta_ddr</td>
</tr>
<tr>
<td>0x08</td>
<td>gpio_swporta_ctl</td>
</tr>
<tr>
<td>0x30</td>
<td>gpio_inten</td>
</tr>
<tr>
<td>0x34</td>
<td>gpio_intmask</td>
</tr>
<tr>
<td>0x40</td>
<td>gpio_intstatus</td>
</tr>
<tr>
<td>0x48</td>
<td>gpio_debounce</td>
</tr>
<tr>
<td>0x4c</td>
<td>gpio_porta_eoi</td>
</tr>
<tr>
<td>0x50</td>
<td>gpio_ext_porta</td>
</tr>
<tr>
<td>0x74</td>
<td>gpio_config_reg1</td>
</tr>
</tbody>
</table>

Table 4.20: General Purpose Input/Output port register map

<table>
<thead>
<tr>
<th>Bit #</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-14</td>
<td>-</td>
<td>R</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>13</td>
<td>DEBOUNCE</td>
<td>RW</td>
<td>0</td>
<td>Debounce Mode Enable</td>
</tr>
<tr>
<td>12</td>
<td>PORTA_INTR</td>
<td>RW</td>
<td>0</td>
<td>PortA interrupt Enable</td>
</tr>
<tr>
<td>11-4</td>
<td>-</td>
<td>R</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>3-2</td>
<td>NUM_PORTS</td>
<td>RW</td>
<td>00</td>
<td>Number of Used Ports</td>
</tr>
<tr>
<td>1-0</td>
<td>APB_DATA_WIDTH</td>
<td>RW</td>
<td>00</td>
<td>APB Data Width</td>
</tr>
</tbody>
</table>

Table 4.21: gpio_config_reg1 definition. (TBD)
4.10.2 I/O Description

<table>
<thead>
<tr>
<th>Name</th>
<th>I/O</th>
<th>Description</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCLK</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PRESETn</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PSEL</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PWRITE</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PADDR[7..0]</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PWDATA[31..0]</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PENABLE</td>
<td>I</td>
<td>AHB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>GPIO_IRQ[7..0]</td>
<td>O</td>
<td>Interrupt Request to Interrupt Controller</td>
<td>Disabled by default</td>
</tr>
<tr>
<td>PORTA[7..0]</td>
<td>IO</td>
<td>GPIO PortA</td>
<td>Input[7..4]/Output[3..0]</td>
</tr>
</tbody>
</table>

Table 4.22: General Purpose Input/Output port interface description

4.11 Serial Communication Interface

There are two Serial Communication Interface (SCI) devices provided by the AS3222 SoC and they are SCI Master/Slave devices. If SCI Boot up is required, SCI Master device starts and controls the communication. SCI Slave device responds when maintenance of communication is needed. Both of these devices are fully configurable.

Main features:

- Two independently configurable SCI Master and Slave devices.
- Tx/Rx lines must drive 2m distance at 2MHz to the RS422 driver. The target RS422 transceiver is the MAX490 from MAXIM. This driver has to be located in the low temperature area.
- Command and Data communication mechanism.
- 16 Words Read Buffer.
- 16 Words Write Buffer.
- 8 bits fixed frame size.
- Configurable number of STOP bits.
• Configurable PARITY bit.

• When data Transmission or Reception is finished, interrupt is generated; Configurable number of frames to provoke an interrupt generation.

• AMBA APB Slave device.

4.11.1 Block Description

Figure 4.17 represents the SCI Master/Slave block diagram.

SCI Master initiates all transactions by asserting a START bit. The Data is clocked out on the middle of a bit interval on the Receiver side. Then, it is transferred serially as Most Significant Bit (MSB) first, Least Significant Bit (LSB) last. SCI Master device sends Command, Number of Data Frames and Data Frames. The communication ends with a single or double STOP bits which is supported by a control bit. This control bit could be implemented in PARITY fashion.
4.11.2 Register Definition

### SCI Master/Slave Register Map

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>sci_config</td>
</tr>
<tr>
<td>0x04</td>
<td>sci_command</td>
</tr>
<tr>
<td>0x08</td>
<td>sci_dnum</td>
</tr>
</tbody>
</table>

Table 4.23: SCI Master/Slave register map

### sci_config Definition

<table>
<thead>
<tr>
<th>Bit #</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-5</td>
<td>-</td>
<td>R</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>4</td>
<td>SCI_TX</td>
<td>RW</td>
<td>0</td>
<td>SCI Master Writes</td>
</tr>
<tr>
<td>3</td>
<td>SCI_RX</td>
<td>RW</td>
<td>0</td>
<td>SCI Master Reads</td>
</tr>
<tr>
<td>2</td>
<td>SCI_TX_RDY</td>
<td>R</td>
<td>1</td>
<td>SCI Transmitter Ready</td>
</tr>
<tr>
<td>1</td>
<td>SCI_RX_RDY</td>
<td>R</td>
<td>1</td>
<td>SCI Receiver Ready</td>
</tr>
<tr>
<td>0</td>
<td>SCI_EN</td>
<td>RW</td>
<td>1</td>
<td>SCI Enable</td>
</tr>
</tbody>
</table>

Table 4.24: sci_config definition

### sci_command definition

<table>
<thead>
<tr>
<th>Bit #</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-0</td>
<td>SCI_CMD</td>
<td>RW</td>
<td>0x00000000</td>
<td>SCI Command</td>
</tr>
</tbody>
</table>

Table 4.25: sci_command definition

### sci_dnum Definition

<table>
<thead>
<tr>
<th>Bit #</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-0</td>
<td>SCI_DNUM</td>
<td>RW</td>
<td>0x00000000</td>
<td>Number of SCI Data Frames</td>
</tr>
</tbody>
</table>

Table 4.26: sci_dnum definition
### 4.11.3 I/O Description

<table>
<thead>
<tr>
<th>Name</th>
<th>I/O</th>
<th>Description</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCLK</td>
<td>I</td>
<td>APB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PRESETn</td>
<td>I</td>
<td>APB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PSEL</td>
<td>I</td>
<td>APB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PWRITE</td>
<td>I</td>
<td>APB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PADDR[7..0]</td>
<td>I</td>
<td>APB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PWDATA[31..0]</td>
<td>I</td>
<td>APB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PENABLE</td>
<td>I</td>
<td>APB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>SCI_TX</td>
<td>O</td>
<td>SCI Transmitter Line</td>
<td></td>
</tr>
<tr>
<td>SCI_RX</td>
<td>I</td>
<td>SCI Receiver Line</td>
<td></td>
</tr>
<tr>
<td>SCLTX_IRQ</td>
<td>O</td>
<td>SCI Transmission Done Interrupt</td>
<td></td>
</tr>
<tr>
<td>SCI_RX_IRQ</td>
<td>O</td>
<td>SCI Reception Done Interrupt</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.27: SCI Master/Slave interface description

### 4.12 Serial Peripheral Interface Master

Four Serial Peripheral Interface (SPI) communication devices, which are Master devices, are provided by the AS3222 SoC. These SPI Master devices starts and control the communication. One of the SPI Master devices is connected to the SEFORA ASIC.

Main features:

- 4 separately configurable SPI Master devices.
- These SPI Master devices are compatible to the full-duplex communication.
- Configurable Word size: 16 Words Read Buffer and 16 Words Write Buffer.
- Configurable SPI phase properties.
- Configurable clock frequency in the range of 100 kHz to 10 MHz.
- Command and Data communication mechanism.
- Cyclic Redundancy check.
• When data Transmission or Reception is finished, interrupt is generated; Configurable number of frames to provoke an interrupt generation.

• AMBA APB Slave device.

4.12.1 Block Description

Figure 4.18 represents the SPI Master block diagram. SPI Master initiates all trans-
actions by asserting a falling edge on the $SPI_{CS.L}$ signal. The Data is clocked out on the rising edge of $SPI_{SCLK}$ and clocked in on the falling edge of $SPI_{SCLK}$. Then, it is transferred serially as Most Significant Bit (MSB) first, Least Significant Bit (LSB) last. $SPI_{CS.L}$ signal should not be toggle between the transmitted SPI words and it should stay at a low level (active level) during the whole SPI frame. When the SPI frame ends, it should be asserted to high level (inactive level).
4.12.2 Register Definition

<table>
<thead>
<tr>
<th>Address Offset</th>
<th>Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>spi_config</td>
</tr>
<tr>
<td>0x04</td>
<td>spi_command</td>
</tr>
<tr>
<td>0x08</td>
<td>spi_dnum</td>
</tr>
<tr>
<td>0x0c</td>
<td>spi_phase</td>
</tr>
<tr>
<td>0x10</td>
<td>spi_clkdiv</td>
</tr>
</tbody>
</table>

Table 4.28: SPI Master register map

<table>
<thead>
<tr>
<th>Bit #</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-6</td>
<td>-</td>
<td>R</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>5-4</td>
<td>SPI_SIZE</td>
<td>RW</td>
<td>00</td>
<td>SPI Word Size:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>00-SPI is Disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>01-8bit Word</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10-16bit Word</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>11-32bit Word</td>
</tr>
<tr>
<td>3</td>
<td>SPI_RX</td>
<td>RW</td>
<td>0</td>
<td>SPI Master Reads</td>
</tr>
<tr>
<td>2</td>
<td>SPI_TX</td>
<td>RW</td>
<td>0</td>
<td>SPI Master Writes</td>
</tr>
<tr>
<td>1</td>
<td>SPI_RDY</td>
<td>R</td>
<td>1</td>
<td>SPI Ready</td>
</tr>
<tr>
<td>0</td>
<td>SPI_EN</td>
<td>RW</td>
<td>1</td>
<td>SPI Enable</td>
</tr>
</tbody>
</table>

Table 4.29: spi_config definition

<table>
<thead>
<tr>
<th>Bit #</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-0</td>
<td>SPI_CMD</td>
<td>RW</td>
<td>0x00000000</td>
<td>SPI Command</td>
</tr>
</tbody>
</table>

Table 4.30: spi_command definition

<table>
<thead>
<tr>
<th>Bit #</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-0</td>
<td>SPI_DNUM</td>
<td>RW</td>
<td>0x00000000</td>
<td>Number of SPI Data Frames</td>
</tr>
</tbody>
</table>

Table 4.31: spi_dnum definition
### spi_phase Definition

<table>
<thead>
<tr>
<th>Bit #</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-0</td>
<td>-</td>
<td>R</td>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>0</td>
<td>SPI_MODE_0_0</td>
<td>RW</td>
<td>1</td>
<td>Number of SPI Data Frames</td>
</tr>
</tbody>
</table>

Table 4.32: spi_phase definition

### spi_clkdiv Definition

<table>
<thead>
<tr>
<th>Bit #</th>
<th>Name</th>
<th>Access Type</th>
<th>Reset Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>31-0</td>
<td>SPI_CLKDIV</td>
<td>RW</td>
<td>0x00000000</td>
<td>SPI Clock Divider</td>
</tr>
</tbody>
</table>

Table 4.33: spi_clkdiv definition
4.12.3 I/O Definition

<table>
<thead>
<tr>
<th>Name</th>
<th>I/O</th>
<th>Description</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCLK</td>
<td>I</td>
<td>APB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PRESETn</td>
<td>I</td>
<td>APB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PSEL</td>
<td>I</td>
<td>APB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PWRITE</td>
<td>I</td>
<td>APB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PADDR[7..0]</td>
<td>I</td>
<td>APB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PWDATA[31..0]</td>
<td>I</td>
<td>APB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>PENABLE</td>
<td>I</td>
<td>APB Slave Signal</td>
<td></td>
</tr>
<tr>
<td>SPI_SCLK</td>
<td>O</td>
<td>SPI Serial Clock Signal</td>
<td></td>
</tr>
<tr>
<td>SPI_CS_L</td>
<td>O</td>
<td>SPI Chip Select Signal</td>
<td>Active Low</td>
</tr>
<tr>
<td>SPI_MOSI</td>
<td>O</td>
<td>SPI Master Output Slave Input Signal</td>
<td></td>
</tr>
<tr>
<td>SPI_MISO</td>
<td>O</td>
<td>SPI Master Input Slave Output Signal</td>
<td></td>
</tr>
<tr>
<td>SPI_IRQ</td>
<td>O</td>
<td>SPI Interrupt</td>
<td></td>
</tr>
</tbody>
</table>

Table 4.34: SPI Master interface description

4.13 Package

A custom ceramic package which is developed for the CREAM project is preferred as the target package for the AS3222. In addition, it supports the high temperature operation. For testing purposes, another packaging option is chip-on-board (COB) which contains a custom header designed to increase the device temperature to a high value during the evaluation phase. Both of these packaging are handled by EPFL-LPM laboratory which is involved in this European project.

4.14 Circuit Area and Floorplan

The estimated circuit area is around 25 mm² and the target shape is 5 by 5 mm square. For the early specification phase, floorplan was not defined but mainly, it depends on the package bonding requirements. During the project design phase, the device area and floorplan were fixed.
Chapter 5

Memory Blocks

At high temperatures, memory elements (bistable elements) are found to be the critical components and they are expected to fail before the combinational (regenerative) logic gates. Specifically, SRAM memory cells are expected to be more prone to the failure in comparison with flip-flops (FFs) and latches. On the other hand, a 32-Kbyte (K), at least, cache memory is required by the processor core. The processor core is soft-core and FF based, meaning the FFs are used as the sequential elements. Hence, the alternative memory blocks to IBM MM should be investigated in case the cache memory consisting of IBM MM fails much before the processor core. Also note that the high temperature (200 - 225 °C) reliability of combinational and sequential logic gates, especially the FFs, in IBM standard-cell library still needs to be verified as they are characterized up to 125 °C. Nevertheless, they are expected to be more reliable than IBM MM at high temperatures.

In the following three different memory implementations: IBM MM, FF and latch based SCMs will be described in detail. Given their larger size and long access time in comparison with IBM MM, trade-off analysis for SCMs in between area, access/setup time and memory size is essential for the efficient design of such memories, and will also be demonstrated.

5.1 Technology compiled IBM MM

IBM offers various compilable memory array configurations (SRAM1SF) ranging from 64 to 16K or from 8 to 128, where the first range corresponds to number of words and the latter corresponds to data width in bits. The key features of these compilable MMs
are:

- Full static array
- Configurations up to 16K or 128 bits supported
- One read and write port
- Multiple decode options for area optimization
- Latched output data until next read cycle
- Bit write control for data masking
- Single clock edge operation through the use of self-timed restore

Although the processor core requires a 32-Kbyte (at least) MM, a smaller memory array (8K x 8) will be implemented in order to evaluate high temperature performance of IBM compilable memory array. On the other hand, the high temperature behavior of different array sizes may vary. For instance, different failure issues may arise in larger array sizes than that of smaller MMs. Nevertheless, the data extracted from the high temperature test of 8K MM will still provide valuable information regarding the behavior of compilable memory arrays due to regularity of memory array organization. In the following, an IBM MM with 8K x 8 array configuration will be described. Specifically, the memory cell, array structure and the array functional operation will be presented. Due to limited information from IBM, some other peripheral circuitry specifics such as the sense amplifiers and precharge circuitry will not be included.

5.1.1 SRAM Memory Cell and Array

A basic SRAM cell is illustrated in Figure 5.1. Note the consistency of transistor sizes with constraints (3.5 and 3.6). The transistors named with _lc suffix are used in cross-coupled inverters. This indicates that body terminal of these type transistors are shorted to source terminal in order to avoid floating body effects as described in Chapter 2.2.3 and 2.2.4. In addition, the other transistor alternative with _bc naming convention to indicate the possibility of connecting body terminal explicitly to any other potential than the source terminal is also not preferred. This is because of the increased area for these type of transistors. On the other hand, access transistors have their body floating (without any suffix in their names). Since floating body effects are more critical for the transistors in cross-coupled inverters compared to the access
transistors, this method is adopted in order to decrease the cell area. The basic memory cell operation is described in Chapter 3.2, hence will not be included in this chapter.

