Multilevel Gain Cell Arrays for Fault-Tolerant VLSI Systems

Muhammad Umer Khalid
Multilevel Gain Cell Arrays for Fault-Tolerant VLSI Systems

Master’s Thesis Spring 2011

Muhammad Umer Khalid
Exchange student from Linköping University Sweden to EPFL Switzerland

Supervisors
Prof. Andreas Burg, TCL, EPFL.
Dr. J Jacob Wikner, ISY, LiU.

Advisor
Pascal Meinerzhagen, TCL, EPFL.

Examiner
Prof. Dr. Mark Vesterbacka, ISY, LiU.
**Abstract**

Embedded memories dominate area, power and cost of modern very large scale integrated circuits system on chips (VLSI SoCs). Furthermore, due to process variations, it becomes challenging to design reliable energy efficient systems. Therefore, fault-tolerant designs will be area efficient, cost effective and have low power consumption. The idea of this project is to design embedded memories where reliability is intentionally compromised to increase storage density.

Gain cell memories are smaller than SRAM and unlike DRAM they are logic compatible. In multilevel DRAM storage density is increased by storing two bits per cell without reducing feature size. This thesis targets multilevel read and write schemes that provide short access time, small area overhead and are highly reliable. First, timing analysis of reference design is performed for read and write operation. An analytical model of write bit line (WBL) is developed to have an estimate of write delay. Replica technique is designed to generate the delay and track variations of storage array. Design of replica technique is accomplished by designing replica column, read and write control circuits. A memory controller is designed to control the read and write operation in multilevel DRAM. A multilevel DRAM is with storage capacity of eight kilobits is designed in UMC 90 nm technology. Simulations are performed for testing and results are reported for energy and access time. Monte Carlo analysis is done for variation tolerance of replica technique. Finally, multilevel DRAM with replica technique is compared with reference design to check the improvement in access times.

**Keywords**

DRAM, SRAM, gain cell, multilevel, fault-tolerant, replica technique, finite state machine, PVT variations.
Acknowledgements

There are many individuals that made this thesis successful. I would like to express my gratitude for them.

I am thankful to Prof. Dr. Andreas Burg for providing me the opportunity to carry out my thesis at Telecommunication Circuits Laboratory, EPFL. I appreciate his support for administrative issues and technical discussions. Furthermore, I thank Christian Senning for setting up the computers and tools. I would also like to thank Dr. Jacob Wikener for support and guidance. Above all, I am extremely grateful to Pascal Meinerzhagen for valuable guidance and support. He was always there for discussions and reviewing my manuscripts.

I thank Rashid Iqbal from the core of heart for support and encouragement. Finally, I would also like to thank my parents for their constant and valuable support during thesis.
Abstract

Embedded memories dominate area, power and cost of modern very large scale integrated circuits system on chips (VLSI SoCs). Furthermore, due to process variations, it becomes challenging to design reliable energy efficient systems. Therefore, fault-tolerant designs will be area efficient, cost effective and have low power consumption. The idea of this project is to design embedded memories where reliability is intentionally compromised to increase storage density.

Gain cell memories are smaller than SRAM and unlike DRAM they are logic compatible. In multilevel DRAM storage density is increased by storing two bits per cell without reducing feature size. This thesis targets multilevel read and write schemes that provide short access time, small area overhead and are highly reliable. First, timing analysis of reference design is performed for read and write operation. An analytical model of write bit line (WBL) is developed to have an estimate of write delay. Replica technique is designed to generate the delay and track variations of storage array. Design of replica technique is accomplished by designing replica column, read and write control circuits. A memory controller is designed to control the read and write operation in multilevel DRAM. A multilevel DRAM is with storage capacity of eight kilobits is designed in UMC 90 nm technology. Simulations are performed for testing and results are reported for energy and access time. Monte Carlo analysis is done for variation tolerance of replica technique. Finally, multilevel DRAM with replica technique is compared with reference design to check the improvement in access times.
## List of Acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Stands for</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSP</td>
<td>Digital signal processing</td>
<td>Representation of discrete time signals by a sequence of numbers and processing these signals.</td>
</tr>
<tr>
<td>GC</td>
<td>Gain cell</td>
<td>Basic storage cell of memory. Composed of three transistors and storing two bits.</td>
</tr>
<tr>
<td>WT</td>
<td>Write transistor</td>
<td>pMOS transistor in gain cell to provide write access.</td>
</tr>
<tr>
<td>ST</td>
<td>Storage transistor</td>
<td>nMOS transistor in gain cell to store two bit data.</td>
</tr>
<tr>
<td>RT</td>
<td>Read transistor</td>
<td>nMOS transistor in gain cell to provide read access.</td>
</tr>
<tr>
<td>SN</td>
<td>Storage node</td>
<td>The gate of storage transistor where data is stored.</td>
</tr>
<tr>
<td>AGC</td>
<td>Active gain cell</td>
<td>Gain cell storing 2 bit data.</td>
</tr>
<tr>
<td>RGC</td>
<td>Reference gain cell</td>
<td>Gain cell storing reference voltage level for comparison during read operation.</td>
</tr>
<tr>
<td>MLDRAM</td>
<td>Multilevel DRAM</td>
<td>Multilevel dynamic random access memory having more than one bit per cell.</td>
</tr>
<tr>
<td>WWL</td>
<td>Write word line</td>
<td>The gate of write transistor that controls write access.</td>
</tr>
<tr>
<td>RWL</td>
<td>Read word line</td>
<td>The gate of write transistor that controls read access.</td>
</tr>
<tr>
<td>WBL</td>
<td>Write bit line</td>
<td>Bit line is storage array to which gain cell is connected through write transistor.</td>
</tr>
<tr>
<td>RBL</td>
<td>Read bit line</td>
<td>Bit line is storage array to which gain cell is connected through read transistor.</td>
</tr>
<tr>
<td>PVTS</td>
<td>Process, voltage, temperature, scenario</td>
<td>Process corner, supply voltage, operating temperature and scenario represents voltage levels in active gain cell and reference gain cell.</td>
</tr>
<tr>
<td>kbs</td>
<td>Kilo bits</td>
<td>Unit to represent the storage capacity of memory.</td>
</tr>
<tr>
<td>nMOS</td>
<td>n-type MOS transistor</td>
<td>n-channel metal oxide semiconductor field effect transistor with majority of electrons.</td>
</tr>
<tr>
<td>pMOS</td>
<td>p-type MOS transistor</td>
<td>p-channel metal oxide semiconductor field effect transistor with majority of holes.</td>
</tr>
<tr>
<td>FSM</td>
<td>Finite state machine</td>
<td>A sequential logic circuit with finite number of states. Some logic operations are performed in each state.</td>
</tr>
</tbody>
</table>
# Table of Contents

Chapter 1: Introduction .......................................................................................................................... 1
  1.1 Introduction ................................................................................................................................. 1
  1.2 Reference Design ....................................................................................................................... 1
  1.3 Contribution ............................................................................................................................... 2
  1.4 Project Outline ............................................................................................................................ 2

Chapter 2: Analysis of Reference Design ............................................................................................ 3
  2.1 Multilevel Gain Cell ..................................................................................................................... 3
  2.2 Storage and Reference Levels ...................................................................................................... 4
  2.3 Storage Array .............................................................................................................................. 5
    2.3.1 Array Column ......................................................................................................................... 5
  2.4 Multilevel Write ........................................................................................................................... 6
    2.4.1 Write Circuit ......................................................................................................................... 6
    2.4.2 Voltage Level Generation ...................................................................................................... 6
    2.4.3 Write Operation .................................................................................................................... 7
  2.5 Timing Analysis for Write Operation .......................................................................................... 8
    2.5.1 Worst Case for Write Operation ............................................................................................. 9
    2.5.2 Monte Carlo Analysis for Write Operation ............................................................................. 9
    2.5.3 Timing Improvements for Write Bit Line .............................................................................. 11
    2.5.4 Effect of Charge Sharing on WBL ....................................................................................... 13
  2.6 Multilevel Read ............................................................................................................................ 13
    2.6.1 Read Operation ...................................................................................................................... 13
    2.6.2 Worst Case for Read Operation ............................................................................................ 14

Chapter 3: Analytical Model for Write Bit Line .................................................................................. 16
  3.1 Introduction ............................................................................................................................... 16
  3.2 Write Bit Line Model ................................................................................................................... 16
  3.3 Comparison of Model with Actual Circuit ................................................................................. 18

Chapter 4: Replica Technique for MLDRAM ..................................................................................... 22
  4.1 Problem and Solution ................................................................................................................... 22
  4.2 Multilevel DRAM with Replica Technique ................................................................................ 22
  4.3 Organization of Modules ............................................................................................................. 23
    4.3.1 Signal Flow: Write Operation ............................................................................................... 24
    4.3.2 Signal Flow: Read Operation ............................................................................................... 24
7.5 Access Time ........................................................................................................... 49
7.6 Improvement ........................................................................................................... 49
Chapter 8: Conclusion ........................................................................................................... 51
Chapter 9: Future Work ........................................................................................................... 53
References .......................................................................................................................... 54
Appendix .......................................................................................................................... 56
   A. VHDL Code for Memory Controller ............................................................................. 56
   B. Synopsys Script for Synthesis ...................................................................................... 64
   C. MatLab files for Memory Model and Stimuli Generation ............................................ 66
   D. OceanScript for Comparison ..................................................................................... 75
List of Figures

Figure 1: Multilevel gain cell ................................................................. 3
Figure 2: Area comparison of 8 kilobits SRAM and multilevel gain cell DRAM ............... 4
Figure 3: Marco memory architecture ....................................................... 5
Figure 4: Single array column ................................................................. 5
Figure 5: Write circuit ........................................................................ 6
Figure 6: Simple model of WBL ............................................................. 7
Figure 7: Write operation .................................................................... 8
Figure 8: Precharging time .................................................................. 10
Figure 9: Charge sharing and transferring ............................................. 10
Figure 10: Error in generated level ....................................................... 11
Figure 11: Simplified schematic and waveforms for read operation ................. 14
Figure 12: Model of single segment of WBL ........................................... 16
Figure 13: Simplified model of single WBL segment ................................ 16
Figure 14: Simplified model of WBL ..................................................... 17
Figure 15: Block diagram of WBL for level1 scenario .............................. 18
Figure 16: RC model of WBL for level1 scenario ................................... 19
Figure 17: Simplified RC chain for WBL ............................................... 19
Figure 18: Comparison of modeled and specter simulation for step input ........... 21
Figure 19: Block diagram of MLDRAM with replica technique .................. 22
Figure 20: System level waveforms ...................................................... 23
Figure 21: Organization of modules ...................................................... 24
Figure 22: State diagram of memory controller ....................................... 26
Figure 23: Replica column input and output waveforms ............................ 28
Figure 24: Replica column circuit diagram ............................................ 30
Figure 25: Write control circuit schematic ............................................. 31
Figure 26: Write control signal waveforms .......................................... 33
Figure 27: Read control circuit schematic ............................................. 34
Figure 28: Read control signal waveforms ............................................ 36
Figure 29: 2 to 4 NAND decoder ......................................................... 37
Figure 30: Multiplexer and enable circuit .............................................. 38
Figure 31: Simulation flow ................................................................ 39
Figure 32: Process variations tracking for write operation ........................................42
Figure 33: Voltage variation tracking for write operation ........................................43
Figure 34: Temperature variation tracking for write operation ...................................44
Figure 35: Process variation tracking for read operation ..........................................45
Figure 36: Voltage variation tracking for read operation ..........................................46
Figure 37: Temperature variation tracking for read operation ...................................47
List of Tables

Table 1: Storage and reference levels ................................................................. 4
Table 2: Voltage levels and pre(dis)charging capacitors .................................. 7
Table 3: Write worst case .................................................................................... 9
Table 4: Effect of sizing and initial conditions on WBL performance .............. 12
Table 5: WBL performance under worst case .................................................. 12
Table 6: Effect of charge sharing on WBL ....................................................... 13
Table 7: Worst case for read operation ............................................................. 15
Table 8: Values of R and C extracted from WBL circuit .................................. 19
Table 9: Transistor sizes for storage array WBL components ......................... 27
Table 10: Transistor sizes for replica WBL components .................................. 28
Table 11: Transistor sizes for replica gain cell and read bit line pull up device .... 29
Table 12: Access times for nominal operating conditions .................................. 40
Table 13: Access times for worst case conditions ............................................ 40
Table 14: Energy consumption per bit ............................................................... 41
Table 15: Frequency comparison .................................................................... 49
Table 16: Access time comparison .................................................................. 49
Chapter 1: Introduction

1.1 Introduction

Embedded memories dominate area, cost and power consumption of modern digital signal processing (DSP) systems with applications ranging from telecommunication to cryptography [1]. Because of increasing process variations, higher defect levels and large leakage currents, it is difficult to design reliable systems in deep submicron CMOS technologies. Causing a shift in design approach toward fault-tolerant VLSI design [2, 3] that can result in small area reduced cost and low power consumption. The idea behind this project is to increase the storage density by intentionally compromising the reliability.