The layout of an SRAM cell is shown in Figure 5.2. The cell has the dimensions of 3.16 \( \mu m \times 6.3 \mu m \) as the width and length, respectively. Also note the area overhead for body connection in \( lc \) transistors. The complete layout of 8K x 8 IBM MM can be found in Appendix B.1. The array has the dimensions of 1746.85 \( \mu m \times 1009.7 \mu m \) as the length and width, respectively.
5.1.2 Array Functional Operation

A symbolic representation of IBM MM is shown in Figure 5.3. The description of these pins are provided in Table 5.1.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A00-A12</td>
<td>The address input pins define the address from or to which data will be read or written. Pins are used starting at the A00 least significant bit and counting upwards</td>
</tr>
<tr>
<td>DIN0-DIN7</td>
<td>Data input pins are non-inverting and they are used starting at the DIN0 least significant bit and counting upwards</td>
</tr>
<tr>
<td>BW0-BW7</td>
<td>The bit write control input pins are active high, and one pin is required for each data input bit. The bit write control pins allow masking of the input data. If the pin is held high, the corresponding data input bit is written into the array, If not, the array retains its previous contents for that bit. Pins are used starting at the BW0 least significant bit and counting upwards</td>
</tr>
<tr>
<td>CCLK</td>
<td>The clock pin initiates a read or write access of the SRAM on its falling edge during functional mode operations</td>
</tr>
<tr>
<td>RDWRT</td>
<td>The read/write control pin causes a read of the array to be performed when held high or a write to be performed when held low when the clock is strobed active</td>
</tr>
<tr>
<td>DOUT0-DOUT7</td>
<td>The data output pins are non-inverting. Pins are used starting at the DOUT0 least significant bit and counting upwards. The results of a read of the array are latched in the output, therefore, the results of a read remain on the data output pins until the next read. Write operations do not affect the data output pins</td>
</tr>
</tbody>
</table>

Table 5.1: IBM MM - 8K x 8 array pin description
In a read operation, RDWRT must be held high or brought high before the CCLK edge. Array clocked read operation is illustrated in Figure 5.4. The falling edge of CCLK initiates the access cycle. The data read from the corresponding address in the array appears at the output pins after the access time has elapsed. The output data is valid until the next read cycle. The data input pins and bit write control bits must meet their setup and hold times relative to the CCLK edge, even though they perform no logical function. The corresponding setup and hold times are listed in Table 5.2. Note that the nominal supply voltage is 2.5 V and the reported timings are the worst case values where the supply voltage drops to 2.25 V. They are quoted for 125 °C.
Table 5.2: Worst case SRAM1SF timing at 2.25 V [49]

Array Clocked Write Mode

In a write operation, RDWRT must held low or brought low before the CCLK edge. Array clocked write operation is illustrated in Figure 5.5. The falling edge of CCLK initiates the write cycle. Data output pins do not change while the data is being written into the corresponding address in the array. The data input pins and bit write control bits must meet their setup and hold times relative to the CCLK edge. Refer to Table 5.2 for the setup and hold times.

Figure 5.5: IBM MM array clocked write operation [49]
5.2 Standard Cell Based Memories

Unlike MMs which need to be created again for each new technology by a dedicated memory compiler, the use of SCMs described in a hardware description language (HDL) eases the portability of a design for any other technology [50, 51]. In addition, SCMs can be described in a generic way, which renders it easy to modify the number of words (R) or the number of bits per word (C) at design time. Once described in a HDL, SCMs can be placed automatically using the standard placement tool, whereas MMs need to be placed manually.

Despite these advantages, an SCM cell consisting of a FF or a latch is much bigger than the one of MMs (SRAM cell). However, more peripheral circuitry such as precharge circuitry and sense amplifiers is required by MMs than SCMs. For MMs with small storage capacity, the area overhead due to peripheral circuitry can be significant. Therefore, SCMs can outperform MMs in terms of area for small storage capacities, but become much bigger for large storage capacities. This was demonstrated in [50] for various bulk CMOS technologies ranging from 180 nm to 90 nm. In this chapter, both area and performance analysis will be presented for IBM 180 nm SOI technology. This study is essential in order to be able to implement a standard cell based cache memory for the processor core in case IBM MM fails first at high temperature.

Any SCM consists of the following building blocks: \textit{write logic, read logic} and \textit{array of storage cells}. In order to have the same functional behavior as in IBM MM, word address scheme with bit masking, falling edge clock operation and the same read/write modes are assumed.

\textbf{Write Logic}

The write logic needs to first select one out of R words, then the specific bit of this word, according to the given address and bit write control signal and update the content of the corresponding SCM cell on the next falling clock edge. This is accomplished by the \textit{write address decoder} which produces one-hot encoded row select signals and selects one row of the FF array [50, 51]. Logic-AND operation incorporates the bit write control signal for each cell in a word. Next, the selected cells need to update their state according to the data to be written.

In FF SCMs, one possibility consists in using FFs with an enable feature or a with a corresponding logic; all FFs in one row are enabled by the same row select and the corresponding bit write control signal. Another possibility is using basic FFs, i.e., FFs without enable feature, in conjunction with clock gates. In this case, a separate
clock signal is generated for each row and the currently selected row receives a clock pulse [50]. For each FF in the selected row, this pulse is enabled or disabled according to the corresponding bit write control signal. Hence, the selected FFs samples the provided data, while all others receive a silenced clock, thereby keeping their previous data. On the other hand, the second approach requires specific clock-gating standard cell in order to prevent functional hazards which may occur as a result of glitches. In other words, any glitch activity in clock gate enable signal, the combination of row select and bit write control signal in this case, is passed on to the gated clock unless a clock-gating cell is used, thereby jeopardizing the correct functioning of the entire system [51]. In latch based SCMs, a simple clocked latch is used, which also necessitates the usage of clock-gating cells [50].

A preliminary synthesis for various R values (from 8 to 8K) is done in order to have a first order estimation of the area dependence of the write address decoder on the number of rows in SCM. AND gates required for masking the specific bits are excluded in the synthesis results. Note that these results may vary for the actual SCM design when each block is synthesized together. In this case, the setup times and the input capacitances of storage cells also become involved and this may significantly change the area of the synthesized decoder. Nevertheless, this study gives a rough idea about the limits of the write address decoder.

![Write Address Decoder: Area vs # of Rows](image)

Figure 5.6: Area of the write address decoder versus the number of words

From Figure 5.6, a linear dependence is observed between the area of the one-hot write address decoder and the number of rows. Beyond 2K, area becomes much larger. Delay, on the other hand, increases logarithmically with respect to number of rows (Figure 5.7). Beyond 2K it grows large in size similar to area. These results suggest that write address decoder is expected to be a major obstacle in terms of area and delay for the design of SCMs with number of rows beyond 2K. Hence, the number of rows is limited to 2K in SCM study.
Read Logic

The read logic can be purely combinational or contain sequential elements, which leads to a read latency [50]. Given a word address scheme, one out of R words must be routed to the data output, according to the read address. The typical one clock cycle latency is obtained by inserting flip-flops either at the read address or the data output. The task of routing one out of R words to the output can be done using either tri-state buffers or multiplexers (MUXs) [50]. First approach asks for RxC tri-state buffers, i.e. exactly one per storage cell. R tri-state buffers are connected to one bit line, which has a large lumped capacitance if R is large. Stronger buffers are required for higher R values, which further increases the lumped capacitance and thus requires even stronger buffers. Hence, this approach is expected to result in a large overall area. Furthermore, it is difficult to buffer tri-state buses, which might be necessary to maintain reasonable slew rates if these buses are routed over long distances, especially for high R values [50]. Also note that DC paths from $V_{DD}$ to GND can arise, if two or more row select signals accidentally overlap.

In MUX based read logic, C parallel R-to-1 MUXs are required to route the selected word to the output. R-to-1 MUX itself can be implemented in many ways [50]. Most MUX architectures such as binary selection tree, do not require one-hot encoded row select signals. Some other MUX architectures, on the other hand, accept one-hot encoded row select signals and perform a logic-AND operation between each row select signal and the corresponding data bit, and finally performs a logic-OR operation on all AND-gate outputs [50].
A preliminary synthesis is done in order to compare the performances of these two MUX architectures in terms of area and access time for various R values (from 8 to 8K). Note that these results may vary for the actual SCM design when each block is synthesized together. Nevertheless, this study gives a rough idea about the limits of the read logic.

---

Figure 5.8: Binary Tree MUX and One-Hot Encoded MUX: area versus number of rows

Figure 5.9: Binary Tree MUX and One-Hot Encoded MUX: delay versus number of rows
A linear dependence is observed between the area and the number of rows for both MUX types (Figure 5.8). Given the larger slope beyond 2K, area becomes much larger. Delay increases logarithmically with respect to number of rows (Figure 5.9). On the other hand, One-Hot Encoded MUX outperforms Binary Tree MUX as the number of rows increases; it has much lower area and delay. Therefore, One-Hot Encoded MUX based read logic will be implemented in SCM study.

Array of Storage Cells

Both FFs and latches can be used as storage cells. The previous discussions on the write and read logic remain valid for both cell types. However, the difference of setup and hold time requirements for read and write operation between these two cells should be considered. Furthermore, FFs are clearly larger than latches. Hence, the storage cell area in FF based SCMs is expected to be much larger than that of latch based SCMs [50]. However, the total SCM area difference is less than the expected due to some area overhead associated with the implementation of latch based SCMs especially for the clock tree synthesis (CTS).

5.2.1 FF based SCM

Based on the previous discussion, a generic FF based SCM design is illustrated in Figure 5.10. Since no clock-gating cell exists in the target standard cell library, clock gates are not implemented. Instead, one-hot encoded row select signals produced by the write address decoder are connected to enable input of the corresponding flip-flop. The bit write functionality is realized by a logic-AND operation between the corresponding row select (DA, stands for decoded address) and bit write (BW) signal. Flip-Flops are inserted at the read address input in order to align data access to active clock edge.
CHAPTER 5. MEMORY BLOCKS

Figure 5.10: A generic FF based SCM
Another important issue is the selection of the storage cell. The target library offers only D type rising-edge triggered flip-flop (DFF) with various options, which consist of a simple DFF, a DFF with asynchronous set, reset or both functions. However, DFFs with enable feature are not provided. Hence, scannable DFFs (SDFF) are used as the storage cells. These cells consist of a 2-1 MUX and a simple DFF. They sample the scan-in data (SI) when the scan enable (SE) is high, otherwise they operate as a simple DFF, sampling the input data in the rising-edge. In order to mimic the enable feature for storage cells, data output (Q) pin is connected to data input pin (DI) and SI is connected to the corresponding data input bit (DIN) of SCM. In this way, the cell retains the stored value if the enable signal is low, if not, a new value is written into the cell. Also note that each SDFF cell clock needs to be inverted in order to align SCM operation to the falling clock edge as in IBM MM.

Clearly, an SDFF cell is larger than a simple DFF cell (28 - 35 %, depending on the driving strength). However, the area and speed overhead for the realization of enable feature with simple DFFs are more than that of SDFFs. This is because a 2-to-1 MUX is still required in order to be able to select either the previously stored value or the new data to be written. The exact same configuration is already implemented in SDFFs in a more compact way. This yields a smaller area, as well as a faster cell, compared to other approach. Thus, the area overhead of SDFF is less than expected. Nevertheless, missing DFF cells with enable feature is the main reason for the area overhead.

There are four different driving strengths defined for the SDFF cell: _A, _E, _H, and _K, where _A denotes the lowest and _K denotes the highest driving strength. _A, _E and _H have the same cell dimensions of 32.48 µm x 10.08 µm (327.4 µm²), and _K has the dimensions of 35.84 µm x 10.08 µm (361.26 µm²) as the width and length, respectively. SDFF cell layouts for different driving strengths are shown in Figure 5.11.

Among these cells, _A has the minimum transistor dimensions for both NMOS and PMOS transistors. For other driving strengths, the transistor sizes are the same and larger than that of _A. As shown in Figure 5.11, the only difference is the output stage, which is the strongest for _K, i.e. it can drive the largest output capacitance. Based on the discussions in Chapter 2 and 3, the largest leakage current at the output stage is expected for _K. Regarding these facts, _H appears to be a good compromise between area, driving strength and leakage current. Thus, it will be implemented as the storage cell.

Next, FF based SCMs having SDFF _H with enable feature as the storage cells, and using MUXs with one-hot encoded row select signals for the read logic are implemented in Verilog HDL, synthesized using Synopsys Design Compiler, placed, and routed (PAR) using Cadence Silicon Encounter for different memory sizes in IBM 180 nm SOI technology (cf. Table 5.3, Figure 5.12-5.13). In these memories, 8-bit word
size and various number of rows ranging from 8 to 2K are implemented. The reported timings are the minimum achievable access and setup times with the smallest area after PAR. The functionality of each SCM is verified by the post-layout simulations, in which random data is written into random addresses, while data is read from random addresses, for 1’000’000 times. The detailed information about register transfer level (RTL) implementations and the resulting SCM layouts are provided in Appendix C.1 and B.3, respectively.
### Table 5.3: FF based SCMs: area, access and setup time for different memory sizes

<table>
<thead>
<tr>
<th>R</th>
<th>Area ($mm^2$)</th>
<th>Access Time (ns)</th>
<th>Setup Time (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0.046</td>
<td>1.87</td>
<td>1.15</td>
</tr>
<tr>
<td>16</td>
<td>0.084</td>
<td>2.31</td>
<td>1.36</td>
</tr>
<tr>
<td>32</td>
<td>0.160</td>
<td>2.80</td>
<td>1.87</td>
</tr>
<tr>
<td>64</td>
<td>0.308</td>
<td>3.24</td>
<td>2.20</td>
</tr>
<tr>
<td>128</td>
<td>0.583</td>
<td>3.74</td>
<td>2.33</td>
</tr>
<tr>
<td>256</td>
<td>1.147</td>
<td>4.39</td>
<td>2.61</td>
</tr>
<tr>
<td>512</td>
<td>2.274</td>
<td>4.83</td>
<td>2.98</td>
</tr>
<tr>
<td>1024</td>
<td>4.626</td>
<td>5.32</td>
<td>3.39</td>
</tr>
</tbody>
</table>

Figure 5.12: FF based SCM: memory size versus area

Figure 5.13: FF based SCM: memory size versus access and setup times
As shown in Figure 5.12, area increases linearly with respect to memory size. On the other hand, both access and setup times increase logarithmically (Figure 5.13). For larger sizes (beyond 1K), the rate of change increases. This is more pronounced for access time. The main reason is that all inputs of FF based SCMs can be driven by buffers; highly capacitive nets are buffered inside. For example, read/write control (RDWRT), data inputs (DIN) and BW signals drive the corresponding inputs of each cell, thus, more buffers with larger driving strengths are needed as the size increases. Beyond 2K, FF based SCMs can not be placed and routed effectively due to excessive amount of buffers required to drive highly capacitive nets. Hence, larger FF based SCMs should be designed as the combination of smaller and effectively allocated memory modules.

5.2.2 Latch based SCM

A generic latch based SCM design is illustrated in Figure 5.14. Since no specific clock-gating cell exists in the target standard cell library, a two input NOR gate is used for this purpose. Specifically, one-hot encoded row select signals produced by the write address decoder controls the clock pulse provided to storage cells. The bit write functionality is realized by a logic-NAND operation between the corresponding DA and BW signal. In this way, only the selected cells receive a clock pulse, while all others receive a silenced clock. For example, if a current row is selected and BW of the corresponding cell is high, a clock pulse is provided, otherwise it is masked. Also note that the latches at the address inputs are avoided in order not to introduce strict timing requirements on the read ports.
CHAPTER 5. MEMORY BLOCKS

Figure 5.14: A generic Latch based SCM
Another important issue is the selection of the storage cell. The target library offers only SR type transparent-high latch (LATSR) with asynchronous set and reset functions. Hence, unused set and reset inputs are connected to low and high, respectively. This causes an area overhead compared to a simple D-latch with only enable (clock) and data inputs. There are three different driving strengths defined for the LATSR cell: _E, _H, and _K, where _E denotes the lowest and _K denotes the highest driving strength. _E and _H have the same cell dimensions of 21.84 µm x 10.08 µm (220.148 µm²), and _K has the dimensions of 25.76 µm x 10.08 µm (259.66 µm²) as the width and length, respectively. LATSR cell layouts for different driving strengths are shown in Figure 5.15. Based on the discussions in Chapter 2 and 3, the largest leakage current at the output stage is expected for _K. Regarding these facts, _H appears to be a good compromise between area, driving strength and leakage current. Thus, it will be implemented as the storage cell.

It should be also noted that missing clock-gating cells may cause functional hazards. Specifically, any glitch activity in the address decoder may create a small timing window with respect to the clock signal in which wrong data would be sampled. In order to avoid a possible circuit failure, two non-overlapping clock signals are introduced (Figure 5.16). In this clocking scheme, input signals are aligned to the first clock (CLK1), while the storage cells receives the second clock (CLK2).

Next, latch based SCMs having LATSR_H as the storage cells, and using MUXs with one-hot encoded row select signals for the read logic are implemented in Verilog.
HDL, synthesized using Synopsys Design Compiler, placed, and routed using Cadence Silicon Encounter for different memory sizes in IBM 180 nm SOI technology (cf. Table 5.4, Figure 5.17-5.18). In these memories, 8-bit word size and various number of rows ranging from 8 to 512 are implemented. The reported timings are the minimum achievable access and setup times with the smallest area after PAR. The functionality of each SCM is verified by the post-layout simulations, in which random data is written into random addresses, while data is read from random addresses, for 1'000'000 times. The detailed information about the RTL implementations and the resulting SCM layouts are provided in Appendix C.2 and B.3, respectively.

<table>
<thead>
<tr>
<th>R</th>
<th>Area (mm²)</th>
<th>Access Time (ns)</th>
<th>Setup Time (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0.046</td>
<td>1</td>
<td>1.10</td>
</tr>
<tr>
<td>16</td>
<td>0.075</td>
<td>1.39</td>
<td>1.38</td>
</tr>
<tr>
<td>32</td>
<td>0.134</td>
<td>1.82</td>
<td>1.67</td>
</tr>
<tr>
<td>64</td>
<td>0.258</td>
<td>2.26</td>
<td>1.95</td>
</tr>
<tr>
<td>128</td>
<td>0.503</td>
<td>2.70</td>
<td>2.18</td>
</tr>
<tr>
<td>256</td>
<td>0.956</td>
<td>3.43</td>
<td>2.51</td>
</tr>
<tr>
<td>512</td>
<td>1.898</td>
<td>4.01</td>
<td>3.32</td>
</tr>
</tbody>
</table>

Table 5.4: Latch based SCMs: area, access and setup time for different memory sizes

Figure 5.16: Two non-overlapping clock scheme

Figure 5.17: Latch based SCM: memory size versus area
Figure 5.18: Latch based SCM: memory size versus access and setup times

As shown in Figure 5.17, area increases linearly with respect to memory size. On the other hand, both access and setup time increases logarithmically (Figure 5.18). In larger array sizes, the rate of change increases especially for setup time. This is mainly because all inputs of latch based SCMs can be driven by buffers; highly capacitive nets are buffered inside. For example, RDWRT, DIN and BW signals drive the corresponding inputs of each cell, thus, more buffers with larger driving strengths are needed as size increases. Furthermore, unlike FF based SCMs, CTS for latch based SCMs are problematic. This stems from the gated clock implementation. During CTS, PAR tool tries to balance the skew between each storage element. In the absence of clock-gating cells, this task introduces excessive amount of clock insertion delay. Beyond a certain point, it becomes impossible to have a reasonable insertion delay compared to clock period. In such a case, the area overhead for CTS is also large and the insertion of additional buffers to meet the timing constraints fails. On the other hand, looser timing and area constraints would solve this problem. Nevertheless, it is not feasible to place and route latch based SCMs with memory sizes larger than 512-Byte.

5.3 Summary and Discussion

In this chapter, three different memory implementations: IBM MM, FF and latch based SCMs have been described in detail. In addition, the trade-off analysis for SCMs in between the area, access and setup times, and memory size have been presented. This study reveals that beyond 2K memory size (in number of rows), SCMs can not be placed and routed effectively due to excessive amount of buffers required to drive highly capacitive nets. On the other hand, the size threshold is much lower for latch
based SCMs due to area overhead during CTS. Latch based SCMs with memory sizes larger than 512-Byte are found to be infeasible to place and route in terms of both area and access/setup times. The expected area overhead for FF based SCMs in comparison with latch based SCMs is also much lower.

Another important remark is that given much higher area and access/setup times for such memories compared to IBM MM, larger SCMs should be designed as the combination of smaller and effectively allocated memory modules. For testing purposes, an FF and a latch based SCMs both with 512 Bytes memory size will be implemented.
Chapter 6

Test Structures

This chapter of the thesis covers testing structures to be implemented. First, various memory testing algorithms along with the memory fault models are briefly explained. The testing algorithm that will be implemented in BIST block is determined based on this discussion. Then, memory BIST implementation is described in detail. In addition, other test structures are implemented in order to assess the performance of IBM 180nm SOI technology at high temperatures. These test structures are: single transistors, basic resistors and diodes, a memory cell extracted from the technology compiled IBM SRAM, various standard cells with different driving strengths, and a basic frequency divider. All of these structures will be tested separately and their performances will be characterized at 225 °C.

6.1 Memory Testing

Memories are regular structures typically consist of a 2-dimensional array of identical, densely packed cells surrounded by dedicated I/O blocks of circuitry that provide access to the array, as illustrated in Figure 6.1.

Memory testing algorithms take the advantage of the regularity of the array for efficient testing time and cost, as well as achieving maximum fault coverage [52]. In many cases, the close packing of the cells in the array leads to faults that are not typically of concern in general sequential logic. Hence, fault modeling and test algorithms are different, and more complex than for general sequential logic [52].
6.1.1 Memory Fault Models

There are two types of fault models associated with memories: *classical* and *non-classical* faults [52]. Classical faults models include *cell array faults*, *addressing faults*, *dataline faults*, and *read/write faults*. Cell array faults result when the memory cell contents are *stuck-at-0* (sa0) or *stuck-at-1* (sa1). In other words, regardless of the values written into memory cells, their logic values will always be 0 and 1 for sa0 and sa1, respectively. Both logic-0 and logic-1 must be read from each cell in order to detect these faults. Addressing faults arise when there is a fault in the address decoder logic and a wrong address is selected. One must test all addresses with unique data to detect these faults. Data line faults result when *bit* and *bar* lines, where *bar* is the complement of *bit*, have faults which prevent correct data from being written into or read from the cells in the array. Both logic-0 and logic-1 values should pass through every data input to detect these faults. Read/write faults are originated from stuck-at faults on the read/write control lines or faults in the read/write control logic that prevent the proper read or write operations in the array.

On the other hand, non-classical faults are more specific to memories and are a result of dense packing of the cells in the array [53]. These faults can be categorized as *transition faults*, *data-retention faults*, *destructive read faults*, *pattern sensitivity faults* and *coupling faults*. Transition faults occur when the cells fail to make a $0 \rightarrow 1$ or $1 \rightarrow 0$ transition. In order to detect these faults, each cell must be tested for both transitions. Data retention faults result when the cells fail to maintain their contents after a certain period of time. To detect these faults, data must be read from each cell after a period of no activity, which means no read or write operation is in progress.
In destructive read faults, read operations change the contents of the cells. Hence, one must perform multiple reads of the cells for both logic-0 and logic-1 values after data has been written. Pattern sensitivity faults result when the content of a given cell is affected by the contents of other cells. For instance, a bridging fault between cells causes such faults. In order to detect these faults, one must surround each cell with the specific logic values in adjacent cells and check if its content is changing or not. Similarly, in coupling faults, the contents of a given cell is affected by operations on other cells. However, coupling faults are the result of transitions in other cells due to read or write operations while pattern sensitivity faults are a function of the static contents of neighboring cells [54]. Detection of these faults depend on the type of coupling.

The logic values to detect pattern sensitive faults in a given cell depend on the physical layout of the cell array. For example, a checker board in the cell will put the opposite logic values in directly adjacent cells as illustrated in Figure 6.2(a).

![Figure 6.2: Logic values to detect pattern sensitive faults in a given cell](image)

However, diagonally adjacent cells will still have the same logic values. Thus, different patterns with respect to the suspected adjacency pattern sensitivity should be applied in order to consider the nature of the possible pattern sensitivity [52]. This, on the other hand, is not a straightforward task because of the fact that in large memories the bits of multiple words are often interleaved [53]. For example, \( i^{th} \) bit from multiple words may be grouped together as opposed to all bits associated with a single word being grouped together. Moreover, a memory cell contains both \( \text{bit} \) and \( \bar{\text{bit}} \) subcircuit. Hence, the pattern sensitivity may be with respect to any of them. In some cases, the physical layout of the cell array may be such that the orientation of a cell is mirror-imaged with respect to adjacent cells in the array as illustrated in Figure 6.2(b). In these cases, the checker board pattern may hold for both \( \text{bit} \) and \( \bar{\text{bit}} \) portions of the cell. However, for the physical layouts in which the orientation of a cell is non-mirror imaged (Figure 6.2(c)), a different pattern may be required to obtain checker board pattern at \( \text{bit} \) and \( \bar{\text{bit}} \) subcircuits.
CHAPTER 6. TEST STRUCTURES

6.1.2 Memory Test Algorithms

Memory testing algorithms can be divided into two groups: traditional tests and March-based tests. Traditional test algorithms are either simple, fast but have a poor coverage, such as zero-one, checker board; or have better fault coverage but very complex and slow, such as Walking, GALPAT [52, 53]. On the other hand, March-based tests are a good compromise between the complexity and the fault coverage [55]. Hence, they are the dominant algorithms used for memory testing. In the following March-based tests will briefly be explained.

A March-based test algorithm is a finite sequence of March elements. A March element is a finite sequence of operations specified by an address order and a number of reads and writes. An operation can consist of: write-0 (w0), write-1 (w1), read-0 (r0), and read-1 (r1). The order of memory addresses is given as ↑, ↓, and ⇕ for ascending, descending and irrelevant order, respectively. All operations of a March element are applied to the current cell before proceeding to the next cell. Consider a March test, ⇕(w0) ↑(r0,w1) ↓(r1,w0) consisting of the March elements M₀: ⇕(w0), M₁: ↑(r0,w1) and M₂: ↓(r1,w0). M₀ implies writing logic-0 to all memory locations in any order. However, the ordering is explicitly stated for M₁ and M₂. M₁ implies; first reading and checking if the value read from the corresponding location is logic-0, then writing logic-1 into the same address. These operations must be performed in the ascending order for each memory cell. Similarly, M₂ means first reading logic-1 from an address and writing logic-0 into the same address in descending order. It is important to note that ascending and descending address orders are defined with respect to each other. In other words, the only requirement is that ↑ and ↓ be each other’s inverse. A monotonic increase and decrease are not strictly necessary for ↑ and ↓, respectively. However, from the implementation point of view of these algorithms, it is common to consider ↑ as a monotonic increase and ↓ as a monotonic decrease.

The most common March-based test algorithms and their fault detection capabilities in terms of the type of faults detected by the tests are provided in Table 6.1.2 and Table 6.1.2. March C- algorithm has been observed to provide the highest fault coverage with a reasonable complexity [52]. It is also possible to detect data retention faults by inserting a delay between each March sequence. There are other test algorithms that have a higher fault coverage compared to March C-, but with a higher test length [55].

The test algorithms presented here are intended for memories with one data bit per word. However, they can also be used to test memories with multiple bits per word. In some cases, this requires possible modification to the algorithm to increase fault detection capability especially for coupling and pattern sensitivity faults within the bits of the word of the memory [52]. For the cases in which the bits of different words are interleaved, the March algorithms can be used as shown, with all logic-
Algorithm | Algorithm Sequence | Test Length
---|---|---
*MATS* | $\uparrow (w0) \downarrow (r0,w1) \uparrow (r1)$ | $4N$
*MATS+* | $\uparrow (w0) \uparrow (r0,w1) \downarrow (r1,w0)$ | $5N$
*MATS++* | $\uparrow (w0) \uparrow (r0,w1) \downarrow (r1,w0,r0)$ | $6N$
*March X* | $\uparrow (w0) \uparrow (r0,w1) \downarrow (r1,w0) \uparrow (r0)$ | $6N$
*March Y* | $\uparrow (w0) \uparrow (r0,w1,r1) \downarrow (r1,w0,r0) \uparrow (r0)$ | $8N$
*March C-* | $\uparrow (w0) \uparrow (r0,w1) \uparrow (r1,w0) \downarrow (r0,w1) \downarrow (r1,w0) \uparrow (r0)$ | $10N$

Table 6.1: March-based test algorithms

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Stuck-at Faults</th>
<th>Address Faults</th>
<th>Transition Faults</th>
<th>Coupling Faults</th>
</tr>
</thead>
<tbody>
<tr>
<td>MATS*</td>
<td>yes</td>
<td>some</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>MATS+*</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>
*MATS++* | yes | yes | yes | no |
*March X* | yes | yes | yes | some |
*March Y* | yes | yes | yes | some |
*March C-* | yes | yes | yes | yes |

Table 6.2: Fault detection for March-based test algorithms [52]

0s and logic-1s for data patterns written into and read from the memory to detect coupling faults. On the other hand, if the bits of a given word are physically adjacent in the memory, some modifications need to be made to the algorithms to ensure the detection of coupling and pattern sensitivity faults [56]. The modifications consist of running the algorithm multiple times with a different data background sequence during each execution. A data background sequence is a pattern that can detect coupling and pattern sensitivity faults depending on the physical layout of the memory [52,56]. Therefore, the collection of all data background sequences should then provide testing independent of the physical layout of the memory at the expense of multiple executions of the algorithm with a different data background sequence used during each execution. Each data background sequence and its inverse pattern are used to replace all logic-0s in w0 and r0 operations and all logic-1s in the w1 and r1 operations, respectively.

### 6.1.3 Testing Memory Blocks

Memories are the most critical components for reliable operation of the overall system at high temperature. Since they are more susceptible to high temperature effects compared to digital logic blocks, they are more likely to fail; they dictate the overall system performance at elevated temperature. Hence, assessing their performance at high temperature is of great importance. On the other hand, large I/O pin count for
memories complicates the testing procedure of such blocks. In addition, implementing a certain memory test algorithm requires extensive read and write operations to all memory locations, which may translate into long testing times depending on the test frequency and the memory size.

In order to overcome these problems, BIST approach is adopted. The basic idea of BIST is to design a circuit so that the circuit (e.g. memory in this case) can test itself and determine whether it is fault-free or faulty [54]. This is typically achieved by incorporating additional circuitry and functionality into the design of the circuit to facilitate the self-testing feature (Figure 6.3). This additional functionality must be capable of generating test patterns and analyzing the test responses of the circuit under test. The output responses to a sequence of patterns generated on-chip will be compacted into some type of Pass/Fail indication.

![Figure 6.3: A typical BIST architecture](image)

The incorporation of BIST also requires some I/O pins for activating the test mode (BIST Start), reporting the test results (the Pass/Fail indication) and indicating that BIST sequence is complete and the results are valid. The coupling of BIST block and memory blocks enables the testing to be run at circuit speed as well as with a less number of I/O pins.

On the other hand, the incorporation of BIST increases the silicon area required to implement the chip. The additional hardware added to the circuit can also increase the delays of normal circuit paths, decreasing the speed at which the memory can be used during its normal operation. These two costs are commonly referred to as area overhead and performance penalty, respectively [53].
However, both of these costs are not important since our concern is to evaluate the high temperature behavior of these memories. After the failures of different memory blocks at high temperature are analyzed, performance and area will be of importance but not as much as the robustness; but in this case, BIST hardware will not be included in the high temperature memory design. Still, implementing a reliable memory block for high temperature operation remains the main concern, rather than a high performance block.

Furthermore, apart from the testing point of view, BIST hardware itself is of great importance. Since the whole system will not be integrated into the test chip, BIST hardware composed of sequential elements, flops, and some combinational logic will mimic the processor which is also a soft-core IP to be mapped to the same standard-cell library. There will be of course some differences between their temperature behaviors since the processor core is a much more complicated design with various tasks assigned apart from accessing the memories. Nevertheless, BIST hardware is a valuable part of the test chip to obtain some information regarding the processor behavior at high temperature.