Embedded memories are usually implemented as: (a) flip-flops or latch arrays, (b) SRAM macrocells and (c) DRAM macrocells [4]. Flip-flops and latch arrays are suitable for small storage and their area becomes excessively large for storage bigger than few kilobits [5]. Conventional 1T1C embedded DRAM is not logic compatible because it requires special processes to build high density stacked or trench capacitors [6]. On the other hand, conventional 6T SRAM is compatible with standard digital CMOS technologies. However, the area of 6T SRAM one-bit per storage cell is 12.5 times bigger than a typical 1T1C DRAM one-bit per storage cell [7].

Gain Cell based DRAMs can result in area smaller than SRAM, while at the same time they are logic compatible [7]. Various gain-cell-based DRAMs have been proposed to date, with different inherent gain cells [7–11].

By reducing the physical size of storage cell and by adopting three-dimensional cell capacitor structures, the per area storage density of conventional 1T1C DRAM has been increased dramatically [12]. Multilevel DRAM (MLDRAM) exploits an additional dimension to increase storage density by storing more than one-bit per cell, without further reduction in feature size [12]. Various MLDRAMs have been proposed to date [12–15]. In 2001, Koob et al. have designed the first MLDRAM that has been proven in silicon for two to six signal levels [16]. All MLDRAMs proposed to this day use the conventional 1T-1C one-bit storage cell. To our knowledge, the fact that several bits can as well be stored in a single gain cell has not been exploited yet.

1.2 Reference Design

There are two previous designs that increase the storage density at the cost of reduced retention time [17–18]. This first multilevel gain cell array uses an optimized storage and reference level allocation scheme for recovering as much retention time as possible when storing 2 bits per cell, but exhibits excessively high read failure rates due to process variations when implemented in sub-100-nm CMOS technologies. A second multilevel gain cell array [19, 20] uses a more conservative level allocation scheme and operates with a reasonably small read failure rate in a 90-nm CMOS technology. In this design, silicon area has successfully been reduced by giving up 100% reliable operation, thereby providing an interesting storage solution for fault-tolerant systems. However, there is still a lot of room for improvements, especially in the array access time.
1.3 Contribution

The aim of this Master’s Thesis is to further optimize the multilevel gain cell array reported in [18] to make it even more attractive for the integration in fault-tolerant VLSI systems in deep-submicron CMOS technologies. The main bottleneck of the reference design [18] is the long write and read access times.

First, the reference design is analyzed to understand multilevel read and write operations. I investigated and compared different multilevel write and read schemes, aiming for short access times, small area overhead, and high reliability. Furthermore, I designed a replica technique to dynamically track the delay of storage array in presence of process, voltage and temperature (PVT) variations. A memory controller is designed that controls the read and write operations. Simulations are performed to report access times and energy consumption. Monte Carlo analysis is done to show the variation tolerance of design. Finally, a comparison between the current and reference design is done.

1.4 Project Outline

This research project is carried out as a master’s thesis in Telecommunication Circuits Laboratory at École Polytechnique Fédérale de Lausanne Switzerland.

The objective of this project is to design an access scheme to improve the write and read access times of reference design, at the cost of small area overhead and this scheme should not introduce errors in system in addition to error due small size transistors in gains. In order to accomplish this objective, parasitic extraction is done and reference design is analyzed for read and write operation with the help Monte Carlo simulations. An analytical model for write bit line is developed to study the effect of changing the width of pull up device and transmission gates in write bit line. The accuracy of model is verified by Spectre simulations. Access times are improved by adopting replica technique. One addition column is designed that tracks the variations and provides the required amount of delay. Feedback based write and read control circuits are designed in Cadence Virtuoso to generate the necessary control signals. A finite state machine coded in VHDL, simulated in ModelSim and synthesized by Synopsys Design Vision is also designed to handle the inputs and outputs. An 8 kilobit multilevel dynamic random access memory is designed in UMC 90nm technology. Furthermore, a simple model of memory is developed in MatLab to verify memory operation. Finally, Monte Carlo and parametric analysis is performed utilizing Cadence Spectre simulator to show the variation tolerance of design.
Chapter 2: Analysis of Reference Design

The detailed analysis of reference design is presented in this chapter. Section 2.1 discusses the multilevel gain cell which is the basic storage cell of reference design. Data and reference voltage levels are discussed in section 2.2. The following section provides information about memory macro architecture. Write operation and timing analysis of write operation is discussed in section 2.4 and 2.5 respectively. Section 2.6 gives knowledge about read operation and timing analysis.

2.1 Multilevel Gain Cell

Multilevel gain cell used for this memory is composed of three transistors. As shown in figure 1, the write transistor (WT) is a pMOS device whereas the storage transistor (ST) and read transistor (RT) are nMOS devices. Here a separate transistor RT is used for read access in order to avoid the masking issues during read operation [19].

In the above circuit, the drain current of storage transistor is modulated by the voltage at storage node [21]. Furthermore, due to capacitive coupling storage node voltage is boosted when read word line pull up during read [19]. Since two bits are stored in a single cell so the above circuit is given the name multilevel gain cell.

Multilevel gain cell is implemented in UMC 90 nm technology that provides both standard performance and low leakage transistors. Therefore, the write transistor is chosen as low leakage high threshold voltage (LLHVT) transistor to minimize the sub threshold leakage and increase retention time. Storage transistor is implemented as low leakage low threshold voltage (LLLVT) device to minimize gate tunneling current and maximum storage node voltage range for which storage transistor is on. Whereas, for fast read operation, read transistor is implemented as standard process low threshold voltage (SPLVT) device [17].

The area of the mixed nMOS and pMOS gain cell is more than all nMOS or pMOS configuration gain cell. Since all nMOS or pMOS configuration will result in compact layout. But its disadvantage is that all nMOS or pMOS approach requires a boosted supply and level
shifters to transfer highest storage level to gain cell [19]. Since the cell area is mostly limited by contacts. Therefore, layout of the mixed gain cell is drawn in such a way to share area between contacts. Hence, the overall area for the mixed gain cell memory is less than only nMOS or pMOS approach [19].

Multilevel gain cell has a number of advantages. The area of multilevel gain cell DRAM is approximately half of the area of SRAM of same capacity [19]. Unlike DRAM, multilevel gain cell is logic compatible. Furthermore, the read through multilevel gain cell is non-destructive because the data is stored at the gate of storage transistor. Non-destructive read is extremely important for multilevel sensing where successive comparisons are done to read back data. Figure 2 shows the area comparison of eight kilobits SRAM and multilevel gain cell DRAM.

Figure 2: Area comparison of 8 kilobits SRAM and multilevel gain cell DRAM

2.2 Storage and Reference Levels

Since two bit are stored per cell in the multilevel gain cell DRAM. Therefore four voltage levels are required to represent two bit data. Also, three reference levels are required for comparison during read operation. The storage and reference level are shown in table 1.

<table>
<thead>
<tr>
<th>Storage Levels</th>
<th>Reference Levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>Voltage</td>
<td>Data</td>
</tr>
<tr>
<td>1.1 V</td>
<td>11</td>
</tr>
<tr>
<td>900 mV</td>
<td>10</td>
</tr>
<tr>
<td>700 mV</td>
<td>01</td>
</tr>
<tr>
<td>500 mV</td>
<td>00</td>
</tr>
</tbody>
</table>

Table 1: Storage and reference levels
The storage range on the upper side is limited by the supply voltage, while on the lower side it is limited by the threshold of storage transistor. There is a noise margin of 100 mV between two consecutive levels.

2.3 Storage Array

In the reference design memory is composed of two storage arrays each one with a capacity of four kilobits (4 kbs). A single storage array has 128 words for data and four words for reference voltage levels, with 32-bit per each word. Figure 3 shows the memory macro highlighting bit line switches, the reference cell, the sense amplifier and the bit line equalizer.

In figure 3 blocks labeled with GC represent the gain cells storing data, whereas the blocks labeled as RGC represent reference gain cells storing reference voltage levels. WBL is the write bit line for storage array a and storage array b. RBL is the read bit line for both storage arrays. WWL and RWL are the write and read word line signals for different words in both storage arrays. The control signal for bit line equalizer is BLP. However, SAP and SAN are the control signals for pMOS and nMOS devices in the sense amplifier respectively. C0 and C1 are control signals to open and close bit line switches.

2.3.1 Array Column

The storage array has 16 columns. A single column is shown in figure 4.
As shown in figure 4 each column in the storage array has write bit line (WBL) and read bit line (RBL). WBL start with a pull up device and ends with a pull down device. WBL is divided into 12 segments by eleven transmission gates. Each segment has eleven gain cell attached to it. All the gain cells are attached to RBL via read access transistor.

2.4 Multilevel Write

Since two-bits are stored per gain cell in MLDRAM. For this purpose, there is a need to generate four storage levels representing two-bit data. All the voltage levels are generated locally by charge sharing among the bit line segments [14, 22]. Charge sharing approach is area efficient because it utilizes the hardware that is already there in the array column without any area overhead [17, 19].

2.4.1 Write Circuit

The write circuit in MLDRAM is composed of the write bit lines and gain cells attached to them. Basic building blocks of a single write bit line are pull up, a sub bit line segment, transmission gate switches and a pull down circuit. It can be seen in figure 5 that the WBL is organized in such a way that there are the twelve sub bit line segments (each having eleven gain cells). These segments are connected together through eleven transmission gates. The top most bit line segment is connected to the pull up while the bottom segment is connected to the pull down circuit. In order to generate the voltage levels for data and reference (seven levels in total), seven switches are required on WBL for level generation. But there are eleven switches in total on WBL. Four dummy switches from pull up side are used to have uniform capacitance along bit line and for layout symmetry.

2.4.2 Voltage Level Generation

The voltage levels are generated locally by charge sharing among the bit line segments. The process of voltage generation can be easily understood with the help of simple model for WBL shown in figure 6.
Initially all the switches on the WBL are closed, while the pull up and pull down switches are open. Now if we want to generate the storage level 1 (1.1 V). First step is to divide the WBL into two parts in order to get the right number of capacitance for both the segments i.e. 11 for the segment connected to the pull up and 1 for the segment connected to the pull down switch. This is accomplished by opening the corresponding switch sw11 on WBL. Now the pull up and pull down devices are enabled to precharge one segment with 11 capacitors to vdd and predischarge the other segment with 1 capacitor to gnd. After that the required voltage level is achieved on both the segments the pull up and pull down devices are turned off. The last step is to close the switch corresponding to the required voltage level i.e. sw11 to share the charge among the two segments of bit line. In this way the required voltage level i.e. 1.1 V is generated locally without any area overhead.

The following expression is used to calculate the voltage being generated.

\[ V_x = \frac{\alpha}{\alpha + \beta} V_{dd} \]  

Eq. 2.1

Where \( \alpha \) and \( \beta \) are the number of unit capacitors being precharged and predischarged respectively.

The voltage levels used in the reference design along with the number of capacitors per segment (i.e. \( \alpha \) and \( \beta \)) are shown in table 2.

<table>
<thead>
<tr>
<th>Level</th>
<th>Voltage</th>
<th>Caps (precharged)</th>
<th>Caps (predischarged)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage level 1</td>
<td>1.1 V</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>Reference level 1</td>
<td>1 V</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>Storage level 2</td>
<td>900 mV</td>
<td>9</td>
<td>3</td>
</tr>
<tr>
<td>Reference level 2</td>
<td>800 mV</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>Storage level 3</td>
<td>700 mV</td>
<td>7</td>
<td>5</td>
</tr>
<tr>
<td>Reference level 3</td>
<td>600 mV</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Storage level 4</td>
<td>500 mV</td>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 2: Voltage levels and pre(dis)charging capacitors

2.4.3 Write Operation

Write operation in MLDRAM is completed in three phases. Phase one is precharging, second is charge sharing phase and third phase is level transfer. Write operation is expatiated with the aid of figure 7. Figure 7 shows simplified schematic of WBL and a single gain cell. First of
all, WBL is divided into two segments by opening the switch. Pull up and pull down devices are enabled. As a result segment1 is precharged to Vdd, while segment2 is predischarged as shown in waveforms in figure 7. When precharging is done, the pull up and pull down devices are disabled. The switch on WBL is closed to share the charge between two segments and desired voltage level is generated. It is important to make sure that control signals for precharging and charge sharing are non-overlapping for accurate level generation. Finally, write word line (WWL) is pulled down to enable write access transistor (WT) and data is written to gain cell at storage node (SN). The waveforms of the control signals and voltage at SN are shown in figure 7.

![Diagram of the Write Operation](image)

**Figure 7: Write operation**

### 2.5 Timing Analysis for Write Operation

Timing analysis for the write bit line has been performed. To make the analysis accurate, parasitic extraction was performed for pull up, pull down, gain cell and transmission gate. Test bench with av_extracted views for all the components has been designed. Times are reported for the signals values form 1% to 99% of Vdd.

**Simulation Results for Storage Level 1 (1.1 V)**

The simulation results for the storage level 1 are presented. Charging sharing and writing are performed at the same time.

- Precharging time for 11 bit line segments = 5.424 ns
- Charge sharing and writing time = 2 ns
Multilevel Gain Cell Arrays for Fault Tolerant VLSI Systems

- Voltage level produced = 1.095 V and error in produced level is -5 mV.

The mentioned numbers indicate the amount of time required by the reference design to write data.

### 2.5.1 Worst Case for Write Operation

The worst case PVTS (process, voltage, temperature and scenario) for write operation is determined. The circuit is simulated with different PVTS condition and results were observed to find worst case.

**Scenario:** There are two extreme scenarios, one is storage level1 where eleven capacitors have to be precharged and the second is storage level4 where seven capacitors have to be predischarged. Simulation results show that storage level1 takes more time and is worst scenario.

**Process:** For storage level1 due to slow-slow process corner precharging will take more time and is worst process corner.

**Voltage:** Low voltage 1.08 V (-10% of Vdd) produces less current and as a result the precharging is slow.

**Temperature:** Simulations results show that at a high temperature of 85°C precharging is slow.

Following table 3 shows the worst case.

<table>
<thead>
<tr>
<th>Process</th>
<th>Voltage</th>
<th>Temperature</th>
<th>Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS process corner</td>
<td>1.08 V</td>
<td>85°C</td>
<td>Storage level 4 (1.1 V)</td>
</tr>
</tbody>
</table>

**Table 3: Write worst case**

**Simulation Results for Worst Case**

- Precharging time for eleven bit line segments = 8.176 ns
- Charge sharing and writing time = 2.66 ns
- Voltage level produced = 986 V and error in produced level is -4 mV.

The above mentioned results show the worst case timing for write operation of reference design.

### 2.5.2 Monte Carlo Analysis for Write Operation

Monte Carlo analysis was performed to have the worst case timing due to variations and mismatch between different components. Simulation results are shown in the following figures.

The results of Monte Carlo show that the precharging time for eleven segments can take 8.2 nsec in the worst case as shown in figure 8. The time for charge sharing and storing data at storage node can be 3.3 nsec, shown in figure 9. Furthermore, a variation of 1.37 mV in generated voltage is observed in figure 10.
Figure 8: Precharging time

Figure 9: Charge sharing and transferring
2.5.3 Timing Improvements for Write Bit Line

The WBL takes a time of almost 14 nsec to write data at storage node. This corresponds to a frequency of 71.4 MHz which is quite slow. To make the write process fast different techniques such as sizing of transistors and initial condition on bit line are applied to check the improvement in timing and area overhead cost.

The test bench with av_extracted view of gain cell and schematic views to observe the effect of sizing for the pull up device, pull down device and transmission gate are setup. A resistor capacitor (RC) network was added to each segment of bit line to make the test bench similar to actual WBL. The values of resistor and capacitor were calculated from the netlist of WBL with av_extracted views for all the blocks. This RC network represents the resistance and capacitance of interconnect of pull up, pull down and transmission gates to WBL.
The timing results for different sizing of components and initial conditions are presented in the following table 4.

<table>
<thead>
<tr>
<th>Size</th>
<th>Same size as reference</th>
<th>Double pull up</th>
<th>Double transmission gate</th>
<th>Double pull up and transmission gate</th>
<th>Double pull up and transmission gate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial conditions</td>
<td>Default</td>
<td>Default</td>
<td>Default</td>
<td>Default</td>
<td>IC1</td>
</tr>
<tr>
<td>Precharging time</td>
<td>5.092 n</td>
<td>4.132 n</td>
<td>3.978 n</td>
<td>2.768 n</td>
<td>2.413 n</td>
</tr>
<tr>
<td>Sharing time</td>
<td>2.12 n</td>
<td>1.9 n</td>
<td>1.21 n</td>
<td>1.22 n</td>
<td>1.04 n</td>
</tr>
<tr>
<td>Voltage generated</td>
<td>1.097 V</td>
<td>1.098 V</td>
<td>1.1 V</td>
<td>1.101 V</td>
<td>1.103 V</td>
</tr>
<tr>
<td>Error in voltage generated</td>
<td>-3 mV</td>
<td>-2 mV</td>
<td>0</td>
<td>+1 mV</td>
<td>+3 mV</td>
</tr>
<tr>
<td>Timing improvement</td>
<td>18.85 %</td>
<td>21.87 %</td>
<td>45.64 %</td>
<td>52.61 %</td>
<td></td>
</tr>
<tr>
<td>Area overhead</td>
<td>1.32 %</td>
<td>8 %</td>
<td>9.32 %</td>
<td>9.32 %</td>
<td></td>
</tr>
</tbody>
</table>

**Table 4: Effect of sizing and initial conditions on WBL performance**

Default: represents the initial condition when 11 segments of WBL are predischarged to 0 V and 12th segment is precharged to 1.2 V.

IC1: represents the initial condition of WBL when all segments are precharged to 500 mV.

The table 4 shows the tradeoffs under normal operating conditions. But under worst case PVTS condition the results are shown in table 5.

<table>
<thead>
<tr>
<th>Size</th>
<th>Same size as reference</th>
<th>Double pull up and transmission gate</th>
<th>Double pull up and transmission gate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial conditions</td>
<td>Default</td>
<td>Default</td>
<td>IC1</td>
</tr>
<tr>
<td>Precharging time</td>
<td>7.661 n</td>
<td>4.002 n</td>
<td>3.434 n</td>
</tr>
<tr>
<td>Sharing time</td>
<td>2.42 n</td>
<td>1.64 n</td>
<td>1.36 n</td>
</tr>
<tr>
<td>Voltage generated</td>
<td>986 mV</td>
<td>991 V</td>
<td>995 mV</td>
</tr>
<tr>
<td>Error in voltage generated</td>
<td>-4 mV</td>
<td>+1 mV</td>
<td>+5 mV</td>
</tr>
<tr>
<td>Timing improvement</td>
<td>47.76 %</td>
<td>55.17 %</td>
<td></td>
</tr>
<tr>
<td>Area overhead</td>
<td>9.32 %</td>
<td>9.32 %</td>
<td></td>
</tr>
</tbody>
</table>

**Table 5: WBL performance under worst case**
In worst case conditions with double size pull up and transmission gates the writing process can be completed in 5 nsec corresponding to a frequency of 200 MHz.

### 2.5.4 Effect of Charge Sharing on WBL

WBL is simulated in such a way that after that one segment of WBL is precharged to Vdd and the other segment is predischarged to gnd. Then the corresponding switch is closed and charge transistor is turned on and voltage is passed to storage node.

A comparison of 2 scenarios for storage level 1 (1.1 V) is presented in the following table 6.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Precharging time</th>
<th>Sharing time</th>
<th>Storing time</th>
<th>Voltage generated</th>
<th>Voltage stored</th>
</tr>
</thead>
<tbody>
<tr>
<td>First charge sharing then storing</td>
<td>10.1 nsec</td>
<td>4 nsec</td>
<td>2.2 nsec</td>
<td>1.105 V</td>
<td>1.094 V</td>
</tr>
<tr>
<td>Sharing and storing at same time</td>
<td>10.1 nsec</td>
<td>4.41 nsec</td>
<td></td>
<td></td>
<td>1.094 V</td>
</tr>
</tbody>
</table>

**Table 6: Effect of charge sharing on WBL**

It has been observed that the approach of first charge sharing and then storing (writing) data takes more time and has no effect on bit line voltage. Also the final voltage stored is same for both cases.

Therefore it better to perform the charge sharing and writing at the same time for better performance.

### 2.6 Multilevel Read

Reading in MLDRAM is done in sequential fashion, where successive comparisons are done to read back two-bit data. Sequential reading results in small area of readout circuits [17, 19].

#### 2.6.1 Read Operation

Read operation is accomplished in three steps. First step is precharging of RBLs, second is voltage difference development and last step is sensing.

Figure 11 shows that the circuit for read operation in composed of active gain cell (AGC) storing data, a reference gain cell (RGC) storing reference level, a convention cross coupled inverter sense amplifier to do comparison and a bit line precharge and equalizer circuit. Read operation is explained in detail with the help of figure 11 in the following paragraph.

Read operation starts with the precharging of RBLs associated with gain cell to be compared. Bit line precharge and equalizer circuit is enabled to precharge the RBLs to Vdd. Read word line (RWL) signal for both the gain cells are pull high at the same time to discharge RBLs. Since the voltage levels at the storage node of active and reference gain cells are different. Therefore the discharging currents of two gain cells are also different as shown in waveforms in figure 11. As a result, the RBLs associated with active and reference gain cell discharge unequally fast and a voltage difference is developed. Sense amplifier is
then kicked-in to do comparison. Hence, sense amplifier pulls one RBL to Vdd while the other one is pulled to ground as shown in waveforms of figure 11. First comparison provides the most significant bit (MSB) of two bit data. To read back least significant bit (LSB) second reference level is written to reference gain cell and comparison is done in the similar way.

Figure 11: Simplified schematic and waveforms for read operation

### 2.6.2 Worst Case for Read Operation

The worst case PVTS (process, voltage, temperature and scenario) for read operation is determined. The circuit is simulated with different PVTS condition and results were observed to find worst case.

**Scenario:** There are two extreme scenarios. First scenario indicates the comparison between the lowest storage and reference levels i.e. 500 mV and 600 mV respectively. Second scenario indicates the comparison between the highest storage and reference levels i.e. 1.1 V and 1 V respectively. It is observed in simulation that first scenario takes more time because the gain cell will discharge slowly due to small voltage level at SN as compared to second scenario. Hence first scenario defines the worst case from access time point of view.

**Process:** Due to slow-slow process corner discharging of RBLs will take more time and is worst process corner.

**Voltage:** Low voltage 1.08 V (-10% Vdd) results in less current and reading is slow.

**Temperature:** Simulations results show that at a high temperature of 85°C defines the worst case temperature.
Following table 7 shows the worst case for read operation.

<table>
<thead>
<tr>
<th>Process</th>
<th>Voltage</th>
<th>Temperature</th>
<th>Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS process corner</td>
<td>1.08 V</td>
<td>85°C</td>
<td>Storage level 4 (500 mV) and reference level 3 (600 mV)</td>
</tr>
</tbody>
</table>

**Table 7: Worst case for read operation**

Table 7 indicates the worst case for read operation and simulation results show that reference design takes eight nsec to perform single comparison.
Chapter 3: Analytical Model for Write Bit Line

This chapter sheds light on the design of analytical model to estimate the delay of write bit line. Also the comparison of this model with Spectre simulation is present in this chapter.

3.1 Introduction

An analytical model for write bit line (WBL) is developed to study the sizing effect of different WBL components on delay of WBL. This model is based on Elmore delay [6].