A generic BIST hardware is implemented to test both the FF based SCM and IBM MM. On the other hand, the latch based SCM will be tested externally. This is because of the fact that BIST hardware is implemented with flops as the sequential elements. Thus, the temperature behavior of BIST block may differ from the latch based SCM which affect the test itself. Important remark is that due to their different complexities, the different cells with also different driving strengths and complex gates with multiple inputs (e.g. AOI221 that is specific and-or-invert gates or four input NAND gates (NAND4)) are most likely to be inferred in these blocks. This is perfectly natural from the semi-custom design perspective which is basically to utilize all available cells to generate a hardware with the maximum achievable performance in terms of speed and area. Therefore, high temperature performances of the latch based SCM and BIST modules imply mostly the performance of the sequential elements within the design which are basically, latches and flip-flops. With BIST failing before latch based SCM, for instance, it may not be possible to evaluate the high temperature performance of the latch based SCM properly. A latch based BIST design should then be designed for this purpose. However, this further complicates the design since the standard cell library provided by the technology does not include a clock-gating cell. The design may suffer from an intensive glitch problem which makes it difficult to meet the timing for the design as explained in the previous chapter. Again for the same reason, another IBM MM stand alone, directly connected to the I/O pads, is also implemented and will be tested externally. With half of the blocks are to be tested via BIST blocks, it is possible to allocate all I/O pads to the memory blocks.
BIST Design

BIST implementation consists of two modules: the scan-chain interface (SCI) and the core, as illustrated below. While SCI is connected to I/O pins and is responsible of the serial communication between the BIST hardware and the outside, the core coupled to the memory performs the actual testing. Let us first explain SCI in detail.

A serial communication between the BIST hardware and the outside is implemented in order to keep the number of required I/O pins minimum. This, on the other hand, introduces a loading latency depending on the number of user-accessible setup registers (UASR) to be programmed.
Four internal registers: \texttt{start\_addr}, \texttt{stop\_addr}, \texttt{test\_data} and \texttt{control} can be serially programmed via SCI. During the test, the memory region defined by \texttt{start\_addr} and \texttt{start\_addr} will be covered. Depending on the size (i.e. number of rows) of the memory to be tested, \texttt{start\_addr} and \texttt{stop\_addr} are defined as 9 and 13 bit-registers, respectively. The \texttt{test\_data} register corresponds to data background pattern used during the test. Two different test modes are defined by the \texttt{control} register: full-scan (FS) mode and stop-at-first-fault (SFF) mode. While the first continues testing regardless of the faults detected in the memory, the latter will abort the process in case of a failure. These registers along with their default values are illustrated in Table~6.3.

<table>
<thead>
<tr>
<th>Register</th>
<th>Length (bits)</th>
<th>Function</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>start_addr</td>
<td>9 or 13</td>
<td>start address</td>
<td>0</td>
</tr>
<tr>
<td>stop_addr</td>
<td>9 or 13</td>
<td>start address</td>
<td>511 or 8191</td>
</tr>
<tr>
<td>test_data</td>
<td>8</td>
<td>data background pattern</td>
<td>“00000000”</td>
</tr>
<tr>
<td>control</td>
<td>2</td>
<td>test mode</td>
<td>“10” (full scan - FS)</td>
</tr>
</tbody>
</table>

Table 6.3: User-Accessible Setup (Test Configuration) Registers

A serial communication protocol is implemented to program UASRs. The corresponding finite state machine (FSM) implementation is illustrated in Figure 6.5. SC\_IN input pin should be kept at low all the time except the configuration phase. A rising edge will initiate the process; a logic-1 should be applied for a single clock cycle. This will trigger a transition from \texttt{ST\_IDLE} to \texttt{ST\_CONF} state. In the next two cycles, the appropriate configuration bits should be applied. Two possible configurations to program UASRs are available (Table~6.4). In order to set one of these configurations, MSB and then LSB of the corresponding serial input pattern must be provided consecutively. No other input patterns are allowed.

<table>
<thead>
<tr>
<th>Value</th>
<th>Configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>“00”</td>
<td>programming UASRs disabled, use default values</td>
</tr>
<tr>
<td>“11”</td>
<td>programming UASRs enabled</td>
</tr>
</tbody>
</table>

Table 6.4: Configurations to program UASRs
Figure 6.5: SCI state diagram
After the configuration bits are issued properly, UASRs should be programmed one-by-one by providing the corresponding input stream. Any input sequence to program a register should be initiated with a logic-1 for one clock cycle. If not (e.g. logic-0 is issued in the beginning), the default value be will loaded into the corresponding register. This means to bypass the corresponding stream load state (Figure 6.5). In the following clock cycles, the stream bits should be provided starting from MSB due to the shift-register nature of UASRs. Input streams for UASRs should be provided in the following order; control, start_addr, stop_addr, and data_pattern as illustrated in Figure 6.6. $ST_{LOAD1-4}$ states are implemented for loading the registers in this order (Figure 6.5). The protocol should be terminated with a logic-1 input for one clock cycle. This condition is checked in $ST_{CHECK}$. Otherwise, the protocol is considered to be incomplete and should be started over.

![Figure 6.6: SCI shift register and data direction](image)

Once UASRs are programmed, SCI proceeds to $ST_{READY}$ state in which $B_{start\_in}$ signal required by the core to initiate the test will be unmasked. This means unless SCI is configured properly (e.g. $ST_{READY}$ is reached), asserted $B_{start\_in}$ signal will not be able to initiate the test. From this point on, SCI will be waiting a response from the core. Now let us switch the focus from SCI to the core. As explained before, the core is responsible of testing the memory. It basically applies stimuli, and analyzes the memory responses to verify the correct functionality. However, the test patterns are provided externally via SCI, unlike the case of the general BIST architecture in which the patterns are generated on-chip (Figure 6.3). Input isolation circuitry is also eliminated since the the memories are not connected to the system.

The core implements March C- algorithm consisting of the march elements $M_0$: $\uparrow(w0)$, $M_1 : \uparrow(r0,w1)$, $M_2 : \uparrow(r1,w0)$, $M_3 : \downarrow(r0,w1)$, $M_4 : \downarrow(r1,w0)$, and $M_5 : \downarrow(r0)$. Here, “0” and “1” denote test pattern and its complement, respectively. The state diagram for core FSM is illustrated in Figure 6.7.
Figure 6.7: Core state diagram
The assertion of \( B_{\text{start}} \) will set \( B_{\text{busy}} \) high and trigger a transition from \( ST_{\text{IDLE}} \) to \( ST_{\text{CONF}} \), where the contents of core registers are updated according to test configuration loaded into UASRs. Next, the memory contents are initialized in ascending order. The similarity of operations in the same address order enables implementing a common state for \( M_1 \) and \( M_2 \), \( ST_{M_2,M_3} \). The difference of the order of data patterns applied for read and write operations is compensated with a \( \text{data\_dir} \) flag. Depending on the march element \( (r_1,w_0) \) or \( (r_0,w_1) \), the value of this flag is set and the read and write data patterns are swapped. Similarly, \( ST_{M_4,M_5} \) applies both \( M_5 \) and \( M_4 \) by using the same flag. \( ST_{M_6} \) performs the final read operation. If, at any time, the read data differs from the expected data, a failure exists and FSM proceeds to \( ST_{\text{SC\_OUT}} \). In this state, the failure details such as the corresponding address, read data, and the march element in which the error occurs are scanned-out via SCI. Since different clock domains are used for SCI and the core (i.e. a slower clock for SCI), a handshake protocol is implemented in order to ensure the proper communication between them. This is illustrated in Figure 6.8.

![Handshake protocol between SCI and the core](image)

Figure 6.8: Illustration of handshake protocol between SCI and the core

Once \( ST_{\text{SC\_OUT}} \) is reached, the core FSM forwards the failure details, called \( \text{frame} \), to SCI and asserts \( SC_{\text{start}} \) signal high. After SCI has sampled this signal, it acknowledges the core by setting \( SC_{\text{done}} \) low. Then, the core samples \( SC_{\text{done}} \), de-asserts \( SC_{\text{start}} \) and it holds. In the next clock cycle, SCI FSM proceeds to \( ST_{\text{SC\_OUT}} \) state and serially scans the frame out. A rising edge at \( SC_{\text{OUT}} \), which is the serial output pin, indicates the start of a new scan-out process. Otherwise, it is kept at low. In the following clock cycles, the address bits are scanned out starting from the MSB. Then a logic-1 is asserted for one clock cycle, and the data bits are scanned out. Again a logic-1 is provided at the output for a single clock cycle, and the march element bits are scanned out. The output streams “000”, “001”, “010”, “011”, and “100” represent \( M_1 \), \( M_2 \), \( M_3 \), \( M_4 \), and \( M_5 \), respectively. The process is terminated with a logic-1 signal for one clock cycle. The frame elements, their default values and the number of clock cycles (NCC) required to scan-out each element are described in Table 6.5.
CHAPTER 6. TEST STRUCTURES

<table>
<thead>
<tr>
<th>Frame Element</th>
<th>Description</th>
<th>NCC</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>error_addr</td>
<td>the address of the failure</td>
<td>9 - 13</td>
<td>0 - 511 or 0 - 8191</td>
</tr>
<tr>
<td>error_data</td>
<td>the faulty data</td>
<td>8</td>
<td>“00000000” - “11111111”</td>
</tr>
<tr>
<td>error_marchST</td>
<td>the march element of the failure</td>
<td>3</td>
<td>“000” - “100”</td>
</tr>
</tbody>
</table>

Table 6.5: The serial output frame details

When the scan-out is completed, SCI sets $SC_{done}$ high back again and the core samples it. Next, the core sets $B_{failed}$ high and $B_{succeed}$ low to indicate the presence of a failure. If FS mode is enabled, the test continues where it left. Otherwise, it is terminated.

Another important point regarding the core operation is the two clock cycles latency due to the comparator block which analyzes the output responses. This is because both data read from the memory and comparator output are latched. Consider a failure during the exercise of $M_1$ element under FS mode set. The possible performance degradation is handled by adopting a pipelined operation as described in the following. Imagine there has been an error at the address location $A$, this will be manifested at the comparator output in the third clock cycle after $A$ is issued. Since the data is read from and written into the same address location, a single clock cycle latency is effectively introduced to the operation. By the time $ST_{SC\_OUT}$ is reached, the data is already read from $A+1$. However, during the scan-out process, the core holds. In the next clock cycle following up on the acknowledge from SCI, the comparator response is ready as well as the inverse data pattern is written into $A+1$. By this way, no extra latency is introduced and the test continues as if the core FSM does not hold.

On the other hand, for the final read operation in $ST_{M_5}$, the two clock cycle latency is effective. Consider the scenario in the following. If there exists an error at the address location $A$, it manifests itself in the third clock cycle. However, unlike $M_1$, $M_5$ proceeds with the next address, $A+1$, in the next clock cycle. Thus, the test result for $A+1$ is already out and the data is read from $A+2$ when $ST_{SC\_OUT}$ is reached. In the following clock cycle, If there has been an error at $A+1$, present state is retained, the result for $A+2$ becomes valid and data is read from $A+3$, concurrently. On the other hand, next state will be $ST_{M_5}$, if no error exists at $A+1$. Therefore, any additional latency is avoided.

The core acknowledges the completion of the process by de-asserting $B_{busy}$. If no errors are present in the memory, $B_{succeed}$ is set high, and $B_{failed}$ remains low. On the contrary, if the test is completed with error(s), they retain their previous values set when the first failure occurs.
The hardware is implemented in Verilog HDL (Appendix C.3) and synthesized into IBM 180 nm SOI standard-cell library by using Synopsys Design Compiler (DC) tool. The synthesized design is simulated in MODELSIM and its correct functionality is verified. The RTL and post-synthesis simulation results will not be provided here. Instead, the post-PAR simulations for the test chip can be found in the next chapter.

### 6.2 The Other Test Structures

The digital test structures implemented to assess the high temperature performance of memory blocks may be problematic. Specifically, all digital memory blocks including the corresponding BIST hardwares will be connected to digital I/O pads. These pads consist of strong amplifier with an I/O buffer in order to drive large capacitance off chip, as well as electrostatic discharge (ESD) protection circuitry. At high temperatures, the amplifier itself is expected to survive in comparison with the I/O buffer. This is because the excessive amount of leakage current will cause I/O buffer to fail. In other words, it will not be able to manipulate the output any more, i.e., its switching characteristics will be degraded. Given the fact that beyond some temperature, digital I/O drivers itself may fail, the test results may not indicate the actual failures of the memory blocks. In another extreme case, the degradation of voltage levels within a digital logic or memory element results in the circuit failure and the corresponding I/O pad may also mask this failure.

The observation of analog voltage levels at the intermediate nodes before the output buffers is of great significance. By this way, we can have a detailed understanding of the operation of digital logic blocks and memory elements at high temperatures. Hence, direct access to single devices need to be provided via bond pads. However, this may introduce additional issues that need to be considered. For example, the output capacitance of such devices will be significantly increased by the bond wires connected for measurement purposes. In such a case, they will not be able to drive high capacitances.

Based on these facts, we implement different structures including single transistors with various widths, basic resistors and diodes, a memory cell extracted from the technology compiled IBM SRAM, various standard cells (two and three input NAND gates, inverters, FFs, and latches) with different driving strengths, a basic frequency divider, a ring oscillator, and a basic linear amplifier which consist of a PMOS transistor (W=4.6 µm, L=320 nm). Also antenna cells (FGTIE) are connected to gate terminals of these test blocks in order to avoid possible gate breakdown due to the wiring of large pads to gates.
The output of these gates are connected to bond pads via large PMOS transistors (W=4.6 µm, L=320 nm). In this configuration, drain, gate and source terminals of these transistors are connected to GND, the outputs of the corresponding test blocks, and to bond pads for measurement purposes, respectively. It should be also noted that all three transistor alternatives in target technology (bc, lc, and floating body) are implemented also to assess floating body effects at high temperature. Furthermore, we also implement a super-flop that is a custom DFF cell with enable feature. Depending on its high temperature performance, super-flop can facilitate the design of smaller and reliable SCMs. All of these structures will be tested separately and their performances will be characterized at 225 °C. Such characterization will allow us to attain valuable information regarding high temperature operation of various blocks in SOI technology in order to tackle the design of reliable memory blocks at 225 °C. Consider, for instance, the most basic linear amplifier, a single NMOS transistor. At high temperatures, its device characteristics will be degraded. Specifically, it loses the linearity and the device experiences premature saturation. However, the linear amplifier still retains its fundamental function. Therefore, it can be exploited to figure out until what temperature IBM 180 nm SOI technology can be utilized.

The detailed explanation of these blocks are beyond the scope of this thesis and will not be provided. Nevertheless, the data to be extracted from their high temperature testing will exploited to design reliable memories at high temperatures.

6.3 Summary and Discussion

In this chapter, the essentials of memory testing have been briefly described. Different testing algorithms, as well as the BIST concept have been discussed. The implementation of a BIST hardware which employs March C- algorithm to perform high temperature testing of different memory blocks has been explained in detail.

On the other hand, various test structures including some standard cells, a basic linear amplifier, an IBM MM SRAM cell, a frequency divider, a ring oscillator, and a custom DFF cell with enable feature have been implemented in order to characterize the target technology at high temperatures. By this way, we aim to figure out until what temperature IBM 180 nm SOI technology can be utilized. Therefore, these test structures are essential for the design of reliable memory blocks, as well as other digital blocks comprising the processor at high temperatures.
Chapter 7

The Test Chip

This chapter of the thesis covers the test chip, prometheus, details \(^1\) as well as the high temperature testing methodology to be adopted. Post-PAR simulation results for the digital part are also presented.

7.1 The Test Chip

We have completed and sent the test chip tape-out. For the chip realization a silicon area of 25 \(mm^2\) is available with 116 I/O pins. Our goal is to place different memory types: an IBM MM, an FF and a latch based SCMs, and various analog and digital test structures as described in Chapter 6.

Two distinct regions are implemented on the chip. The first region, referred as the digital part, contains various memory implementations as well as on-chip test structures to perform BIST. The block diagram of digital part is illustrated in Figure 7.1; I/O pads are also included. The I/O cells can be configured as both input and output ports, the corresponding delays are \(\approx 0.8\text{ns}\) and \(\approx 2\text{ns}\), respectively. Also note that each memory block is completely independent from each other. Therefore, each block will be tested separately.