3.2 Write Bit Line Model

The write bit line composed of a number of segments. Each segment has a number gain cells attached to it. These segments are connected together through transmission gates. As the gain cells are inactive during the voltage generation phase so the WBL consists of the pull up and transmission gates. The exact model a single segment of WBL is shown in figure 12.

![Figure 12: Model of single segment of WBL](image)

In the above model $C_{wbl1}$ and $C_{wbl2}$ are the collective junction capacitances at two ends of transmission gate. $Ron_p$ and $Ron_n$ are the on resistances of pMOS and nMOS in the transmission gate. The junction capacitances of the write access transistor of gain cell attached to a segment of bit line are also shown in the above model in figure 12.

A simplified model of a single segment of WBL is shown in figure 13. For transmission gates the two parallel resistors $Ron_p$ and $Ron_n$ are replaced with the equivalent resistance $Req$. Also all parallel capacitors $C_{wbl2}$, $C_{wbl1}$ and junction capacitors of WT of gains which are attached to the same node are replaced by equivalent capacitance of $Ceq$.

![Figure 13: Simplified model of single WBL segment](image)

In order to make the analysis simple, a simplified model for write bit line is show in figure 14. In this model $Ron_{pu}$ is the on resistance of the pull up circuit. $Ct1$ is the equivalent capacitance of three parallel capacitors $Css_{pu}$, $Ceq_{gc}$ (capacitance due to all gain cell
attached to a segment) and $C_{\text{wbl1}}$ attached to segment1. For the last segment $C_{tN}$ represents the equivalent capacitance of $C_{\text{wbl2}}$ and $C_{eq\_gc}$ attached to segment N. For all the segments between the first and last segment $Req1$ to $ReqN$ is the equivalent resistance of the transmission gates. Also $Ceq1$ to $Ceq\_N-1$ is the equivalent capacitance due to three parallel capacitors $C_{\text{wbl2}}$, $C_{eq\_gc}$ and $C_{\text{wbl1}}$ attached to that node.

![Figure 14: Simplified model of WBL](image)

The Elmore delay at any node $n$ for the above RC chain networks can be derived with the equation 3.1.

$$
\tau_{DN} = \sum_{i=1}^{N} C_i \sum_{j=1}^{i} R_j = \sum_{i=1}^{N} C_i R_{ii} 
$$

**Eq. 3.1**

For example the equivalent time constant at segment 3 is given by:

$$
\tau_{D3} = R_{on\_pu} C_{t1} + (R_{on\_pu} + R_{eq1}) C_{eq1} + (R_{on\_pu} + R_{eq1} + R_{eq2}) C_{eq2}
$$

As the equivalent resistance and capacitance for all transmission gates between two segments are same so they are replaced by RC.

$$
\tau_{D3} = R_{on\_pu} C_{t1} + 2(R_{on\_pu} C) + 3RC
$$

The voltage at any node $n$ on the WBL in response to the step input is given by the exponential function in equation 3.4.

$$
V_n(t) = V_0 (1 - e^{-t/\tau})
$$

**Eq. 3.2**

Where $V_0$ is the input voltage and $\tau_{DN}$ is the time constant of the node $n$. 

---

Muhammad Umer Khalid | Telecommunication Circuits Laboratory, EPFL
The time taken by any node voltage from 0 to 99 % of the input voltage is given by:

\[ V(0.99V) = 0.99 \alpha \Rightarrow t = 4.6 \tau \]  \hspace{2cm} \text{Eq. 3.3}

The simplified model for WBL is an RC chain network that is characterized by a number of time constants. The voltage equation for such a network involves a set of coupled differential equations and no close form solution exists for such a network [23]. Since most of the output waveforms are dominated by a single pole, so the Elmore expression determines the value of dominant one, which is the first moment of the impulse response of the circuit [6]. So the response of circuit is a first order approximation of the actual response of the circuit from input to any node n.

In case of multiple pull up devices for WBL the response of the circuit can be determined with the help of well-known Superposition theorem.

### 3.3 Comparison of Model with Actual Circuit

The actual circuit of the write bit line is compared with the designed model to have an idea of the accuracy of the model. So the circuit used for comparison is a particular scenario of storage level 1 where we have to generate a voltage level of 1.1 V. Figure 15 show the block diagram for this particular scenario.

![Figure 15: Block diagram of WBL for level1 scenario](image)

Block diagram shows that the WBL consists of eleven segments. Segment one is connected to the pull up circuit. First five segments are connected together through 4 dummy transmission gates. As a result segments one to five are shorted together. The rest of segments are connected via transmission gates.

The RC model for the write bit line circuit for level1 scenario is shown in figure 16. In the following model Ron\_pu is the on resistance and Cpu the junction capacitance of drain terminal of the pull up device. CgcN is the collective drain junction capacitance of write transistors (WT) of eleven gain cells attached to segment N. Total capacitance due to dummy transmission gates is represented by CdcN. Ron\_tN is the on resistance, Cwbl1 and Cwbl2 are the junction capacitance at the terminals of any transmission gate.
The values of resistances and capacitances extracted by the DC analysis of the circuit are shown in the following table 8.

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>On resistance of pull up (Ron_{pu})</td>
<td>1.48 K ohm</td>
</tr>
<tr>
<td>Junction capacitance of pull up (C_{pu})</td>
<td>1.5 f Farad</td>
</tr>
<tr>
<td>Junction capacitance of 11 gain cells (C_{gc})</td>
<td>11 * 221 a = 2.43 f Farad</td>
</tr>
<tr>
<td>Junction capacitance of dummy transmission gate (C_{dc})</td>
<td>2.96 f Farad</td>
</tr>
<tr>
<td>On resistance of transmission gate (Ron_t)</td>
<td>1.48 K ohm</td>
</tr>
<tr>
<td>Junction capacitance of transmission gate terminal 1 (C_{wbl1})</td>
<td>1.4 f Farad</td>
</tr>
<tr>
<td>Junction capacitance of transmission gate terminal 2 (C_{wbl2})</td>
<td>1.57 f Farad</td>
</tr>
</tbody>
</table>

Table 8: Values of R and C extracted from WBL circuit

The segments 1 to 5 are shorted together, so the capacitors attached to these segments are parallel to each other and are replaced by equivalent capacitor C_{t1}.

\[
C_{t1} = C_{pu} + C_{gc1} + C_{dc1} + C_{gc2} + C_{dc2} + C_{gc3} + C_{dc3} + C_{gc4} + C_{dc4} + C_{gc5} + C_{wbl1}
\]

\[
C_{t1} = C_{pu} + 5C_{gc} + 4C_{dc} + C_{wbl1} = 26.89 fF
\]

A simplified model for the above circuit of WBL is shown in figure 17. Req is the equivalent resistance of transmission gates.

Figure 17: Simplified RC chain for WBL

C_{eq} is the equivalent capacitance for segments six to ten.
Multilevel Gain Cell Arrays for Fault Tolerant VLSI Systems

\[ C_{eq} = C_{wbl2} + C_{gc} + C_{wbl1} = 5.4 \text{ fF} \]

Similarly \( C_{t11} \) is the equivalent capacitance of segment eleven.

\[ C_{t11} = C_{wbl2} + C_{gc} = 4 \text{ fF} \]

Time constant at segment eleven for the above RC chain can be calculated by the Elmore delay formula.

\[
\tau_{d11} = R_{pu}C_{t1} + (R_{pu} + R_5)C_{eq6} + (R_{pu} + R_5 + R_6 + R_7)C_{eq8} + (R_{pu} + R_5 + R_6 + R_7 + R_8)C_{eq9} \\
+ (R_{pu} + R_5 + R_6 + R_7 + R_8 + R_9)C_{eq10} \\
+ (R_{pu} + R_5 + R_6 + R_7 + R_8 + R_9 + R_{10})C_{eq11}
\]

\[ \tau_{d11} = R_{pu}C_{t1} + 20RC + 7RC_{t11} = 214.4 \text{ psec} \]

The step response of the circuit is given by first order exponential function:

\[ V_{11}(t) = V_0 \left(1 - e^{-t/\tau}\right) \]

The time taken by segment eleven voltage to go from 0 to 99% of the input voltage \((V_0 = 1.2 \text{ V})\) is given by:

\[ V(t) = 0.99V_0 \Rightarrow t = 4.6 \tau_{..} = 986.3 \text{ psec} \]

Charging time calculated from the model is compared with time given by Spectre simulation that is 999 psec. Therefore the comparison results show that the developed model is accurate and can be very helpful for the design of WBL.
Spectre simulation and modeled responses of WBL are shown in graph of figure 19.

Figure 18: Comparison of modeled and specter simulation for step input
Chapter 4: Replica Technique for MLDRAM

Chapter 4 address the problem associated with the design of access scheme for multilevel DRAM. In the coming sections the design of new memory is present in a top-down manner. First Replica approach for MLDRAM is addressed in section 4.2. Section 4.3 discusses how different modules are in new design. Design of memory controller is presented in section 4.4. Structure of replica column is discussed in section 4.6. Sections 4.7 and 4.8 discuss the implementation and working of write and read control circuits respectively. Finally, section 4.9 addresses the issues related to peripheral circuits.

4.1 Problem and Solution

Multilevel embedded dram requires a number of control signals to successfully write and read data from memory. Both read and write operation involve sequence of events with small delays between these events. In the reference design this objective was fulfilled by the digital controller. This approach was simple but conservative where whole clock cycle was dedicated each individual delay between the events. Therefore performance was degraded and resulted in long read and write access times.

The idea was to design dedicated delay line. But the problem with the conventional delay elements like chain of invertors or transmission gates is that their delay is largely affected by PVT (process, voltage, temperature) variations [24] and memory operation may become unreliable. Therefore, large margin have to be provided for reliable operation that will also degrade performance. The solution to the problem is a scheme that provides short access times, small area overhead and high reliability. This solution is achieved by using replica technique [24, 25]. Replica technique is a self-timed approach, where delay generator tracks the bit line delay across operating conditions [24].

4.2 Multilevel DRAM with Replica Technique

New MLDRAM has a storage capacity of eight kilobits (8 kbs). Block diagram of memory is shown in figure 19. Clock, reset, write and read are control signals for memory. Input data is 32-bits while output data is 16-bits. The address is 8-bits long. 2-bit reference address is used for selection of reference gain cell.

![Figure 19: Block diagram of MLDRAM with replica technique](image-url)
In current design write operation takes a single clock cycle. Whereas, read operation takes four clock cycles as shown in figure 20. In the first clock cycle for read operation, middle reference level is written to reference gain cell. During second cycle first comparison is done that gives the MSB. In third cycle depending upon the MSB, second reference level is written. Finally, in forth clock cycle the LSB is read out as result of second comparison.

**Figure 20: System level waveforms**

### 4.3 Organization of Modules

Multilevel DRAM has two storage arrays each one with capacity of 4 kbs. There are 16 sense amplifiers to do comparison for reading data. There are separate control circuits for read and write operation as shown in figure 21. Replica column forms a feedback system with read and write control circuits. There is a memory controller that handles inputs and output. Since there are two storage arrays in the current design therefore separate address and reference decoder are designed for each array. A multiplexer and enable circuit is designed to control address decoding during write and read operations. Address MSB is connected to memory controller for predecoding whereas the rest of 7-bits address is provided to address decoders as shown in figure 21.
4.3.1 Signal Flow: Write Operation

During write operation memory controller will perform predecoding to select one of the two storage arrays where data has to be written. Memory controller will enable write control circuit. Write control circuit will control the charging and discharging of replica WBL to generate the control signals for address decoding, level generation and transferring level to gain cell as shown figure 21. At a time, word by word data is written to one storage array.

4.3.2 Signal Flow: Read Operation

During the read operation memory controller predecodes the address to read back data. Depending on the predecoded address memory controller will enable address decoder for one storage array to read data and reference decoder for the other array to compare data with reference level. Memory controller will enable read control circuit.
then controls the discharging of replica RBL to generate control signals for address and reference decoding and activation of sense amplifiers. Data read during read operation is provided to memory controller which then transports it to output as shown in figure 21.