\(^1\)We have sent the tape-out on 14 March, so we can not include the measurements on the chip in this thesis. Nevertheless, we intend to continue the research in the future.
Figure 7.1: Test chip block diagram
We can identify four different blocks in Figure 7.1:

- **IBM MM + BIST block**: This unit consists of an IBM MM with 8K x 8 array configuration (64-Kbyte) connected to a BIST block to perform on-chip testing (refer to Chapter 6.1.3 for details). This part utilizes 10 I/O pins (6 input and 4 output pins), which are described in Table 7.1. Three $V_{DD}/GND$ ($E_4$, $E_3$ and $E_2$) pairs on the east-side of the chip are placed nearby in order to avoid higher IR drop. The minimum achievable clock period for this block is reported as 13 ns (with 40% duty cycle) by the PAR results.

- **IBM MM stand alone**: The same IBM MM with 8K x 8 array configuration (64-Kbyte) is implemented stand alone and directly connected to I/O pins of the chip. This block utilizes 31 I/O pins (23 input and 8 output pins), as described in Table 7.1. Due to the limited number of available I/O pins, the corresponding BW signals are hardwired to logic-1, enabling the write access for each bit at all times. Two $V_{DD}/GND$ ($W_0$, $W_1$) pairs on the west-side and one $V_{DD}/GND$ ($S_0$) pair on the south-side of the chip are placed nearby in order to avoid higher IR drop. This memory will be tested externally. The minimum achievable clock period for this block is reported as 8 ns by the PAR results. One clock cycle latency is assumed for both read and write operations, whereas IBM MM itself can be used without any clock latency due to the falling-edge operation. Since the access time is more than the setup time, the first dictates the clock period. The reported timing is more than that of IBM MM (7.2 ns). This is because of the I/O delays.

- **Latch based SCM stand alone**: A latch based SCM with 512 x 8 array (4-Kbyte) configuration is implemented and directly connected to I/O pins of the chip. It utilizes 27 I/O pins (19 input and 8 output pins), as described in Table 7.1. Due to the limited number of available I/O pins, the corresponding BW signals are hardwired to logic-1, enabling the write access for each bit at all times. Two $V_{DD}/GND$ ($W_0$, $W_1$) pairs on the west-side and two $V_{DD}/GND$ ($S_1$, $S_2$) pairs on the south-side of the chip are placed nearby in order to avoid higher IR drop. This memory will be tested externally. The access time for this block is indicated as 4.85 ns, which is more than that of the latch based SCM in Chapter 5.2.2. This because of the I/O delay. The setup time is reported as 3.21 ns.

- **FF based SCM + BIST block**: This unit consists of a FF based SCM with 512 x 8 array (4-Kbyte) configuration connected to a BIST block to perform on-chip testing (refer Chapter 6.1.3). This part utilizes 10 I/O pins (6 input and 4 output pins), as described in Table 7.1. Two $V_{DD}/GND$ ($W_2$, $W_3$) pairs on the west-side are placed nearby in order to avoid higher IR drop. The minimum
CHAPTER 7. THE TEST CHIP

achievable clock period for this block is reported as 9.95 ns (with 46 % duty cycle) by the PAR results.

In addition, PADTESTIN and PADTESTOUT inputs are used to test I/O pad itself at high temperature (Table 7.1). The output signal is the buffered version of PADTESTIN through a chain of inverters. Note that, three sides of the chip are allocated for I/O placement of digital part. A narrow region at full chip length on the north-side, on the other hand, is allocated for other test structures, referred as the analog part. These structures, which are described in Chapter 6.2, are implemented in order to assess the high temperature performance of the target technology. The corresponding input and output pins of these structures are directly connected to the bond-pads without any I/O cell. The test chip layout can be found in Appendix B.2.

Prometheus Digital-part pin description

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IAxSI_0 - IAxSI_12</td>
<td>The address input pins of the IBM MM stand alone.</td>
</tr>
<tr>
<td>IDINxDI_0 - IDINxDI_7</td>
<td>Non-inverting data input pins of the IBM MM stand alone.</td>
</tr>
<tr>
<td>ICLKxCI</td>
<td>The clock pin of the IBM MM SRAM stand alone.</td>
</tr>
<tr>
<td>IRDWRTxSI</td>
<td>The read/write control pin of the IBM MM stand alone. It should be high and low asserted for read and write operations, respectively.</td>
</tr>
<tr>
<td>IDOUTxDO_0 - IDOUTxDO_7</td>
<td>Non-inverting data output pins of the IBM MM stand alone.</td>
</tr>
<tr>
<td>LAxSI_0 - LAxSI_8</td>
<td>The address input pins of the latch based SCM.</td>
</tr>
<tr>
<td>LDINxDI_0 - LDINxDI_7</td>
<td>Non-inverting data input pins of the latch based SCM.</td>
</tr>
<tr>
<td>LBCLKxCI</td>
<td>The clock pin of the latch based SCM. Also note due to non-overlapping clocking scheme, this clock is received by the memory elements while the control signals need to be aligned with another clock (5.16).</td>
</tr>
<tr>
<td>L_RDWRTxSI</td>
<td>The read/write control pin of the latch based SCM. It should be high and low asserted for read and write operations, respectively.</td>
</tr>
<tr>
<td>L_DOUTxDO_0 - L_DOUTxDO_7</td>
<td>Non-inverting data output pins of the latch based SCM.</td>
</tr>
<tr>
<td>DBCLKxCI</td>
<td>The clock pin of the FF based SCM + BIST block. This clock is used during the test.</td>
</tr>
</tbody>
</table>
### CHAPTER 7. THE TEST CHIP

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBRESETnxSI</td>
<td>Active low reset signal for the FF based SCM + BIST block. This clock is used to initialize the BIST core registers.</td>
</tr>
<tr>
<td>DSCCLKxCI</td>
<td>The clock pin of the FF based SCM + BIST block. This clock is used for external communication, i.e., serial I/O.</td>
</tr>
<tr>
<td>DSCRESETnxSI</td>
<td>Active low reset signal for the FF based SCM + BIST block. This clock is used to initialize the SCI registers. This clock will be much slower than DBCLKxCI.</td>
</tr>
<tr>
<td>DSC_INxDI</td>
<td>Serial input pin of the FF based SCM + BIST block in order to program the internal registers.</td>
</tr>
<tr>
<td>DSTARTxSI</td>
<td>Start signal of the FF based SCM + BIST block. A rising edge at this pin starts BIST operation if SCI registers are properly configured.</td>
</tr>
<tr>
<td>DB_BUSYxSO</td>
<td>Test status flag the FF based SCM + BIST block. This signal is set high during the test.</td>
</tr>
<tr>
<td>DB_FAILEDxSO</td>
<td>Test result flag the FF based SCM + BIST block. This signal is set high whenever a failure is detected and once asserted, it remains high until a new BIST operation.</td>
</tr>
<tr>
<td>DB_SUCCESSxSO</td>
<td>Test result flag the FF based SCM + BIST block. This signal is set high whenever a test is completed without any failure.</td>
</tr>
<tr>
<td>DSC_OUTxDO</td>
<td>Serial output pin of the FF based SCM + BIST block. This pin is used to scan the current failure details out.</td>
</tr>
<tr>
<td>IBCLKxCI</td>
<td>The clock pin of the IBM MM + BIST block. This clock is used during the test.</td>
</tr>
<tr>
<td>IBRESETnxSI</td>
<td>Active low reset signal for the IBM MM + BIST block. This clock is used to initialize the BIST core registers.</td>
</tr>
<tr>
<td>ISCCLKxCI</td>
<td>The clock pin of the IBM MM + BIST block. This clock is used for external communication, i.e., serial I/O.</td>
</tr>
<tr>
<td>ISCRESETnxSI</td>
<td>Active low reset signal for the IBM MM + BIST block. This clock is used to initialize the SCI registers. This clock will be much slower than IBCLKxCI.</td>
</tr>
<tr>
<td>ISC_INxDI</td>
<td>Serial input pin of the IBM MM + BIST block in order to program the internal registers.</td>
</tr>
</tbody>
</table>
### Table 7.1: Prometheus Digital-part Pin description

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ISTARTxSI</td>
<td>Start signal of the IBM MM + BIST block. A rising edge at this pin starts BIST operation if SCI registers are properly configured.</td>
</tr>
<tr>
<td>IB_BUSYxSO</td>
<td>Test status flag the IBM MM + BIST block. This signal is set high during the test.</td>
</tr>
<tr>
<td>IB_FAILEDxSO</td>
<td>Test result flag the IBM MM + BIST block. This signal is set high whenever a failure is detected and once asserted, it remains high until a new BIST operation.</td>
</tr>
<tr>
<td>IB_SUCCESSxSO</td>
<td>Test result flag the IBM MM + BIST block. This signal is set high whenever a test is completed without any failure.</td>
</tr>
<tr>
<td>ISC_OUTxDO</td>
<td>Serial output pin of the IBM MM + BIST block. This pin is used to scan the current failure details out.</td>
</tr>
<tr>
<td>PADTESTIN</td>
<td>Input signal for testing input I/O pad.</td>
</tr>
<tr>
<td>PADTESTOUT</td>
<td>Output signal for testing output I/O. This signal is the buffered version of PADTESTIN through a chain of inverters.</td>
</tr>
<tr>
<td>VDD/GND</td>
<td>The core is powered by 12 power/ground pin pairs. They are named as _E, _W, and _S. (4 of such pairs located at the west side, 3 located at the south side, and 5 located at the east side of the chip)</td>
</tr>
<tr>
<td>VDD/GND _I/O</td>
<td>6 power/ground pin pairs are allocated to power I/O pads. They are named as _E, _W, and _S. (2 of such pairs located at the west side, 2 located at the south side, and 2 located at the east side of the chip)</td>
</tr>
</tbody>
</table>
7.2 Post-PAR Simulation Results

The chip functionality, as well as I/O timings, is verified by the post-PAR simulations in MODELSIM. First, IBM MM + BIST block is simulated with the default BIST configuration (cf. Table 6.3). In this case, data background pattern is 00 (in hexadecimal notation), full memory address range is scanned and test mode is set to FS (full scan). Then the simulations are repeated with both various data background patterns including FF, 55, and AA, and address ranges. Then, faults are injected into IBM MM by forcing some data output bits to a fixed value. The simulations are performed again for both SFF (stop-at-first-fault) and FS modes. The functionality of BIST core + SCI under a failure of the corresponding memory block is verified. The timing diagrams in Figure 7.2-7.3 correspond to BIST configuration with AA data background pattern, full address range and FS mode. Note that no faults are injected in this case. Figure 7.2 shows the waveform diagram obtained during the programming phase of the corresponding UASRs. At the beginning of the simulation, the asynchronous reset signals IBRESETnxSI and ISCRESETnxSI (active low) are set to logic-0 to clear all of the registers; after this step the resets are set to logic-1 and the UASRs are programmed. IBCLKxCI and ISSCLKxCI clock signals correspond to BIST and SCI clocks, respectively. The circuit is simulated at speed, that is the maximum clock frequency that can be achieved. IBM MM dictates the minimum clock period as 16.4 ns due to minimum clock active and restore time constraints. Hence, the periods of IBCLKxCI and ISSCLKxCI are set to 16.4 ns and 32.8 ns, respectively. Note that in the actual testing, a much more slower clock will be used for ISSCLKxCI, but a faster clock is preferred in the simulations for visualization purposes.

It takes 44 clock cycles to serially load the configuration bits. After the programming phase, ISTARTxSI signal is set to logic-1 for one clock cycle to initiate the BIST operation; IB_BUSYxSO is set to logic-1. Figure 7.3 depicts the testing phase. Once the test finishes, IB_BUSYxSO is set back to logic-0 and IBxSUCCEEDxSO is set to logic-1, indicating the test is completed successfully, without any failure. Also note ISC_OUTxDO is always at logic-0 during the simulations since no failure exists.
CHAPTER 7. THE TEST CHIP

Figure 7.2: Timing diagram during the programming phase for IBM MM + BIST

Figure 7.3: Timing diagram during the testing phase for IBM MM + BIST
Then, the FF based SCM + BIST block is simulated with the default BIST configuration. The simulations are repeated with both various data background patterns including FF, 55, and AA, and address ranges. Faults are also injected into the FF based SCM by forcing some data output bits to a fixed value. The simulations are performed again for both SFF (stop-at-first-fault) and FS modes. The functionality of BIST core + SCI under a failure of the memory block is verified. The timing diagrams in Figure 7.4-7.5 correspond to BIST configuration with AA data background pattern, full address range and FS mode. Figure 7.4 shows the waveform diagram obtained during the programming phase of the corresponding UASRs. At the beginning of the simulation, the asynchronous reset signals DBRESETnxSI and DSCRESETnxSI (active low) are set to logic-0 to clear all of the registers; after this step the resets are set to logic-1 and the UASRs are programmed. DBCLKxCI and DSSCLKxCI clock signals correspond to BIST and SCI clocks, respectively. The circuit is simulated at speed. The periods of DBCLKxCI and DSSCLKxCI are set to 10.8 ns (45% duty cycle) and 21.6 ns, respectively. Again, in the actual testing a much more slower clock will be used for DSSCLKxCI, but a faster clock is preferred in the simulations for visualization purposes.

It takes 36 clock cycles to serially load the configuration bits. After the programming phase, DSTARTxSI signal is set to logic-1 for one clock cycle to initiate the BIST operation; DBBUSYxSO is set to logic-1. Figure 7.5 depicts the testing phase. Once the test finishes, DBBUSYxSO is set back to logic-0 and DBxSUCCEEDxSO is set to logic-1, indicating the test is completed successfully, without any failure. Also note DSC_OUTxDO is always at logic-0 during the simulations since no failure exists.
CHAPTER 7. THE TEST CHIP

Figure 7.4: Timing diagram during the programming phase for FF based SCM + BIST

Figure 7.5: Timing diagram during the testing phase for FF based SCM + BIST
Afterwards, the latch based SCM and IBM MM stand alone are simulated. For both memories, random data is written into random addresses, while data is read from random addresses, for 1’000’000 times. Two non-overlapping clocks TEST_CLK and LBCLKxCI used to test latch based SCM are illustrated in Figure 7.6, where the data and the control signals are synchronized to the first clock while the memory itself receives the latter.

Figure 7.6: Two non-overlapping clocks used during the simulations
CHAPTER 7. THE TEST CHIP

7.3 Future Work

The fabricated test-chip will be heated up to 225 °C with a micro heater, which is designed and implemented by LPM at EPFL. It provides very precise control of temperature with 1 °C intervals. By this way, we will be able to control the chip temperature precisely during the measurements. We will test both packaged samples and bare dies for better characterization.

Digital part will be tested with Agilent Technologies 16717A Logic Analyzer and Pattern Generator. Its memory depth is 2M per channel at 167/333 MHz sampling frequency, and maximum number of channels is 48 (6 pods, 8 channels per pod) at 180 MHz frequency [57, 58]. These specifications are enough for high temperature testing purposes of digital blocks since we target 25 MHz clock frequency and independent test for each block.

The temperature behavior of each test structure in Analog part will also be analyzed. For instance, the output voltage levels of digital gates at various temperatures will be observed. The butterfly curves of SRAM cell will be obtained. In addition, by analyzing the behavior of single MOS transistors, we aim to have a clear understanding about the temperature limit of SOI 180 nm CMOS technology.

After the detailed measurements of Digital and Analog parts, the test data will be analyzed. IBM MM is expected to fail first compared to SCMs. In such a case, the future task is to design a reliable SCM with a smaller area. Based on the discussion in Chapter 5, the area of storage cell in SCMs is much larger than IBM MM cell. Therefore, a smaller storage cell needs to be designed in light of the feedback from the individual test structures. Specifically, the custom DFF cell with enable that we implement may need to be revised in order to achieve more robust and smaller cell design. On the other hand, it is hard to make an estimation regarding the performances of FF and latch based SCMs. This is because FFs and latches are implemented differently. In other words, these cells consist of different bistable elements. It is not possible to clearly interpret the layout of the LATSR cell. Depending on NAND or NOR based implementations, its high temperature behavior may vary immensely compared to FF cell which is cross-coupled inverter based. Moreover, forcing (or limiting) the synthesis of prospective SCM into specific cells can also be in question depending on the high temperature behavior of different standard cells. Nevertheless, we aim to comprehend this issue after the high temperature testing.
Chapter 8

Conclusions

In this thesis main objective was to design and implement a high temperature electronic control unit, which consists of a processor core, on-chip cache memories and the corresponding peripherals, for engine and control systems in new generation all-electric aircrafts as part of CREAM project. This block is required to operate reliably in severe thermal environment (> 200 °C). At such high temperatures, the performance of this unit is dictated by the memory modules since they are more prone to failures. Hence, the design of reliable high temperature memory blocks is the main emphasis of this thesis.

After reviewing basic transistor physics, high temperature effects in transistor level have been analyzed in detail in Chapter 2. At high temperatures, device characteristics are severely degraded primarily because of the increase subthreshold current as well as the reduction of device on-current. Based on this discussion, SOI technology has been found to be more promising for high temperature operation in comparison with bulk devices. The complete isolation of transistors with buried oxide and STI results in significant reduction of both leakage currents and the associated parasitic diffusion capacitances. Furthermore, improved SOI device switching characteristics along with lower $V_T$ allows drain current to significantly increase for such devices. Therefore, SOI devices are expected to outperform their bulk counterparts at high temperatures. IBM 180 nm SOI process technology is employed in this project.

Detailed analysis of high temperature effects in circuit level reveals the higher susceptibility of basic memory cells to consisting of bistable elements compared to combinational logic circuits. Very small SRAM cell sizes, as well as very close spacing of such cells, to achieve high memory densities also add to this higher susceptibility to the effects of leakage current. Besides, the operation of sense amplifiers, which is based
on detecting of small voltage difference between the bit lines, is also expected to be disturbed due to the increase of leakage current. Consequently, the cell content may be interpreted inaccurately even though the correct value is preserved.

Considering these issues, SCMs can be an alternative to technology compiled MM at high temperatures. A study has been performed to comprehend the trade-off between the area and access/setup time for such memories. This study reveals that SCMs larger than 2-Kbyte should be designed as the combination of smaller and effectively allocated memory modules since their area and access/setup time are much higher than that of IBM MM.

In order to assess high temperature performances of technology compiled MMs and SCMs, a test chip has been designed. Memory modules consisting of IBM MM, latch and FF based SCMs, as well as BIST structures have been implemented for testing purposes. Furthermore, the other test structures consisting of single devices such as transistors, various IBM standard cells, an extracted IBM SRAM cell, a ring oscillator, a frequency divider, and a basic linear amplifier are also included in the Analog part of the chip for high temperature (> 200 °C) characterization purposes of the target technology. On the other hand, this characterization is also essential for the implementation of the soft-core processor in the target technology. The test chip has been recently taped-out. Hence, measurements results can not be included in this thesis. Nevertheless, once the high temperature test of chip is completed, we will exploit the test results to design and implement a robust high temperature memory up to 225 °C.

Burak Erbağcı
April 26, 2011
Appendix A

Simulation Results

A.1 NMOS versus PMOS Transistors at High Temperatures

In the following, NMOS and PMOS transistors (\(L_c\)) with the same dimensions are simulated (\(W=500 \, \mu m, L=320 \, \mu m\)) as a function of temperature across different process corners \(tt, ss, ff, sf,\) and \(fs\). Both devices are degraded as the temperature increases. However, the degradation of NMOS transistor is more than that of PMOS transistor. Note that the dashed lines for 175 °C and 225 °C are extrapolated (non-guaranteed) results because \(BSIMSOI\) model [42] used during the simulation is valid up to 125 °C.

NMOS I-V Characteristics

The degradation is lower for smaller \(V_{GS}\) values. This is expected since the transistor enters the subthreshold regime in low voltage levels and the subthreshold current is significantly elevated at high temperatures. Also note I-V simulations (\(I_D\) with respect to \(V_{DS}\)) are provided only for \(V_{GS}=1.75 \, V\) and \(V_{GS}=2.5 \, V\) for simplicity. Device on currents plotted in Figure A.3 are the maximum \(I_D\) currents in the corresponding I-V curves.
Figure A.1: NMOS device $I_D$ versus $V_{DS}$, as a function of temperature ($V_{GS}=1.75$ V), across different corners: tt, ss, ff, sf, and fs
APPENDIX A. SIMULATION RESULTS

Figure A.2: NMOS device $I_D$ versus $V_{DS}$, as a function of temperature ($V_{GS}=$2.5 V), across different corners: tt, ss, ff, sf, and fs
Figure A.3: NMOS device on-current with respect to temperature, across different corners: tt, ss, ff, sf, and fs
PMOS I-V Characteristics

The degradation is lower for smaller $V_{SG}$ values. This is expected since the transistor enters the subthreshold regime in low voltage levels and the subthreshold current is significantly elevated at high temperatures. Also note I-V simulations ($I_D$ with respect to $V_{SD}$) are provided only for $V_{SG}=1.75$ V and $V_{SG}=2.5$ V for simplicity. Device on currents plotted in Figure A.6 are the maximum $I_D$ currents in the corresponding I-V curves.

![Graphs showing PMOS I-V characteristics](image)

Figure A.4: PMOS device $I_D$ versus $V_{SD}$, as a function of temperature ($V_{SG}$=1.75 V), across different corners: tt, ss, ff, sf, and fs
Figure A.5: PMOS device $I_D$ versus $V_{SD}$, as a function of temperature ($V_{SG}$=2.5 V), across different corners: tt, ss, ff, sf, and fs
Figure A.6: PMOS device on-current with respect to temperature, across different corners: tt, ss, ff, sf, and fs
NMOS versus PMOS

In the following, the ratio of device on-currents \( \frac{I_{D,NMOS}}{I_{D,PMOS}} \) is plotted as a function of temperature across various corners. It can be seen that this ratio drops as the temperature increases. Thus, NMOS transistor degrades more. Also note that for smaller \( V_{GS} \) (or \( V_{SG} \) for PMOS) values, the ratio is higher. This is because \( V_{GS} \) for PMOS (\( \approx 620 \) mV) is larger than that of NMOS (\( \approx 400 \) mV) and PMOS transistor starts to turn-off before.

Figure A.7: The ratio of NMOS and PMOS device on-currents with respect to temperature, across different corners: ss, ff, sf, and fs
A.2 Three-input NAND Gate at High Temperatures

In the following, NAND3 gate is simulated and the corresponding VTC curves (when the input A or B or C equals to logic-1, or all inputs makes a transition) as a function of temperature along with different corners (tt, fs and sf) are obtained. Note that since the degradation of NM and VTC is more pronounced in sf and fs corners, only these corners along with tt is simulated. However, the similar behavior is observed in ff and ss corners as in the inverter gate. Delay is also found to be increased as expected.

Figure A.8: Simulation results of a NAND3 gate (A, B: High), VTC and the corresponding NMs across different corners: tt, sf, and fs
Figure A.9: Simulation results of a NAND3 gate (A, C: High), VTC and the corresponding NMs across different corners: tt, sf, and fs
Figure A.10: Simulation results of a NAND3 gate (B, C: High), VTC and the corresponding NMVs across different corners: tt, sf, and fs
Figure A.11: Simulation results of a NAND3 gate (all inputs connected together), VTC and the corresponding NMs across different corners: tt, sf, and fs
Figure A.12: Intrinsic NAND3 delay with respect to the temperature and the process variation
Appendix B

Layouts

B.1 IBM MM 8K x 8 Array Configuration

The complete layout of IBM MM implemented in the test chip is shown below.

Figure B.1: Layout of IBM MM
B.2 Test Chip

The test chip layout is shown below. Note the intentional separation between each memory blocks in order to minimize any possible inter-block leakage effect.
B.3 FF and Latch based SCMs

In the following, layouts of FF and Latch based SCMs with various memory sizes are shown for comparison. In Figure B.3, the smallest and the largest correspond to 64-bit and 2-Kbyte FF based SCMs, respectively. In Figure B.4, the smallest and the largest correspond to 64-bit and 512-byte Latch based SCMs, respectively. Also note that the layouts for FF and latch based SCMs are not drawn into scale with respect to each other. FF based SCMs have the larger area.

Figure B.3: Layout of the FF based SCMs

Figure B.4: Layout of the Latch based SCMs
Appendix C

Verilog Codes

This appendix contains the Verilog codes for the following designs:

- Generic FF based SCM
- Generic Latch based SCM
- Memory BIST
  - the core
  - SCI

Parametric design of these block eases the synthesis and PAR processes for different memory sizes (number of rows from 8 to 2K).

C.1 Generic FF based SCM Verilog Code

There are four modules in this design: write address decoder, read logic, memory array and top level which instantiates all submodules.
**APPENDIX C. VERILOG CODES**

*Write Address Decoder Verilog Code*

```verilog
module AddrDecoder (A, RDWRT, DecodedAddr);
parameter ADDR = 7;
parameter SIZE = 2**ADDR;

input [ADDR-1:0] A;
input RDWRT;

output [SIZE-1:0] DecodedAddr;

wire [SIZE-1:0] Out_Array;

assign Out_Array = (1<<A);
assign DecodedAddr = (RDWRT) ? 0 : Out_Array;
endmodule
```

*Read Logic Verilog Code*

```verilog
//Read MUX
module ReadMUX(CLK, RDWRT, A, BL0, BL1, BL2, BL3, BL4, BL5, BL6, BL7, DOUT);
parameter ADDR = 7;
parameter SIZE = (2**ADDR);

input CLK;
input RDWRT;
input [ADDR-1:0] A;
input [SIZE-1:0] BL0;
input [SIZE-1:0] BL1;
input [SIZE-1:0] BL2;
input [SIZE-1:0] BL3;
input [SIZE-1:0] BL4;
input [SIZE-1:0] BL5;
input [SIZE-1:0] BL6;
input [SIZE-1:0] BL7;

output [7:0] DOUT;

wire [SIZE-1:0] DecodedAddr;
wire [SIZE-1:0] BL [7:0];
```
APPENDIX C. VERILOG CODES

```verilog
assign BL[0] = BL0;
assign BL[1] = BL1;
assign BL[2] = BL2;
assign BL[3] = BL3;
assign BL[4] = BL4;
assign BL[5] = BL5;
assign BL[6] = BL6;
assign BL[7] = BL7;

ReadMUXdecoder #(ADDR) ReadAddrDecoder (.CLK(CLK), .A(A), .RDWRT(RDWRT), .DecodedAddr (DecodedAddr) );
generate
  genvar i;
  for (i=0; i<8; i=i+1) begin : ReadOut
    assign DOUT[i] = (BL[i] & DecodedAddr);
  end
endgenerate
endmodule

// In order to latch the address input
module ReadMUXdecoder (CLK, A, RDWRT, DecodedAddr);
parameter ADDR = 7;
parameter SIZE = (2**ADDR);
input [ADDR-1:0] A;
input CLK;
input RDWRT;
output [SIZE-1:0] DecodedAddr;
wire [ADDR-1:0] OutA;
genvar l;
generate
  for (l=0; l< ADDR; l=l+1) begin : AL
    SDFF_H_BCC Addr (.CLK(CLK), .D(OutA[1]) , .SI(A[1]) , .SE(RDWRT) , .Q(OutA[1]) , .QBAR() );
  end
endgenerate
assign DecodedAddr = (1<<OutA);
endmodule
```
APPENDIX C. VERILOG CODES

Memory Array Verilog Code

```verilog
// Instantiate memory elements
module MemCells(CLK, DIN, CLS0, CLS1, CLS2, CLS3, CLS4, CLS5, CLS6, CLS7, BL0, BL1, BL2, BL3, BL4, BL5, BL6, BL7);

parameter ADDR = 7;
parameter SIZE = 2**ADDR;

input CLK;
input [7:0] DIN;
input [SIZE-1:0] CLS0;
input [SIZE-1:0] CLS1;
input [SIZE-1:0] CLS2;
input [SIZE-1:0] CLS3;
input [SIZE-1:0] CLS4;
input [SIZE-1:0] CLS5;
input [SIZE-1:0] CLS6;
input [SIZE-1:0] CLS7;

output [SIZE-1:0] BL0;
output [SIZE-1:0] BL1;
output [SIZE-1:0] BL2;
output [SIZE-1:0] BL3;
output [SIZE-1:0] BL4;
output [SIZE-1:0] BL5;
output [SIZE-1:0] BL6;
output [SIZE-1:0] BL7;

wire [SIZE-1:0] CLS [7:0];

assign CLS[0] = CLS0;
assign CLS[1] = CLS1;
assign CLS[2] = CLS2;
assign CLS[3] = CLS3;
assign CLS[4] = CLS4;
assign CLS[5] = CLS5;
assign CLS[6] = CLS6;
assign CLS[7] = CLS7;

wire [7:0] mem [SIZE-1:0];
genvar k,l;
generate

for (k=0; k< SIZE; k=k+1) begin: row
```

APPENDIX C. VERILOG CODES

43  `for (l=0; l<8; l=l+1) begin : column
44      SDFF_H_BC MemCell (.CLK(CLK) , .D(mem[k][l]) , .SI(DIN[1])
45      , .SE(CLS[1][k]) , .Q(mem[k][l]) , .QBAR());
46  end
47 endgenerate
48
49 generate
50 genvar b;
51  `for (b=0; b<SIZE; b=b+1) begin : BL
52    assign BL0[b] = mem[b][0];
53    assign BL1[b] = mem[b][1];
54    assign BL2[b] = mem[b][2];
55    assign BL3[b] = mem[b][3];
56    assign BL4[b] = mem[b][4];
57    assign BL5[b] = mem[b][5];
58    assign BL6[b] = mem[b][6];
59    assign BL7[b] = mem[b][7];
60  end
61 endgenerate
62
dendmodule
63
64 // Incorporate bit write signal to mimic IBM MM
65 module MemArray(CLK, DIN, BW, DecodedAddr, BL0, BL1, BL2, BL3, BL4, BL5, BL6, BL7);
66 parameter ADDR = 7;
67 parameter SIZE = 2**ADDR;
68
69 input CLK;
70 input [7:0] DIN;
71 input [7:0] BW;
72 input [SIZE-1:0] DecodedAddr;
73
74 output [SIZE-1:0] BL0;
75 output [SIZE-1:0] BL1;
76 output [SIZE-1:0] BL2;
77 output [SIZE-1:0] BL3;
78 output [SIZE-1:0] BL4;
79 output [SIZE-1:0] BL5;
80 output [SIZE-1:0] BL6;
81 output [SIZE-1:0] BL7;
82
83 wire [SIZE-1:0] CLS [7:0];
APPENDIX C. VERILOG CODES

generate
genvar i;

for (i=0;i<SIZE;i=i+1) begin : ColumnEnable
  assign CLS[0][i] = BW[0] & DecodedAddr[i];
  assign CLS[1][i] = BW[1] & DecodedAddr[i];
  assign CLS[3][i] = BW[3] & DecodedAddr[i];
  assign CLS[4][i] = BW[4] & DecodedAddr[i];
  assign CLS[5][i] = BW[5] & DecodedAddr[i];
  assign CLS[6][i] = BW[6] & DecodedAddr[i];
  assign CLS[7][i] = BW[7] & DecodedAddr[i];
end
endgenerate

MemCells #(ADDR) DFF

endmodule

Top Level Verilog Code

module DFF_SRAM(A, DIN, BW, CCLK, RDWRT, DOUT);
parameter ADDR = 9;
parameter SIZE = 2**ADDR;
input CCLK;
input [7:0] DIN;
input [7:0] BW;
input RDWRT;
input [ADDR-1:0] A;
output [7:0] DOUT;
wire [SIZE-1:0] DecodedAddr;
wire [SIZE-1:0] BL0;
wire [SIZE-1:0] BL1;
wire [SIZE-1:0] BL2;
wire [SIZE-1:0] BL3;
wire [SIZE-1:0] BL4;
wire [SIZE-1:0] BL5;
wire [SIZE-1:0] BL6;
wire [SIZE-1:0] BL7;
wire CLK;

assign CLK = !CCLK; // invert clock

MemArray #(ADDR) DFF_SRAM_Array (.CLK(CLK), .DIN(DIN), .BW(BW),
  .DecodedAddr(DecodedAddr), .BL0(BL0), .BL1(BL1), .BL2(BL2),
  .BL3(BL3), .BL4(BL4), .BL5(BL5), .BL6(BL6), .BL7(BL7));

AddrDecoder #(ADDR) DFF_SRAM.AddrDecoder (.A(A), .RDWRT(RDWRT),
  .DecodedAddr(DecodedAddr));

ReadMUX #(ADDR) DFF_ReadMUX (.CLK(CLK), .RDWRT(RDWRT), .A(A),
  .BL0(BL0), .BL1(BL1), .BL1(BL1), .BL1(BL1), .BL2(BL2),
  .BL3(BL3), .BL4(BL4), .BL5(BL5), .BL6(BL6), .BL7(BL7), .DOUT(DOUT));

endmodule
C.2  Generic Latch based SCM Verilog Code

There are three modules in this design: read logic, memory array and top level which instantiates all submodules.