### 4.4 Memory Controller

The controller is designed to manage the read and write operations of memory. The controller is implemented in a full digital design flow i.e. coded in VHDL, simulated in ModelSim and synthesized in Synopsys. VHDL code and Synopsys script for synthesis are shown in appendix A and B respectively.

The controller, in fact is a Mealy type finite state machine, since the output depends on both the previous state and input. It has four states. State diagram is shown in figure 22. The operations carried out in each state are elaborated below:

#### 4.4.1 Idle

No operation is performed during this state. Replica read and write control circuits are disabled. The switches on write bit line for level generation are unselected. All the decoders are disabled. Pull up and pull down circuits are disabled. The sense amplifiers are also disabled. At the end of clock cycle controller checks for next state. It can either go to write data or write reference or it can remain in the idle state depending upon reset, read enable and write enable signal. This is also the reset state for memory.

#### 4.4.2 Write Data

Write operation is performed during this state and data is written to the memory. The replica write circuit is enabled that will generate all the necessary control signals for write operation. First predecoding is performed. Address most significant bit is used for this purpose. Based on the predecoded address, the address decoder for either storage array A or storage array B is enabled and the write control signal for decoder is selected. Similarly, pull up and pull down circuits are enabled for one the two storage arrays. To generate the one of the four data voltage level to be stored in a gain cell, the controller checks the 32-bit input data in pairs and selects the switches on the write bit lines to be opened for level generation. Next state for the controller could be write reference or idle state or it can remain in the write data state to write data to new address.

#### 4.4.3 Write Reference

During this state again write operation is performed. But in this case instead of data, reference level is stored in reference gain cell. The procedure for write reference is same as write data. It also involves predecoding, enabling of pull up and pull down circuits and selection of write bit line switch, but the difference is that the reference decoder is enabled instead of address decoder. For first comparison the switch to generate the middle reference level is selected. But for second comparison the selection of switch depends on the result of first comparison. If the comparison resulted in one then the switch to generate the high reference level will be selected. Otherwise it will select the low reference level switch. Next state is read. Since both data and reference voltage level are stored and are ready to be compared.
Figure 22: State diagram of memory controller
4.4.4 Read

In read state data stored in a gain cell is compare with voltage level in the reference gain cell. For this purpose the replica read circuit is enabled. Again predecoding is done to enable the address decoder for storage array containing data and reference decoder is enabled for the storage array containing reference level. Read signal is selected to control the decoder. Sense amplifiers are enabled to do comparison. The result of first comparison gives the most significant bit stored in the gain cell. Next state is write reference, where second reference level is written on basis of first compassion. After writing the second reference level the controller will again come to read state to do second comparison and that will give the least significant bit of data being read. After the second comparison the next state could be either idle or write data or write reference as shown in state diagram of figure 22.

4.5 Transistor Sizes for Storage Array

Storage array in new MDLRAM has the same architecture as in the reference design. Transistor sizes for gain cells, sense amplifiers, bit line precharge and equalizer circuit are same as reference design. However, pull up and pull down devices and transmission gates are made bigger to decrease write access time. Transistor sizes for components of storage array WBL are shown in table 9.

<table>
<thead>
<tr>
<th>Component</th>
<th>Pull up</th>
<th>Pull down</th>
<th>Transmission gate PMOS</th>
<th>Transmission gate NMOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width</td>
<td>1.4 u</td>
<td>690 n</td>
<td>1.2 u</td>
<td>690 u</td>
</tr>
<tr>
<td>Length</td>
<td>80 n</td>
<td>80 n</td>
<td>80 n</td>
<td>80 n</td>
</tr>
</tbody>
</table>

*Table 9: Transistor sizes for storage array WBL components*

4.6 Replica Column Structure

A replica column is designed to track the delay of storage array under process, voltage and temperature variations and provide optimum delays to generate the control signals necessary for read and write operations.

The structure of replica column is similar to the column of storage array so that it can successfully replicate the behavior of storage array. The area overhead of replica column is small because it is just one addition column per whole memory.

4.6.1 Replica Write Bit Line

Write bit line of replica column starts with a pull up device which consists of pMOS transistor. The pull up device for replica column is intentionally made smaller relative to the pull up devices for storage array. As a result the replica write bit line is charged slower than the write bit lines for storage array. Due to slow charging the replica write bit line will produce delay longer than that required for worst case write to precharge (predischarge) the write bit line segments of the storage array. At the bottom of replica column the bit line is connected to the pull down device which consists of NMOS transistor. The pull down device for replica column is made bigger than the pull down devices for the storage array. By having large pull down device the write bit line for replica column is discharged...
quickly. Discharging of replica write bit line is necessary to prepare it for next write operation. The write bit line for replica column is divided into twelve segments. Eleven gain cells are attached to each segment through write transistor of each gain cell. The segments of write bit line are connected together through transmission gates (switches which are always close). The transmission gates in replica column are used to maintain symmetry with storage array. The size of pMOS in transmission gate is same as that of the storage array. But the nMOS device in transmission gate for replica column is made bigger for fast discharging of write bit line. Transistor sizes for components of replica write bit line are shown in the following table 10.

<table>
<thead>
<tr>
<th>Component</th>
<th>Pull up</th>
<th>Pull down</th>
<th>Transmission gate PMOS</th>
<th>Transmission gate NMOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width</td>
<td>250 n</td>
<td>1 u</td>
<td>1.2 u</td>
<td>1 u</td>
</tr>
<tr>
<td>Length</td>
<td>200 n</td>
<td>80 n</td>
<td>80 n</td>
<td>80 n</td>
</tr>
</tbody>
</table>

**Table 10: Transistor sizes for replica WBL components**

Input and output waveforms for replica column are shown in figure 23.

4.6.2 Replica Read Bit Line

Read bit line for replica column is also connected to pull up device used for precharging. Replica RBL pull up has same size as the pull up devices associated with sense amplifiers to precharge the read bit lines for the storage array. Replica read bit line is connected to all (132) gain cells via read transistor of each gain cell. Replica read bit line is discharged only through the first gain cell called replica gain cell. The storage node of replica gain cell is connected to Vdd in order to avoid the delay of writing to replica gain cell highlighted in figure 24. While all the other gain cells in the replica column are hardwired to store zero (i.e. storage node is connected to ground) to minimize leakage. The storage node of replica gain
cell is 100 mV higher than highest storage level (1100 mV). So it will discharge the replica read bit line quickly. As a result the voltage difference between read bit lines of two gain cell to compared will be small. Therefore the sense amplifier can make wrong decision. To avoid this problem, the storage and read transistors in replica gain cell are made smaller than that of the storage array. Whereas the write transistor in replica gain cell has the same size as the write transistors in the storage array gain cells. The replica read bit line will produce control signals for read with sufficient delay to develop significant difference greater than offset of sense amplifier for worst case between read bit lines of two gain cells being compared. Transistor sizes for replica gain cell and read bit line pull up device are shown in following table 11.

<table>
<thead>
<tr>
<th>Component</th>
<th>Pull up</th>
<th>Replica gain cell RT</th>
<th>Replica gain cell ST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width</td>
<td>2 μm</td>
<td>230 nm</td>
<td>230 nm</td>
</tr>
<tr>
<td>Length</td>
<td>80 nm</td>
<td>140 nm</td>
<td>140 nm</td>
</tr>
</tbody>
</table>

**Table 11: Transistor sizes for replica gain cell and read bit line pull up device**

The replica column has two inputs and two outputs as shown in circuit diagram in figure 24.
Figure 24: Replica column circuit diagram
4.7 Write Control Circuit

Write control circuit generates the control signals for address decoding, level generation and transferring level to gain cell in order to accomplish write operation. Write control circuit is composed of replica write bit line, delay elements, inverters, transmission gates, buffers and associated logic gates. The circuit diagram of write control circuit is shown below in figure 25.

For write operation, memory controller will enable write control circuit. The signal coming from replica WBL called “replica_wbl_seg12” is passed through two inverters and a transmission gate to generate signal “seg12”. The inverters “inv1” and “inv2” are used to sharpen the transitions of signal “seg12*”. Signal “seg12” is ANDed with the inverted clock
Multilevel Gain Cell Arrays for Fault Tolerant VLSI Systems

Signal to produce signal “w1”. The enable signal “wt_replica_e” for write control circuit has a propagation delay of approximately 200 psec. Therefore the signal “w1” is delayed by an amount greater than the propagation delay of “wt_replica_e” to avoid unnecessary transitions on “rep_wt” signal. The “rep_wt” signal serves as an input to the pull up and pull devices for replica write bit line. All the control signals for write operation are generated on the basis of “rep_wt” signal. The delayed version of signal “w1” called “w2” is ORed with active low enable signal to generate signal “rep_wt”. When “rep_wt” signal goes low the replica write bit line will start charging. The charging of replica write bit line continues until the voltage at “replica_wbl_seg12” becomes higher than the switching threshold of the inverter “inv1” i.e. Vdd/2 = 600mV, causing the inverter output to change. Therefore, “rep_wt” signal makes a low to high transition as shown in figure 26. As a result replica write bit line stops charging and begins to discharge. In this manner replica WBL and write control circuit form a feedback loop. Discharging of replica write bit line is necessary to prepare for next write operation. Transmission gate “tg1” is utilized to cut the path of control signals when “rep_wt” goes from low to high. This is done to make sure that pre(dis)charging and switch control signals do not change because of replica WBL discharging.

Signal “seg12” is passed through delay element “d2” and inverters shown in figure 25 to generate signal “pu” and “pd”. Signal “pu” is ORed with “pu_e” to generate “pull up” signal that controls the precharging of write bit line segments of the storage array shown in figure 26. Signal “pd” is ANDed with “pd_e” to generate signal “pull down” that controls the predischarging of write bit line segments of the storage array. Signal “pu_e” and “pd_e” are generated by memory controller for both storage arrays during write data and write reference states. Signal “pd” and compliment of signal “rep_wt” are ORed to generate “sw_rep” signal. Since signal “sw_rep” serves as an input to switch multiplexers. Therefore, signal “sw_rep” is driven by buffer “b2”.

Switch multiplexer shown in figure 25 is used to provide the control signal that will turn off one switch on the write bit line of storage array for level generation. Switch multiplexer is composed of a transmission gate, nMOS transistor and an inverter. The selection signals for multiplexer are “cp” and “cn”. These selection signals are generated by memory controller depending upon 32 bit input data. For a selected switch, signal “sw_rep” is passed by multiplexer. Whereas, for unselected switches “gnd!” signal is passed. Switch multiplexer generates signal for pMOS device of switches on write bit line. While an inverter is used to provide the signal for nMOS device in write bit line switch.

Delay element “d2” is inserted in the path of “sw_rep” signal to make sure that the control signals for pre(dis)charging and level generation are non-overlapping. Otherwise, wrong level will be generated that would ultimately result in writing wrong data to memory.

The control signal for decoder is generated by ORing signal “w1” and “w2”. Address decoder provides one hot decoded address shown in figure 26. The decoded address is ORed with “wt_replica_e” signal to generate control signal for write word line. Buffer “b2” is used to drive one input of OR gate for each word line.

Waveforms of control signals for write operation are shown in figure 26. “rep_wt” signal controls charging and discharging of replica write bit line. During write operation, “decoder_clk” signal serves as control signal for address decoder. “Pull up” and “pull down”
control the precharging and predischarging of write bit lines respectively. Signals “switch_p” and “switch_n” control the transmission gates on write bit lines for level generation. Level transferring to gain cell is controlled by “write word line” signal.

Figure 26: Write control signal waveforms
4.8 Read Control Circuit

Read control circuit is responsible for generating control signals necessary to accomplish read operation. Read control circuit is constructed from replica read bit line, buffers, inverters, logic gates and delay elements. Circuit diagram for read control circuit is shown below in figure 27.

As shown in figure 27 “replica_rbl” signal is the input of read control circuit. Furthermore, “rep_rd” generated by read control circuit is dependent on “replica_rbl” signal. Whereas, “rep_rd” signal is an input for replica column, where it is connected to read word line of replica gain cell and control port of replica read bit line pull up device. In this way a feedback loop is formed that controls the charging and discharging of replica RBL and duty cycle of “rep_rd” signal. Input signals “rd_replica_e” and “san_e” are generated by memory controller during read state. “rd_replica_e” signal is used to enable read control circuit and PMOS devices in sense amplifier. Signal “san_e” serves as an enable signal for NMOS devices in sense amplifier. Delays “d1” and “d2” are used to avoid unnecessary transition on “rep_rd” signal due to propagation delay of “rd_replica_e” signal. However, delays “d4, d5”
Multilevel Gain Cell Arrays for Fault Tolerant VLSI Systems

"d6" are used to match the delay of "rep_rd" signal. Delay "d3" is used to generate the control signal for address and reference decoders. Inverter "inv1" and "inv2" are used to make the transitions sharp. Inverter "inv4" is used to generate active high RWL signal as shown in figure 28. Furthermore, buffers are used for driving purpose.

"replica_rbl" signal is passed through inverters "inv1", "inv3" and delay element "d3" to produce signal "r3". Delay "d3" is used to match the delay of "rep_rd" signal. Signals "r3" and "rep_rd" are ANDed together to produce "r4" signal. Signal "rep_rd" is ORed with signal "r4" to produce signals "san" and "sap". Signal "san" is ANDed with delayed version of signal "san_e" to produce "sense_amplifier_n". Signal "sap" is ORed with delayed version of signal "re_replica_e" to produce "sense_amplifier_p" signal.

For read operation, memory controller enables read control circuit. Read control circuit will turn on the pull up devices to precharge replica read bit line and storage array read bit lines to Vdd shown in waveforms of figure 28. In the mean while address and reference decoder will provide one hot decoded addresses for the gain cells to be compared. Soon after the address decoding is done, read bit lines of gain cells to be compared and replica read bit line are allowed to discharge simultaneously. Since, two gain cells being compared are having different voltage levels at storage node. Therefore, the read bit line for one gain cell will discharge faster than the other one. As a result, a voltage difference is developed between the read bit lines. Finally, sense amplifier is enabled to make comparison.

This circuit controls the width of the pulse to discharge read bit lines and the right time to kick in the sense amplifier for accurate read operation. For this purpose, the discharging of read bit lines is controlled to generate sufficient voltage difference between the read bit lines.
Figure 28: Read control signal waveforms
4.9 Peripheral Circuits

Address and reference decoders and enable circuits and multiplexers for decoders constitute peripherals of multilevel DRAM.

4.9.1 Address and Reference Decoders

In MLDRAM each storage array has 128 word for data and four words for reference voltage levels. For reference words 2 to 4 NAND decoder is designed to provide one hot code. The schematic of 2 to 4 NAND decoder is shown in figure 29. In order to decode 128 data addresses, a 7 to 128 NAND decoder is designed.

![Figure 29: 2 to 4 NAND decoder]
4.9.2 Multiplexer and Enable Circuit

Multiplexer and enable circuit is used to provide control signal for both address and reference decoder. During write operation “wt” signal coming from write control signal is selected. While reading “rd” provided by read control circuit is selected by multiplexer. Enable signal is used to control decoders during idle read and write states. The schematic if multiplexer and enable circuit is shown in figure 30.

![Figure 30: Multiplexer and enable circuit](image-url)
Chapter 5: Simulation Setup and Results

5.1 Simulation Setup

In order to check that memory is working properly simulation are performed. The following figure 31 shows the simulation setup.

A simple memory model is developed in MatLab shown in appendix C that generates the stimuli and expected responses. Generated stimuli and expected responses are read into Cadence by file readers. A test bench is designed in Cadence, where memory is simulated for stimuli in Spectre simulator. The result of Spectre simulation are compared with expected response by an OceanScript given in appendix D and result are reported to a text file.
5.2 Simulation Scenarios

Memory is simulated for two different scenarios and results are accurate for both cases. There are no read failures.

First Scenario
- Random data
- Random addresses

Second Scenario
- Random data
- Increasing order addresses

5.3 Simulation Results

Following are the simulation results for access times and energy consumption of MLDRAM.

5.3.1 Write and Read Access Times

Nominal Operating Conditions

Under normal operating conditions i.e. process corner = tt, voltage = 1.2 V and temperature = 27°C following are the access times shown in table 12.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Access Time</th>
<th>Clock Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write</td>
<td>3 nsec</td>
<td>1</td>
</tr>
<tr>
<td>Read</td>
<td>12 nsec</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 12: Access times for nominal operating conditions

Memory is operated at a clock frequency of 333.33 MHz.

Worst Case operating Conditions

For worst case operating conditions i.e. process corner = ss, voltage = 1.08 V and temperature = 85°C following are the access times shown in table 13.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Access Time</th>
<th>Clock Cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write</td>
<td>5 nsec</td>
<td>1</td>
</tr>
<tr>
<td>Read</td>
<td>20 nsec</td>
<td>4</td>
</tr>
</tbody>
</table>

Table 13: Access times for worst case conditions

Memory is operated at a clock frequency of 200 MHz.
5.4 Energy Consumption

Energy consumption per bit under normal operating conditions for multilevel gain cell DRAM is shown in table 14.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Energy per bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write</td>
<td>359 fJ</td>
</tr>
<tr>
<td>Read</td>
<td>1.6 pJ</td>
</tr>
</tbody>
</table>

Table 14: Energy consumption per bit

The reason for high energy consumption for read is that read operation involves two write operations to write reference voltage levels and two sense operations to do comparison.
Chapter 6: Variation Tolerance

The replica technique not only generates the control signals for read and write operation with optimum delay but also tracks the variations in storage array across operating conditions. Following sections discuss the variation tolerance of current design during write and read operations.

6.1 Write Variation Tolerance

6.1.1 Process Variations

Monte Carlo analysis is performed to show the process variation tracking for write operation. During write operation replica WBL charging tracks the delay variation in storage array WBL. Delay of replica WBL is slightly larger than the delay of storage WBL to ensure correct operation without degrading the performance.

Figure 32 shows the distribution of delay difference between replica and storage WBLs. Simulation is done under normal operating conditions and on top of that with-in-die process variations are applied. Results are reported for 1000 Monte Carlo runs.

As shown in figure 32 the mean value is 92.6 psec and standard deviation is 47.8 psec which are quite small. Hence replica WBL successfully tracks the process variations of array storage WBL.
6.1.2 Voltage Variations

Parametric analysis is performed to show how replica WBL tracks the variations in supply voltage during write operation.

Simulation is performed for typical process corner at a temperature of 27 °C and supply voltage is varied from -10% to +10% of nominal supply voltage i.e. 1.2 V. It can be seen in the figure 33 that the delay of replica and storage WBLs is decreasing with the increase of supply voltage.

![Delay variation with respect to supply voltage](image)

**Figure 33: Voltage variation tracking for write operation**

Therefore it can be said that replica technique tracks the variation in supply voltage during write operation.
6.1.3 Temperature Variations

Parametric analysis is performed to show that replica WBL can track the variations in delay due to temperature changes.

Simulation is performed with typical process corner at supply voltage of 1.2 V and temperature is varied from -25 °C to 125 °C. Simulation result shown in figure 34 indicate that delay of both replica and storage WBL increases with the increase of temperature.

![Temperature variation tracking for write operation](image)

Hence it is concluded that replica WBL also tracks the delay of storage array WBL in the presence of temperature variations.
6.2 Read Variation Tolerance

6.2.1 Process Variations

Monte Carlo analysis is performed to show the process variation tracking for read operation. During read operation replica RBL discharging tracks the delay variation in storage array RBL. Delay of sense amplifier activation is slightly larger than the delay of replica RBL without degrading the performance. It is necessary to ensure that sufficient voltage difference (greater than the offset to sense amplifier) is developed between the RBLs of two gain cell to be compared. So that the sense amplifier is kicked in at the right time to make decision.

Figure 35 shows the distribution of the delay difference between sense amplifier activation and replica RBL. Simulation is done under normal operating conditions and on top of that with-in-die process variations are applied. Results are reported for 1000 Monte Carlo runs.

As shown in figure 35 the mean value is 54.4 psec and standard deviation is 15 psec which are quite small. Hence replica RBL successfully tracks the process variations of array storage RBL.
6.2.2 Voltage Variations

Parametric analysis is performed to show how replica RBL tracks the variations in supply voltage during read operation.

Simulation is performed for typical process corner at a temperature of 27 °C and supply voltage is varied from -10% to +10% of nominal supply voltage i.e. 1.2 V. It can be seen in the figure 36 that the delay of sense amplifier activation and replica RBL is decreasing with the increase of supply voltage.

![Delay variation with respect to supply voltage](image)

*Figure 36: Voltage variation tracking for read operation*

Therefore it can be said that replica technique tracks the variation in supply voltage during read operation.
6.2.3 Temperature Variations

Parametric analysis is performed to show that replica RBL can track the variations in delay due to temperature changes.

Simulation is performed with typical process corner at supply voltage of 1.2 V and temperature is varied from -25°C to 125°C. Simulation result shown in figure 37 indicate that delay of both sense amplifier activation and replica RBL increases with the increase of temperature.

![Figure 37: Temperature variation tracking for read operation](image)

Hence it is concluded that replica RBL also tracks the delay of storage array RBL in the presence of temperature variations.
Chapter 7: Comparison with Reference Design

In this chapter of a comparison of newly designed multilevel DRAM with replica technique with the reference design is presented. Two designs are compared for storage array, control mechanism. Improvements in the access times and operating frequency are reported.

7.1 Storage Array

The basic architecture of storage array is same for both the memories. Both the designs have a three transistor mixed nMOS pMOS gain cell as a unit storage element capable of storing two bits. Both the memories have a two similar storage arrays each with capacity of four kilobits (4 kbs). So the total storage is eight kilobits (8 kbs). Each storage array has 132 words, with word length of 32 bits. There are 128 words for storing data whereas four words are used to store reference voltage levels in each storage array. The read bit lines of storage arrays are connected to the terminals of sense amplifiers. Since there are 16 gain cells per word line so both the design have 16 sense amplifiers for comparing voltages. The write bit lines are divided into twelve segments connected through eleven transmission gates. Each segment of write bit line has eleven gain cells attached to it. The transistor sizes for sense amplifier, gain cell and bit line equalizer are same for both designs. However, for new design, the size of pull up, pull down and nMOS transistors in transmission gates of write bit line are different to speed up the write operation.

7.2 Write and Read Access Method

In reference design the write operation is performed by using charge sharing approach to generate the required voltage level and then transferring the level to storage node of gain cell. The multilevel DRAM with replica technique utilizes the same charge sharing approach because of area efficiency. Since it uses the hardware that is already there in the form of bit line segments.

The read operation in the reference design is performed in sequential fashion. The read operation requires two successive comparisons to read back two bit data. The first comparison provides the most significant bit while the second comparison gives the least significant bit of data. For new design the same approach is adopted due of its area effectiveness.

7.3 Control Mechanism

Control mechanism plays very important role in multilevel gain cell DRAMs for access time improvement. As discussed earlier both the designs have similar array architecture and same methods for write and read operations. Therefore, it is the control mechanism of current design that actually led to better performance.

The control mechanism for reference design based on a digital controller. This controller is in fact a finite state machine that controls the read and write operations. The peripherals such as address decoder are also included in the digital controller. In reference design finite state machine has eleven states. Hence it is a conservative design that degraded the
The current design utilizes a sense and trigger scheme to control memory operation. This new design still has a memory controller implemented in digital design flow. However, the new finite state machine is very simple and it has only four states. The memory controller activates different modules of design for write and read operation. The edge of new design is the replica technique. For this purpose an additional column is designed that replicates the behavior of storage array. This replica column not generates the control signals with optimum delay but also tracks the PVT variations thus performance is improved.

### 7.4 Clock Frequency

The operating frequency for multilevel gain cell DRAM is limited by the operation which takes more time. For both the designs it is write operation that takes more time and time required to precharge and predischage write bit line segments is most dominant contribution. MLDRAM with replica technique has higher operating frequency because the pre(dis)charging time is significantly reduced by increasing the width of pull up and transmission gate transistors of write bit line.

Table 15 shows the frequency comparison of reference and current design.

<table>
<thead>
<tr>
<th>Design</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>100 MHz</td>
</tr>
<tr>
<td>Current</td>
<td>333.33 MHz</td>
</tr>
</tbody>
</table>

Table 15: Frequency comparison

### 7.5 Access Time

As mentioned earlier that the main drawbacks of the reference design are long write and read access times. Hence the utilization of replica technique resulted in simple finite state machine controller with just four states that allowed to provide the right amount of delay required and performance is increased.