Read Logic Verilog Code

```verilog
//Address is not latched unlike FF based SCM
module ReadMUX(A, BLS, DOUT);

parameter ADDR = 9;
parameter SIZE = (2**ADDR);
parameter BITS = SIZE*8;

input [ADDR−1:0] A;
input [0:BITS−1] BLS;

output [7:0] DOUT;

wire [SIZE−1:0] DecodedAddr;
wire [SIZE−1:0] BL [7:0];

assign DecodedAddr = (1<<A);

generate
  genvar k;
  genvar l;
  for (k=0;k<8;k=k+1) begin : Column
    for (l =0;l<SIZE ;l=l +1) begin : Row
      assign BL[k][l] = BLS[l*8+k];
    end
  end
endgenerate

generate
  genvar i;
  for (i=0;i<8;i=i +1)begin : ReadOut
    assign DOUT[i] = |(BL[i] & DecodedAddr);
  end
endgenerate
endmodule
```
APPENDIX C. VERILOG CODES

Memory Array Code

Note that Memory Array is implemented in VHDL.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

class latch_mem_1HVS is
  generic
    NUM_OF_WORDS : integer := 512;  -- Specifies the number of words the memory consists of
    BITS_PER WORD : integer := 8;  -- Specifies the number of bits per word
    ADDR_LENGTH : integer := 9;  -- Specifies the number of bits used for addressing, optimally should follow the formula: log2(NUM_OF_WORDS)=ADDR_LENGTH

  port
    L_clk : in std_logic;  -- Memory clock, negative edge
    L_rdwrt : in std_logic;  -- Read/Write choice, high=read, low=write
    L_a : in std_logic_vector(ADDR_LENGTH-1 downto 0);  -- Memory address
    L_bw : in std_logic_vector(BITS_PER_WORD-1 downto 0);  -- Word bit enable, simple masking, e.g L_bw=0011 enables the bits 0 and 1
```
APPENDIX C. VERILOG CODES

of the word
L_din : in std_logic_vector(BITS_PER_WORD-1
downto 0);  — Memory data input
L_array : out std_logic_vector(0 to (NUM_OF_WORDS*
BITS_PER_WORD)-1)  — Memory data output
)
end latch_mem_1HVS;

architecture rtl of latch_mem_1HVS is
  signal latch_en : std_logic_vector((NUM_OF_WORDS*
BITS_PER_WORD)-1 downto 0);  — latch enable, enable=high, disable=low
  signal gated_clk : std_logic_vector((NUM_OF_WORDS*
BITS_PER_WORD)-1 downto 0);  — Gated clock, negative edge

  type mem_array is array (0 to (NUM_OF_WORDS*BITS_PER_WORD)−1) of std_logic;  — Index convention: mem_array(i)=
target_line*BITS_PER_WORD+target_bit
  signal mem : mem_array;
begin
  -- Decoder
  word_dec : for current_word in 0 to NUM_OF_WORDS-1 generate
    bit_dec : for current_bit in 0 to BITS_PER_WORD-1 generate
      latch_en(current_word*BITS_PER_WORD+
current_bit) <= '1' when L_a =
std_logic_vector(to_unsigned( current_word,ADDR_LENGTH)) and L_rdwrt
= '0' and L_bw(current_bit) = '1' else
'0';
    end generate bit_dec;
  end generate word_dec;

  -- Gate Clocking
  word_clkg : for current_word in 0 to NUM_OF_WORDS-1 generate
    bit_clkg : for current_bit in 0 to BITS_PER_WORD-1 generate
      gated_clk(current_word*BITS_PER_WORD+
current_bit) <= L_cclk or not latch_en(
      current_word*BITS_PER_WORD+current_bit);
      — to write to latch, latch_en should be high and clock '0'
    end generate bit_clkg;
  end generate word_clkg;

APPENDIX C. VERILOG CODES

```
-- Write
word_write: for current_word in 0 to NUM_OF_WORDS-1
generate
  bit_write: for current_bit in 0 to BITS_PER_WORD-1
  generate
    signal Din : std_logic;
    begin
      Din <= L_din(current_bit);
      dl: process (gated_clk(current_word*
        BITS_PER_WORD+current_bit), Din)
      begin
        if gated_clk(current_word*
          BITS_PER_WORD+current_bit) = '0'
          then
            mem(current_word*
              BITS_PER_WORD+current_bit) <= Din;
        end if;
      end process;
    end generate bit_write;
  end generate word_write;
--to connect bit lines to One-hot MUX
L_array <= std_logic_vector(mem);
end rtl;
```

Top Level Verilog Code

```
module latch_mem(L_cclk, L_rdwr, L_a, L_bw, L_din, L_dout);
parameter ADDR = 7;
parameter SIZE = (2**ADDR);
parameter BITS = SIZE*8;

input L_cclk;
input L_rdwr;
input [ADDR-1:0] L_a;
input [7:0] L_bw;
input [7:0] L_din;

output [7:0] L_dout;
wire [0:BITS-1] L_array;
```
latch_mem_1HVS #(SIZE, 8, ADDR) latch_memory (.L_clk (L_clk),
   .L_rdwrt (L_rdwrt),
   .L_a (L_a),
   .L_bw (L_bw),
   .L_din (L_din),
   .L_array (L_array)
);

ReadMUX #(ADDR) read_mux (.A (L_a),
   .BLS (L_array),
   .DOUT (L_dout)
);
endmodule
There are two modules in this design: the core and SCI.

Core Verilog Code

```verilog
module mBIST(BCLK, BRESETn, B_a, B_bw, B_rdwrt, B_din,
             B_mem_start_addr, B_mem_stop_addr, B_test_data, B_control,
             B_dout, B_Start, B_failed, B_busy, B_succeed, SC_done, SC_start,
             SC_frame);

parameter ADDR = 9;
parameter frame_size = ADDR + 11;
input B_Start;
input BCLK;
input BRESETn;
input [ADDR-1:0] B_mem_start_addr;
input [ADDR-1:0] B_mem_stop_addr;
input [7:0] B_test_data;
input [1:0] B_control;

output [ADDR-1:0] B_a;
output reg [7:0] B_bw;
output reg B_rdwrt;
output [7:0] B_din;
output reg B_failed, B_busy, B_succeed, SC_start;
output [frame_size-1:0] SC_frame;

// parameters for the states
parameter ST_IDLE = 3’b000,
                ST_M1 = 3’b001,
                ST_M2_M3 = 3’b010,
                ST_M4_M5 = 3’b011,
                ST_M6 = 3’b100,
                ST_SCAN_OUT = 3’b101,
                ST_CONF = 3’b111;
```

C.3 Memory BIST Verilog Code
APPENDIX C. VERILOG CODES

31 reg [ADDR-1:0] cnt, w_cnt, Error_addr, w_Error_addr, counter, w_counter, w_Error_addr_out, Error_addr_out;
32 reg [2:0] pres_state, next_state;
33 reg inc, w_inc;
34 reg addr_dir, w_addr_dir, data_dir, w_data_dir;
35 reg w_latency, latency;
36 reg [7:0] test_data, data_read, w_error_data, error_data, w_error_data_out, error_data_out, data_pattern, w_data_pattern;
37 reg w_trace, trace, last_round_p, w_B_busy, w_B_failed, w_B_succeed;
38 reg [2:0] M_st_error, w_M_st_error;
39 reg w_last_round, last_round, w_SC_start, w_last_round_p;
40 reg ERROR, comp_en, eosc, done;
41
42 assign B_a = cnt; //RAM address is generated
43 assign B_din = !data_dir ? ~data_pattern : data_pattern; //RAM data input generation
44 assign SC_frame = {Error_addr_out, error_data_out, M_st_error}; //frame is constructed
45
46 always@ (posedge BCLK or negedge BRESETn) //2 cycles delayed version of data_read register (to facilitate pipelined operation during the test)
47 begin
48 if (!BRESETn)
49   M_st_error <= 0;
50 else
51   M_st_error <= w_M_st_error;
52 end
53
54 always@ (posedge BCLK or negedge BRESETn) //register for the data read from the corresponding SRAM
55 begin
56 if (!BRESETn)
57   data_read <= 0;
58 else
59   if (B_rdwt)
60     data_read <= B_dout;
61 end
62
63 always@ (posedge BCLK or negedge BRESETn) //1 cycle delayed version of data_read register (to facilitate pipelined operation during the test)
64 begin
65 if (!BRESETn)
always@(posedge BCLK or negedge BRESETn) // B.succeed, B.failed and busy registers
begin
if (!BRESETn)
  begin
    B_busy <= 0;
    B_failed <= 0;
    B_succeed <= 0;
  end
else
  begin
    B_busy <= w_B_busy;
    B_failed <= w_B_failed;
    B_succeed <= w_B_succeed;
  end
end

always@(posedge BCLK or negedge BRESETn) // 2 cycles delayed version of data_read register (to facilitate pipelined operation during the test)
begin
if (!BRESETn)
  error_data_out <= 0;
else
  error_data_out <= w_error_data_out;
end

always@(posedge BCLK or negedge BRESETn) // latched comparator
begin
if (!BRESETn)
  ERROR <= 1′b0;
else if (comp_en)
begin
  if (test_data == data_read)
    ERROR <= 1′b0;
  else
    ERROR <= 1′b1;
end
end
always@(posedge BCLK or negedge BRESETn) // scan chain enable signal (1 cycle delayed version of ERROR)
begin
if (!BRESETn)
  SC_start <= 0;
else
  SC_start <= w_SC_start;
end

always@(posedge BCLK or negedge BRESETn) // data pattern to be tested
begin
if (!BRESETn)
  data_pattern <= 0;
else
  data_pattern <= w_data_pattern;
end

always@(posedge BCLK or negedge BRESETn) // register for the test data pattern
begin
if (!data_dir)
  test_data <= data_pattern;
else
  test_data <= ~data_pattern;
end

always@(posedge BCLK or negedge BRESETn) // counter for the address generation
begin
if (!BRESETn)
  cnt <= 0;
else
  cnt <= w_cnt;
end

always@(posedge BCLK or negedge BRESETn) // 1 cycle delayed counter (to facilitate pipelined operation during the test)
begin
if ( !BRESETn)
    counter <= 0;
else
    counter <= w_counter;
end

always@ (posedge BCLK or negedge BRESETn) // 2 cycles delayed
    counter (to facilitate pipelined operation during the test)
    begin
        if ( !BRESETn)
            Error_addr <= 0;
        else
            Error_addr <= w_Error_addr;
    end

always@ (posedge BCLK or negedge BRESETn) // latency register for
    the final read operation
    begin
        if ( !BRESETn)
            latency <= 0;
        else
            latency <= w_latency;
    end

always@ (posedge BCLK or negedge BRESETn) // read/write indication
    flag for M2, M3, M4 and M5
    begin
        if ( !BRESETn)
            inc <= 0;
        else
            inc <= w_inc;
    end

always@ (posedge BCLK or negedge BRESETn) // address direction flag
    (0: ascending, 1: descending)
    begin
        if ( !BRESETn)
            addr_dir <= 0;
        else
            addr_dir <= w_addr_dir;
    end

always@ (posedge BCLK or negedge BRESETn) // test data flag (0: all
    −0 input test pattern, 1: all−1 input test_pattern)
    begin
if (!BRESETn)
  data_dir <= 0;
else
  data_dir <= w_data_dir;
end

always@(posedge BCLK or negedge BRESETn) // state register
begin
  if (!BRESETn)
    pres_state <= ST_IDLE;
  else
    pres_state <= next_state;
end

always@(posedge BCLK or negedge BRESETn) // last_round and 1 cycle
  delayed version of this flag for the final read operation
begin
  if (!BRESETn)
    begin
      last_round <= 0;
      last_round_p <= 0;
    end
  else
    begin
      last_round <= w_last_round;
      last_round_p <= w_last_round_p;
    end
end

always@(posedge BCLK or negedge BRESETn) // trace flag to restore
  from faulty location (if ctrl[1] is set to 1)
begin
  if (!BRESETn)
    trace <= 1'b0;
  else
    trace <= w_trace;
end

always@(posedge BCLK or negedge BRESETn) // TX_sc_done
begin
  if (!BRESETn)
    done <= 1'b1;
  else
    done <= SC_done;
end
always@(posedge BCLK or negedge BRESETn) // Error_addr (Read Only)
begin
  if (!BRESETn)
    Error_addr_out <= 0;
  else
    Error_addr_out <= w_Error_addr_out;
end

always@* // combinational next state logic
begin
  // initialization of the corresponding internal signals
eosc = 0;
  w_SC_start = SC_start;
  w_B_failed = B_failed;
  w_B_succeed = B_succeed;
  w_B_busy = B_busy;
  next_state = pres_state;
  B_bw = 0;
  B_rdwr = 1'b0;
  w_data_dir = data_dir;
  w_addr_dir = addr_dir;
  w_cnt = cnt;
  w_inc = inc;
  comp_en = 1'b1;
  w_last_round = last_round;
  w_last_round_p = last_round_p;
  w_trace = trace;
  w_Error_addr = Error_addr;
  w_error_data = error_data;
  w_error_data_out = error_data_out;
  w_Error_addr_out = Error_addr_out;
  w_M_st_error = M_st_error;
  w_data_pattern = data_pattern;
  w_latency = latency;
  w_counter = counter;

  // combinational state machine
  case(pres_state)
    ST_IDLE: // idle state
      begin
        w_M_st_error = 3'b000;
        w_counter = 0;
        w_trace = 1'b0;
  endcase
end
comp_en = 1'b0;

if (B_Start) // set the corresponding flags
begin
    if (B_control[1] | B_control[0])
    begin
        next_state = ST_CONF;
        w_data_pattern = B_test_data; // load test data
        w_cnt = B_mem_start_addr; // load memory start address
        w_addr_dir = 1'b0;
        w_data_dir = 1'b1;
        w_inc = 0;
        w_last_round = 1'b0;
        w_B_busy = 1'b1;
        w_B_failed = 1'b0;
        w_B_succeed = 1'b0;
    end
end

ST_CONF: // configuration state for loading internal registers
begin
    comp_en = 1'b0;
    next_state = ST_M1;
end

ST_M1: // the first March element is applied (initialize memory with all 0's in the ascending address order)
begin
    w_cnt = cnt + 1;
    B_bw = 8'hFF;
    comp_en = 1'b0;
    if (cnt == B_mem_stop_addr)
    begin
        next_state = ST_M2_M3;
        w_cnt = B_mem_start_addr;
        w_data_dir = 1'b0;
    end
end

ST_M2_M3: // the second and third March elements are applied for each address location in ascending order (
APPENDIX C. VERILOG CODES

first read 0's, write 1's, read 1's, write 0's)

begin
  if (inc & !ERROR)
    w_M_st_error = {trace, addr_dir, data_dir},
    B_bw = 8'hFF;
    w_inc = ~inc;
  if (!inc)
    begin
      B_rdwrt = 1'b1;
      comp_en = 1'b0;
    end
  else
    begin
      B_rdwrt = 1'b0;
      w_Error_addr_out = cnt;
      if (cnt == B_mem_stop_addr)
        begin
          if (!data_dir)
            w_cnt = B_mem_start_addr;
          else
            w_cnt = cnt;
          w_data_dir = ~data_dir;
        end
      else
        w_cnt = cnt + 1;
    end
  end

w_error_data = data_read;
w_error_data_out = error_data;

if (ERROR)
  begin
    next_state = ST_SCAN_OUT;
    w_B_failed = 1'b1;
    w_SC_start = 1;
  end
else if (cnt == B_mem_stop_addr & data_dir & inc)
  begin
    w_addr_dir = ~addr_dir;
    next_state = ST_M4_M5;
  end
end
ST_M4_M5:  //the fourth and fifth March elements are
applied for each address location in descending order (first read 0’s, write 1’s, read 1’s, write 0’s)
begin
    if (inc & !ERROR)
        w_M_st_error = {trace, addr_dir, data_dir};

    w_Error_addr = B_mem_start_addr;
    B_bw = 8'hFF;
    w_inc = !inc;

    if (!inc)
        begin
            B_rdwrt = 1'b1;
            comp_en = 1'b0;
        end
    else
        begin
            B_rdwrt = 1'b0;
            w_Error_addr_out = cnt;

            if (cnt == B_mem_start_addr)
                begin
                    w_data_dir = !data_dir;
                    w_cnt = B_mem_stop_addr;
                end
            else
                w_cnt = cnt - 1;
        end
    w_error_data = data_read;
    w_error_data_out = error_data;

    if (ERROR)
        begin
            next_state = ST_SCAN_OUT;
            w_B_failed = 1'b1;
            w_SC_start = 1;
        end
    else if (cnt == B_mem_start_addr & inc & data_dir)
        begin
            w_trace = !trace;
            next_state = ST_M6;
        end
end
APPENDIX C. VERILOG CODES