Access times and clock cycles for current and reference design are reported in the table 16.

<table>
<thead>
<tr>
<th>Design</th>
<th>Write</th>
<th>Read</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Access Time</td>
<td>Clock Cycles</td>
</tr>
<tr>
<td>Reference</td>
<td>30 nsec</td>
<td>3</td>
</tr>
<tr>
<td>Current</td>
<td>3 nsec</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 16: Access time comparison

### 7.6 Improvement

Comparison of reference and current designs shows that there is an improvement of 90% in both read and write access time. The reason for that much improvement is that the reference design was a conservative approach where a complete clock cycle is dedicated to a small amount of delay that degraded the performance. The new design employing replica
technique improves the performance by providing the required amount of delay with further compromising the reliability. This new approach led to the design of very simple finite state machine and the states are reduced form eleven to four. Although the new design is not subject to layout but it is obvious that the new design will result in less area. Because in reference design significant area of memory is occupied by digital controller which is now reduced.
Chapter 8: Conclusion

Embedded memories are dominant part of area, power and cost of modern DSP systems. Multilevel gain cell array provide an alternative to technology scaling in order to increase the storage density at the cost reduced reliability. Multilevel gain cell array has area less than SRAM and at the same time it is logic compatible. Furthermore due to increasing process variations and higher defect level fault tolerant VLSI designs with multilevel gain cell memories would result in small area, reduced cost and better performance.

The focus of this work is to design multilevel read and write scheme to improve access time. During this thesis extensive literature study is done on multilevel gain cell arrays and different method to write and read data. Detailed analysis of reference design is done for read and write operation. For this purpose, parasitic resistance and capacitance are extracted from the layout of reference design. First the reference design was simulated with parasitic extracted view to estimate the time required to write data. Furthermore, write operation was performed for different process corners, supply voltage, temperatures and different data (voltage levels) to determine the worst case. Monte Carlo analysis is performed to estimate the delay for with-in-die process variation and error in generated voltage.

Analytical model for write bit line is developed to study the sizing effect of transistors of WBL components on precharging delay. Simulations based on analytical model and analysis of write operation suggests that the write access time can be reduced by precharging the write bit line segments quickly. This is accomplished by increasing the width of pull up and transmission gate transistor. As a result the write access is significantly improved which allowed to operate memory at higher frequency.

The reference design is also analyzed for read operation. Monte Carlo analysis is done to find out the offset voltage of sense amplifier. Read operation in multilevel gain cell DRAM is composed of successive comparisons where data voltage level is compared with two reference voltage levels. Read operation is performed to study the effect of different data and reference voltage levels on the read access time.

The bottleneck of reference design is the long write and read access time. So the idea is to improve the access time by designing the dedicated delay lines that provide the required delay. Therefore a number of delay elements are studied. However, the problem with conventional delay elements such as chain of invertors, transmission gates and Schmitt trigger based delays is that there delay does not vary in accordance with the delay of storage due to increasing process variations in modern deep 100 nm CMOS process. Therefore, sufficient timing margins have to be provided for reliable operation that will also degrade performance.

The solution to the problem is replica technique which is a self-timed approach. Where delay generators track the bit line delay across operating conditions. Previously proposed different version replica techniques are studied. All these approaches deal with reducing the bit line swing and activating the sense amplifier as soon as possible to save power and improve read access time. However, the utilization of replica technique to improve the
access the time of multilevel DRAMs has not been exploited yet. Therefore, a new replica technique is proposed that only with improves the read access time but also deals with generation of control signals to improve write access time.

To implement the replica technique one addition column called replica column is design. The structure of replica column is similar to the column of storage array. The designed replica column serves two purposes, first provides the required amount of delay and second it tracks the delay of storage array for process, voltage and temperature variations. Write and read control circuits designed from logic gates together with replica column form a close loop feedback control to generate the control signals necessary to perform write and read operation.

A memory control is design in full digital design flow. This memory controller is Mealy type finite state machine which is coded in VHDL, simulated in ModelSim and synthesized in Synopsys DesignVision. The memory controller is responsible for controlling write and read control circuits, address decoders and multiplexers during write and read operations. 2 to 4 NAND decoders are designed to for identification of reference gain cells. Whereas, 7 to 128 NAND decoders perform address decoding for storage cells. Furthermore, some multiplexers are design to deliver different control signals to decoder for write and read operation.

Eight kilobits multilevel gain cell DRAM with replica technique is designed in UMC 90 nm CMOS technology. In order to verify the operation of memory simulation are performed. A simple memory model is designed in MatLab that generates stimuli and expected responses which are read into Cadence. Memory is simulated with the stimuli and simulation results are compared with expected responses with the aid of an OceanScript. The design is verified for two different scenarios. The access time and energy consumption of memory is reported. Finally Monte Carlo analysis is performed to show the effectiveness of replica technique in the presence of process voltage and temperature variations.

In short, multilevel gain cell array with replica technique not only improved the access time by 90 % as compared to reference design but also made the design variation tolerant.
Chapter 9: Future Work

The layout of new multilevel gain cell DRAM with replica technique needs to be done. The layout of gain cells, sense amplifier and bit line equalizer is already available from reference design. Since the size of pull up device and transistor of transmission gates are changed their layout needs to be drawn.

Layout of replica column write and read control circuits is to be drawn. For address decoders, the layout of 2 to 4 decoder can serve as a basic building block for 7 to 128 decoder. Although the layout of logic gates is available from standard cell library yet they need to be redrawn to fit with the words of storage array. The layout of memory controller can be designed in Cadence SoC Encounter with the aid TCL script. The floor plan of reference design is good and can be kept the same way.

After the layout of new memory is complete, parasitic resistance and capacitance can be extracted. Memory can be simulated with the av_extracted view to study the effect of parasitics on write and read access times. Furthermore, design can be analyzed by Monte Carlo simulation to study the variation tolerance. Design can be simulated to read failure analysis to find out the range of reliable operation and to further push the design toward better performance (decreased access times) at the cost more errors (reduced reliability).

Write access time can be further reduced by the generating all the data and reference voltage levels once and then transporting these voltage levels to gain cells with the aid of analog switches and transmission gates. For example an analog to digital converter can be used to generate the voltage levels. Therefore, such an approach would be useful for application that requires fast access and area overhead can be afforded. However, if a design has area limitations then area efficient charge sharing approach can be adopted but instead of a single pull up device at the top, multiple pull up devices can be employed between the segments of write bit line to speed up the precharging of write bit line.

Read operation in current design is performed in sequential manner where data voltage level is successively compared with two reference voltage levels. Read access time can be reduced by parallel sensing meaning that data voltage level can be compared with all the three reference voltage levels at the same time. It can be accomplished by using three sense amplifiers per gain cell and copying of gain cell current with aid of current mirrors and applying to terminals of sense amplifiers for comparison. The outputs of sense amplifier can be encoded to provide the final data value. An analog to digital converter can also be used to improve read access time. However, such parallel sensing techniques would increase the area of design significantly.

Furthermore, a new technique can be design to mitigate the effect of with-in-die process variations. In this way not only access time can be reduced but also design will be energy efficient.

It would be to fabricate the chip after layout and do the post silicon measurements to make multilevel gain cell arrays to be integrated with modern VLSI SoC designs.
References


Muhammad Umer Khalid
Telecommunication Circuits Laboratory, EPFL
Appendix

A. VHDL Code for Memory Controller

VHDL code for memory controller (finite state machine) from which hardware is generated.

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
use ieee.numeric_std.all;
use work.dig_pkgs.all;

entity digital_ctrl_n is
port ( 
    clk            : in std_logic; -- system clock
    reset          : in std_logic; -- asynchronous reset
    we             : in std_logic; -- write enable
    re             : in std_logic; -- read enable
    data_in        : in std_logic_vector (word_size-1 downto 0); -- 32 bit input
    rbl_a          : in std_logic_vector (column_size-1 downto 0); -- 16 bit read data
    rbl_b          : in std_logic_vector (column_size-1 downto 0); -- 16 bit read data
    address_msb    : in std_logic; -- msb of address to activate array A or B
    data_out       : out std_logic_vector (column_size-1 downto 0); -- 16 bit output data
    ref_decoder_ae : out std_logic; -- reference decoder enable
    ref_decoder_be : out std_logic; -- reference decoder enable
    add_decoder_ae : out std_logic; -- address decoder enable
    add_decoder_be : out std_logic; -- address decoder enable
    -- level generation switch controls
    cp4a           : out std_logic_vector (column_size-1 downto 0);
    cn4a           : out std_logic_vector (column_size-1 downto 0);
    cp4b           : out std_logic_vector (column_size-1 downto 0);
    cn4b           : out std_logic_vector (column_size-1 downto 0);
    cp5a           : out std_logic_vector (column_size-1 downto 0);
    cn5a           : out std_logic_vector (column_size-1 downto 0);
    cp5b           : out std_logic_vector (column_size-1 downto 0);
    cn5b           : out std_logic_vector (column_size-1 downto 0);
    cp6a           : out std_logic_vector (column_size-1 downto 0);
    cn6a           : out std_logic_vector (column_size-1 downto 0);
    cp6b           : out std_logic_vector (column_size-1 downto 0);
    cn6b           : out std_logic_vector (column_size-1 downto 0);
    cp7a           : out std_logic; -- middle ref level
    cn7a           : out std_logic; -- middle ref level
    cp7b           : out std_logic; -- middle ref level
    cn7b           : out std_logic; -- middle ref level
    cp8a           : out std_logic_vector (column_size-1 downto 0);
    cn8a           : out std_logic_vector (column_size-1 downto 0);
    cp8b           : out std_logic_vector (column_size-1 downto 0);
    cn8b           : out std_logic_vector (column_size-1 downto 0);
    cp9a           : out std_logic_vector (column_size-1 downto 0);
    cn9a           : out std_logic_vector (column_size-1 downto 0);
    cp9b           : out std_logic_vector (column_size-1 downto 0);
    cn9b           : out std_logic_vector (column_size-1 downto 0);
    cp10a          : out std_logic_vector (column_size-1 downto 0);
    cn10a          : out std_logic_vector (column_size-1 downto 0);
    cp10b          : out std_logic_vector (column_size-1 downto 0);
);
Multilevel Gain Cell Arrays for Fault Tolerant VLSI Systems

cn10b : out std_logic_vector (column_size-1 downto 0);
-- replica array enable
wt_replica_e : out std_logic;
rd_replica_e : out std_logic;-- same for sap_e
-- pull up pull down enable
pu_e : out std_logic;
pd_e : out std_logic;
pu_be : out std_logic;
pd_be : out std_logic;
-- sense amplifier enable
san_e : out std_logic;-- sap_e same as read replica_enable
end digital_ctrl_n;

architecture mealy of digital_ctrl_n is

type state is (idle,wst,rst1,rst2); -- 4 states of fsm

signal data_out_int : std_logic_vector (column_size-1 downto 0);
signal pstate,nstate : state;
signal psecond_comparison,nsecond_comparison : std_logic;
signal cplevel1a : std_logic_vector (column_size-1 downto 0);
signal cnlevel1a : std_logic_vector (column_size-1 downto 0);
signal cplevel2a : std_logic_vector (column_size-1 downto 0);
signal cnlevel2a : std_logic_vector (column_size-1 downto 0);
signal cplevel3a : std_logic_vector (column_size-1 downto 0);
signal cnlevel3a : std_logic_vector (column_size-1 downto 0);
signal cplevel4a : std_logic_vector (column_size-1 downto 0);
signal cnlevel4a : std_logic_vector (column_size-1 downto 0);
signal cpref1a : std_logic_vector (column_size-1 downto 0);
signal cnref1a : std_logic_vector (column_size-1 downto 0);
signal cpref2a : std_logic;
signal cnref2a : std_logic;
signal cpref3a : std_logic_vector (column_size-1 downto 0);
signal cnref3a : std_logic_vector (column_size-1 downto 0);
signal cpref2b : std_logic;
signal cnref2b : std_logic;
signal cplevel1b : std_logic_vector (column_size-1 downto 0);
signal cnlevel1b : std_logic_vector (column_size-1 downto 0);
signal cplevel2b : std_logic_vector (column_size-1 downto 0);
signal cnlevel2b : std_logic_vector (column_size-1 downto 0);
signal cplevel3b : std_logic_vector (column_size-1 downto 0);
signal cnlevel3b : std_logic_vector (column_size-1 downto 0);
signal cplevel4b : std_logic_vector (column_size-1 downto 0);
signal cnlevel4b : std_logic_vector (column_size-1 downto 0);
signal cpref1b : std_logic_vector (column_size-1 downto 0);