```verilog
ST_M6: // the last March element is applied for all address locations in descending order (read 0’s)
beg indifferent
    w_inc = 1'b1;
    comp_en = inc & !last_round_p;
    B_rdwrt = !last_round;

    if (cnt != B_mem_start_addr)
        w_cnt = cnt - 1;
    else
        begin
            w_last_round = 1;
            w_cnt = cnt;
            w_latency = last_round_p;
        end

    w_counter = cnt;
    w_Error_addr = counter;
    w_last_round_p = last_round;
    w_error_data = data_read;
    w_error_data_out = error_data;

    if (inc & !ERROR)
        w_M_st_error = {trace, 1'b0, data_dir};
    if (ERROR)
        begin
            w_Error_addr_out = Error_addr;
            next_state = ST_SCAN_OUT;
            w_B_failed = 1'b1;
            w_SC_start = 1;
            w_inc = inc;
        end
    else
        begin
            if (last_round_p)
                begin
                    next_state = ST_IDLE;
                    w_B_succeed = ~B_failed;
                    w_B_busy = 1'b0;
                end
        end
end
```
APPENDIX C. VERILOG CODES

ST_SCAN_OUT:
begin
    comp_en = 0;
    w_B_succeed = 1'b0;
if (!SC_start && done)
begin
    w_last_round_p = last_round;
    w_M_st_error = {trace, addr_dir, data_dir};
if (B_control[1] && !B_control[0])
begin
case ({trace, addr_dir})
2'b00:
begin
    w_Error_addr_out = cnt;
    w_error_data = data_read;
    w_error_data_out = error_data;
    B_bw = 8'hff;
    B_rdwr = 1'b0;
    w_cnt = cnt +1;
    comp_en = 1'b1;
    w_inc = ~inc;
    if (cnt == B_mem_stop_addr)
begin
    if (!data_dir)
begin
        w_cnt = B_mem_start_addr;
        next_state = ST_M2_M3;
        w_SC_start = 0;
    end
else
begin
    next_state = ST_M4_M5;
    w_addr_dir = ~addr_dir;
    w_cnt = cnt;
    w_SC_start = 0;
end
w_data_dir = ~data_dir;
end
else
begin
    next_state = ST_M2_M3;
    w_SC_start = 0;
end
end
end
end
APPENDIX C. VERILOG CODES

492 | end
493 | 2'b01:
494 | begin
495 |     w_Error_addr_out = cnt;
496 |     w_error_data = data_read;
497 |     w_error_data_out = error_data;
498 |     B_bw = 8'hff;
499 |     B_rdwrt = 1'b0;
500 |     w_cnt = cnt - 1;
501 |     comp_en = 1'b1;
502 |     w_inc = ~inc;
503 |     if (cnt == B_mem_start_addr)
504 |         begin
505 |             if (data_dir)
506 |                 begin
507 |                     next_state = ST_M6;
508 |                     w_trace = ~trace;
509 |                 end
510 |             else
511 |                 next_state = ST_M4_M5;
512 |         end
513 |     end
514 | else
515 | begin
516 |     w_cnt = B_mem_stop_addr;
517 |     w_data_dir = ~data_dir;
518 |     w_SC_start = 0;
519 | end
520 | else
521 | begin
522 |     next_state = ST_M4_M5;
523 |     w_SC_start = 0;
524 | end
525 | 2'b11:
526 | begin
527 |     comp_en = !last_round_p;
528 |     B_rdwrt = !last_round;
529 |     w_error_data = data_read;
530 |     w_error_data_out = error_data;
531 |     w_Error_addr = counter;
532 |     w_counter = cnt;
533 |     if (cnt != B_mem_start_addr)
534 |         w_cnt = cnt - 1;
535 | else
536 | begin
w_cnt = cnt;
w_latency = last_round_p;
w_last_round = 1;
end

if(ERROR & inc & !latency)
begin
  next_state = ST_SCAN_OUT;
  w_M_st_error [1] = 1'b0;
  w_Error_addr_out = Error_addr;
  w_SC_start = 1;
end
else
begin
  if(last_round_p)
  begin
    next_state = ST_IDLE;
    w_B_busy = 1'b0;
    w_error_data = error_data;
    w_error_data_out = error_data_out;
    w_M_st_error [1] = 1'b0;
    w_SC_start = 0;
  end
  else
  begin
    next_state = ST_M6;
    w_M_st_error [1] = 1'b0;
    w_SC_start = 0;
    w_inc = 1;
  end
end
default:
begin
  next_state = ST_IDLE;
  w_SC_start = 0;
end
endcase
else
begin
  next_state = ST_IDLE;
  w_B_busy = 1'b0;
  w_SC_start = 0;
end
```verilog
APPENDIX C. VERILOG CODES

582 | end
583 | else if (SC_start && done)
584 |    w_SC_start = 1;
585 | else
586 |    w_SC_start = 0;
587 | end
588 | default:
589 | begin
590 |    next_state = ST_IDLE;
591 | end
592 | endcase
593 | end
594 | endmodule

SCI Verilog Code

1 module SC(SC_CLK, SC_RESETn, SC_start, SC_in, B_busy_in,
            B_failed_in, B_succeed_in, SC_frame, B_start_in, B_busy,
            B_failed, B_succeed, SC_out, SC_done, B_start_out,
            B_mem_start_addr, B_mem_stop_addr, B_test_data, B_control);
2 parameter ADDR = 9;
3 parameter RANGE = 2**ADDR;
4 parameter frame_size = ADDR + 11;
5 parameter scan_out_size = frame_size + 4;
6
7 input SC_CLK;
8 input SC_RESETn;
9 input SC_start;
10 input SC_in;
11 input B_busy_in;
12 input B_failed_in;
13 input B_succeed_in;
14 input [frame_size-1:0] SC_frame; //Current data to be scanned out
   [error addr, error data, MARCH state in which the error occured]
15 input B_start_in;       //Start input
16
17 output B_busy;
18 output B_failed;
19 output B_succeed;
20 output SC_done;
21 output B_start_out;
22 output SC_out;
```
APPENDIX C. VERILOG CODES

output [ADDR−1:0] B_mem_start_addr;
output [ADDR−1:0] B_mem_stop_addr;
output [7:0] B_test_data;
output [1:0] B_control;

parameter ST_IDLE = 4'b0000,
ST_CONF1 = 4'b0001,
ST_CONF2 = 4'b0010,
ST_CONF3 = 4'b0011,
ST_LOAD1 = 4'b0100,
ST_LOAD2 = 4'b0101,
ST_LOAD3 = 4'b0110,
ST_LOAD4 = 4'b0111,
ST_LAST = 4'b1000,
ST_READY = 4'b1001,
ST_SC_OUT = 4'b1010;

reg [scan_out_size−1:0] shift_reg, w_shift_reg; // shift_reg:
["1", error address, "1", error data, "1", MARCH state in which
the error occurred, "1"]
reg [6:0] SC_counter, w_SC_counter;
reg [3:0] pres_state, next_state;
reg start, SC_done, w_SC_done, start_ok, ack, w_ack;
reg [ADDR−1:0] start_addr, w_start_addr;
reg [ADDR−1:0] stop_addr, w_stop_addr;
reg [7:0] data_pattern, w_data_pattern;
reg [1:0] control, w_control;
reg conf, w_conf;

assign B_mem_start_addr = start_addr;
assign B_mem_stop_addr = stop_addr;
assign B_test_data = data_pattern;
assign B_control = control;

assign B_busy = B_busy_in;
assign B_failed = B_failed_in;
assign B_succeed = B_succeed_in;
assign B_start_out = B_start_in & start_ok;

always@(posedge SC_CLK or negedge SC_RESETn) //sampled SC_start
begin
if (!SC_RESETn)
    start <= 1'b0;
else
start <= SC_start;
always@(posedge SC_CLK or negedge SC_RESETn) //SC_done register
begin
  if (!SC_RESETn)
    SC_done <= 1'b1;
  else
    SC_done <= w_SC_done;
end
always@(posedge SC_CLK or negedge SC_RESETn) //state register
begin
  if (!SC_RESETn)
    pres_state <= ST_IDLE;
  else
    pres_state <= next_state;
end
always@(posedge SC_CLK or negedge SC_RESETn) //shift register
begin
  if (!SC_RESETn)
    shift_reg <= 0;
  else
    shift_reg <= w_shift_reg;
end
always@(posedge SC_CLK or negedge SC_RESETn) //counter
begin
  if (!SC_RESETn)
    SC_counter <= 0;
  else
    SC_counter <= w_SC_counter;
end
always@(posedge SC_CLK or negedge SC_RESETn) //handshaking signal
for B_busy received
begin
  if (!SC_RESETn)
    ack <= 0;
  else
    ack <= w_ack;
end
always@(posedge SC_CLK or negedge SC_RESETn) //scanned start_addr

APPENDIX C. VERILOG CODES

begin
if (!SC_RESETn)
  start_addr <= 0;
else
  start_addr <= w_start_addr;
end

always@ (posedge SC_CLK or negedge SC_RESETn) //scanned stop_addr
begin
if (!SC_RESETn)
  stop_addr <= 0;
else
  stop_addr <= w_stop_addr;
end

always@ (posedge SC_CLK or negedge SC_RESETn) //scanned data_pattern
begin
if (!SC_RESETn)
  data_pattern <= 0;
else
  data_pattern <= w_data_pattern;
end

always@ (posedge SC_CLK or negedge SC_RESETn) //scanned control
begin
if (!SC_RESETn)
  control <= 0;
else
  control <= w_control;
end

always@ (posedge SC_CLK or negedge SC_RESETn) //conf register
begin
if (!SC_RESETn)
  conf <= 0;
else
  conf <= w_conf;
end

assign SC_out = shift_reg[scan_out_size - 1];

always@*
begin
  //initialization
  start_ok = 1’b0;
w_SC_done = SC_done;
w_shift_reg = shift_reg;
next_state = pres_state;
w_ack = ack;
w_SC_counter = SC_counter;
w_start_addr = start_addr;
w_stop_addr = stop_addr;
w_data_pattern = data_pattern;
w_control = control;
w_conf = conf;

// combinational state machine
case (pres_state)
ST_IDLE:
begin
  if (SC_in)
    next_state = ST_CONF1;
end

ST_CONF1:
begin
  w_conf = SC_in;
  next_state = ST_CONF2;
end

ST_CONF2:
begin
  if (conf==SC_in)  // 00 or 11
    next_state = ST_CONF3;
  else  // Configuration bits are not properly loaded!
    next_state = ST_IDLE;
end

ST_CONF3:
begin
  w_conf = 0;
  if (SC_in & !conf)
    begin  // default conf
      next_state = ST_READY;
      w_start_addr = 0;
      w_stop_addr = RANGE-1;
      w_data_pattern = 0;
      w_control = 2'b10;
      w_ack = 1'b1;
    end
  else if (SC_in & conf)  // load control
begin
    next_state = ST_LOAD1;
end
else if (!SC_in & conf) // use default control:10
begin
    next_state = ST_LOAD2;
    w_control = 2'b10;
end
else // conf error
    next_state = ST_IDLE;
end

ST_LOAD1: // configure test – control bits
begin
    w_SC_counter = SC_counter + 1'b1;
    w_control = {control[0], SC_in};
    if (SC_counter == 2)
    begin
        w_SC_counter = 0;
        if (SC_in) // load start_address
        begin
            w_control = control;
            next_state = ST_LOAD2;
            w_conf = 1;
            // w_ack = 1'b1;
        end
        else
        begin
            w_control = control;
            w_start_addr = 0; // use default start_addr = 0
            next_state = ST_LOAD3;
            w_conf = 0;
        end
    end
end

ST_LOAD2: // start_address
begin
    w_conf = 1;
    if (conf)
    begin
        w_SC_counter = SC_counter + 1'b1;
    end
w_start_addr = {start_addr[ADDR-2:0], SC_in};
if (SC_counter == ADDR)
begin
  w_SC_counter = 0;
  w_start_addr = start_addr;
  if (SC_in) //load stop_address
  begin
    next_state = ST_LOAD3;
    w_conf = 1;
  end
else //use default stop_address RANGE-1
begin
  w_stop_addr = RANGE-1;
  next_state = ST_LOAD4;
  w_conf = 0;
end
end
end
else
begin
  if (!SC_in) //use default start_addr
  begin
    w_start_addr = 0;
    next_state = ST_LOAD3;
    w_conf = 0;
  end
end
end

ST_LOAD3: //stop_address
begin
  w_conf = 1;
  if (conf)
  begin
    w_SC_counter = SC_counter + 1'b1;
    w_stop_addr = {stop_addr[ADDR-2:0], SC_in};
    if (SC_counter == ADDR)
    begin
      w_stop_addr = stop_addr;
      if (SC_in) //load data pattern
      begin
        next_state = ST_LOAD4;
      end
      w_SC_counter = 0;
    end
  end
end
```verilog
APPENDIX C. VERILOG CODES

289 w_conf = 1;
290 end
else     // use default data pattern = all 0s
292 begin
293 w_SC_counter = 0;
294 w_conf = 1;
295 next_state = ST_LAST;
296 end
end
end
else
begin
301 if (!SC_in)     // use default stop_addr
302 begin
303 w_stop_addr = RANGE-1;
304 next_state = ST_LOAD4;
305 w_conf = 0;
306 end
end
end
end
ST_LOAD4:  // data pattern
310 begin
311 w_conf = 1;
312
313 if (conf)
314 begin
315 w_SC_counter = SC_counter + 1'b1;
316 w_data_pattern = {data_pattern[6:0], SC_in};
317 if (SC_counter == 8)
318 begin
319 w_SC_counter = 0;
320
322 if (SC_in)     // pattern ok - test configuration ok
323 begin
324 w_data_pattern = data_pattern ;
325 next_state = ST_READY;
326 w_ack = 1'b1;
327 end
else
begin     // error—abort scanning in
329 w_conf = 0;
331 next_state = ST_IDLE;
332 end
```

end
end
else begin
    if (!SC_in)
        begin
            w_data_pattern = 0;
            next_state = ST_LAST;
            w_conf = 0;
        end
end
end
ST_LAST:
begin
    f(SC_in)
    begin
        next_state = ST_READY;
        w_ack = 1'b1;
    end
else
    next_state = ST_IDLE;
end
ST_READY:
begin
    start_ok = 1'b1;  // unmask B_start signal
    if (B_busy_in)
        w_ack = 1'b0;
    if (start)
        begin
            w_shift_reg = {1'b1, SC_frame[frame_size -1:11], 1'b1, SC_frame[10:3], 1'b1,
                            SC_frame[2:0], 1'b1};
            w_SC_done = 1'b0;
            next_state = ST_SC_OUT;
        end
    else if (!ack & !B_busy_in)
        begin
            if (SC_in)
                next_state = ST_CONF1;
        end
end

APPENDIX C. VERILOG CODES

```verilog
function ST_SC_OUT;
begin
    
    w_SC_counter = SC_counter + 1'b1;
    w_shift_reg = {shift_reg[scan_out_size-2:0], 1'b0};
    if (SC_counter == scan_out_size-1)
        begin
            w_SC_counter = 0;
            w_SC_done = 1'b1;
            next_state = ST_READY;
        end
    default:
        begin
        end
endcase
end
```
Bibliography