end digital_ctrl_n;
signal cnref1b : std_logic_vector (column_size-1 downto 0);
signal cpref3b : std_logic_vector (column_size-1 downto 0);
signal cnref3b : std_logic_vector (column_size-1 downto 0);
signal adecode_a : std_logic;
signal rdecode_a : std_logic;
signal adecode_b : std_logic;
signal rdecode_b : std_logic;
signal wt_rep_e : std_logic;
signal rd_rep_e : std_logic;
signal pu_ea : std_logic;
signal pd_ea : std_logic;
signal pu_eb : std_logic;
signal pd_eb : std_logic;
signal san : std_logic;
begin
-- output data
data_out <= data_out_int;
-- level switches
cp4a <= cplevel1a;
cn4a <= cnlevel1a;
cp6a <= cplevel2a;
cn6a <= cnlevel2a;
cp8a <= cplevel3a;
cn8a <= cnlevel3a;
cp10a <= cplevel4a;
cn10a <= cnlevel4a;
cp5a <= cpref1a;
cn5a <= cnref1a;
cp9a <= cpref3a;
cn9a <= cnref3a;
cp7a <= cpref2a;
cn7a <= cnref2a;
cp7b <= cpref2b;
cn7b <= cnref2b;
cp4b <= cplevel1b;
cn4b <= cnlevel1b;
cp6b <= cplevel2b;
cn6b <= cnlevel2b;
cp8b <= cplevel3b;
cn8b <= cnlevel3b;
cp10b <= cplevel4b;
cn10b <= cnlevel4b;
cp5b <= cpref1b;
cn5b <= cnref1b;
cp9b <= cpref3b;
cn9b <= cnref3b;
-- decoders
add_decoder_ae <= adecode_a;
add_decoder_be <= adecode_b;
ref_decoder_ae <= rdecode_a;
ref_decoder_be <= rdecode_b;
-- replica wbl and rbl
wt_replica_e <= wt_rep_e;
rd_replica_e <= rd_rep_e;
-- sense amp
san_e <= san;
-- pull up and pull down
pu_ae <= pu_ea;
pu_be <= pu_eb;
pd_ae <= pd_ea;
pd_be <= pd_eb;
second_comparison_reg: process(reset, clk)
begin
if reset = '0' then
  psecond_comparison <= '0';
elsif clk'event and clk = '1' then
  psecond_comparison <= nsecond_comparison;
end if;
end process second_comparison_reg;

combinatorial : process
  (address_msb, reset, data_in, we, re, pstate, psecond_comparison, rbl_a, rbl_b)
begin
  -- assigning default values to avoid unintentional latches
  nstate <= pstate;
  nsecond_comparison <= psecond_comparison;
  -- by default data out will come from array b
  data_out_int <= rbl_a;
  -- by default all connector on wbl are closed (gates connected to vdd and gnd instead of replica signals)
  cplevel1a <= (others => '1');
  cplevel2a <= (others => '1');
  cplevel3a <= (others => '1');
  cplevel4a <= (others => '1');
  cpref1a <= (others => '1');
  cpref3a <= (others => '1');
  cnlevel1a <= (others => '0');
  cnlevel2a <= (others => '0');
  cnlevel3a <= (others => '0');
  cnlevel4a <= (others => '0');
  cnref1a <= (others => '0');
  cnref3a <= (others => '0');
  cpref2a <= '1';
  cnref2a <= '0';
  cpref2b <= '1';
  cnref2b <= '0';
  cplevel1b <= (others => '1');
  cplevel2b <= (others => '1');
  cplevel3b <= (others => '1');
  cplevel4b <= (others => '1');
  cpref1b <= (others => '1');
  cpref3b <= (others => '1');
  cnlevel1b <= (others => '0');
  cnlevel2b <= (others => '0');
  cnlevel3b <= (others => '0');
  cnlevel4b <= (others => '0');
  cnref1b <= (others => '0');
  cnref3b <= (others => '0');
  -- replica wbl and rbl disabled
  wt_rep_e <= '1';
Multilevel Gain Cell Arrays for Fault Tolerant VLSI Systems

rd_rep_e <= '1';
-- pull up and pull down disabled
pu_ea <= '1';
pu_eb <= '1';
pd_ea <= '0';
pd_eb <= '0';
-- disable sense amps
san <= '0'; -- sap disabled by rd_rep_e
-- decoders
adecode_a <= '0';
adecode_b <= '0';
rdecode_a <= '0';
rdecode_b <= '0';
case pstate is
-- idle state
when idle =>
nsecond_comparison <= '0';
if we = '0' and re = '0' then
nstate <= idle;
elsif we = '1' and re = '0' then
nstate <= wst;
elsif we = '0' and re = '1' then
nstate <= rst1;
else
nstate <= idle;
end if;
-- write state
when wst =>
wt_rep_e <= '0'; -- enable replica wbl
if address_msb = '0' then
  adecode_a <= '1'; -- enable address decoder a
  pu_ea <= '0'; -- enable pull up for array a
  pd_ea <= '1'; -- enable pull down for array a
  for i in 0 to column_size-1 loop
    if data_in(2*i+1) = '0' and data_in(2*i) = '0' then
      cplevel1a(i) <= '0'; -- connect to replica signal
      cnlevel1a(i) <= '1'; -- connect to replica signal
    elsif data_in(2*i+1) = '0' and data_in(2*i) = '1' then
      cplevel2a(i) <= '0'; -- connect to replica signal
      cnlevel2a(i) <= '1'; -- connect to replica signal
    elsif data_in(2*i+1) = '1' and data_in(2*i) = '0' then
      cplevel3a(i) <= '0'; -- connect to replica signal
      cnlevel3a(i) <= '1'; -- connect to replica signal
    elsif data_in(2*i+1) = '1' and data_in(2*i) = '1' then
      cplevel4a(i) <= '0'; -- connect to replica signal
      cnlevel4a(i) <= '1'; -- connect to replica signal
    end if;
  end loop;
else
  adecode_b <= '1'; -- enable address decoder b
Multilevel Gain Cell Arrays for Fault Tolerant VLSI Systems

pu_eb <= '0'; -- enable pull up for array b
pd_eb <= '1'; -- enable pull down for array b

for i in 0 to column_size-1 loop
    if data_in(2*i+1) = '0' and data_in(2*i) = '0' then
cplevel1b(i) <= '0'; -- connect to replica signal
cnlevel1b(i) <= '1'; -- connect to replica signal
elsif data_in(2*i+1) = '0' and data_in(2*i) = '1' then
cplevel2b(i) <= '0'; -- connect to replica signal
cnlevel2b(i) <= '1'; -- connect to replica signal
elsif data_in(2*i+1) = '1' and data_in(2*i) = '0' then
cplevel3b(i) <= '0'; -- connect to replica signal
cnlevel3b(i) <= '1'; -- connect to replica signal
elsif data_in(2*i+1) = '1' and data_in(2*i) = '1' then
cplevel4b(i) <= '0'; -- connect to replica signal
cnlevel4b(i) <= '1'; -- connect to replica signal
end if;
end loop;
end if;

-- check for next state
if we = '0' and re = '0' then
    nstate <= idle;
elsif we = '1' and re = '0' then
    nstate <= wst;
elsif we = '0' and re = '1' then
    nstate <= rst1;
else
    nstate <= idle;
end if;

-- read state 1
when rst1 =>
    wt_rep_e <= '0'; -- enable replica wbl

if address_msb = '1' then
    rdecode_a <= '1'; -- enable ref decoder a
    pu_ea <= '0'; -- enable pull up for array a
    pd_ea <= '1'; -- enable pull down for array a
    if psecond_comparison = '0' then
        cpref2a <= '0'; -- connect to replica signal
cnref2a <= '1'; -- connect to replica signal
    else
        for j in 0 to column_size-1 loop
            if rbl_a(j) = '1' then
                cpref3a(j) <= '0';
cnref3a(j) <= '1';
            else
                cpref1a(j) <= '0';
cnref1a(j) <= '1';
        end loop;
    end if;
else
    for j in 0 to column_size-1 loop
        if rbl_a(j) = '1' then
            cpref3a(j) <= '0';
cnref3a(j) <= '1';
        else
            cpref1a(j) <= '0';
cnref1a(j) <= '1';
    end loop;
end if;
Multilevel Gain Cell Arrays for Fault Tolerant VLSI Systems

end if;
end loop;
end if;

else

rdecode_b <= '1'; -- enable ref decoder b
pu_eb <= '0'; -- enable pull up for array b
pd_eb <= '1'; -- enable pull down for array b

if psecond_comparison = '0' then

cpref2b <= '0';
cnref2b <= '1';
else

for j in 0 to column_size-1 loop

if rbl_b(j) = '1' then

cpref3b(j) <= '0';
cnref3b(j) <= '1';
else

cpref1b(j) <= '0';
cnref1b(j) <= '1';
end if;
end loop;
end if;

end if;

-- next state is read state 2
nstate <= rst2;

-- read state 2
when rst2 =>

rd_rep_e <= '0'; -- enable replica rbl -- same for sap
san <= '1'; -- enable sense amplifier

if address_msb = '1' then

rdecode_a <= '1'; -- enable ref decoder a
adecode_b <= '1'; -- enable address decoder b

if psecond_comparison = '0' then

nsecond_comparison <= '1';
data_out_int <= rbl_a;

-- next read state 1
nstate <= rst1;
else


data_out_int <= rbl_a;
nsecond_comparison <= '0';

--check for next state
if we = '0' and re = '0' then
  nstate <= idle;
elsif we = '1' and re = '0' then
  nstate <= wst;
elsif we = '0' and re = '1' then
  nstate <= rst1;
else
  nstate <= idle;
end if;
else
  rdecode_b <= '1'; -- enable ref decoder b
  adecode_a <= '1'; -- enable address decoder a
    if psecond_comparison = '0' then
      nsecond_comparison <= '1';
data_out_int <= rbl_b;
  -- next read state 1
  nstate <= rst1;
else
  data_out_int <= rbl_b;
nsecond_comparison <= '0';

--check for next state
if we = '0' and re = '0' then
  nstate <= idle;
elsif we = '1' and re = '0' then
  nstate <= wst;
elsif we = '0' and re = '1' then
  nstate <= rst1;
else
  nstate <= idle;
end if;
end if;
end case;
end process combinatorial;

state_reg: process(reset,clk)
begin
  if reset = '0' then
    pstate <= idle;
  elsif clk'event and clk = '1' then
    pstate <= nstate;
  end if;
end process state_reg;
end mealy;
B. Synopsys Script for Synthesis

TCL script for Synopsys DesignVision to generate gate level netlist from VHDL code for synthesis of memory controller.

```
set TCLK 3.0

remove_design -designs

# Analyze packages
analyze -library WORK -format vhdl {digital_code/dig_pkgs.vhd}

# Analyze sourcecode
analyze -library WORK -format vhdl {digital_code/dig_ctrl_n.vhd}

echo ---- dig_ctrl_n is being synthesized ----

# Elaborate design
elaborate dig_ctrl_n -architecture mealy -library work -update > reports/elaborated_n.rpt

# save elaborated design
write -hierarchy -format ddc -output DB/dig_ctrl_n_elab.ddc

#read saved design
read_file -format ddc DB/dig_ctrl_n_elab.ddc

# Set design constraints
# Define clock period and duty cycle
create_clock -name "clk" -period $TCLK {clk}

set_ideal_network clk
set_ideal_network reset

#set max area to 0
set_max_area 0

# Set input and output delays
# ------inputs ------
set_input_delay 0.25 -clock clk {address_msb re we data_in}
# signal from SA (also consider inserting a flip-flop after SA)
set_input_delay 0.25 -clock clk {rbl_a rbl_b}

# Set output_delays ------
set_output_delay 0.75 -clock clk {data_out}
I want all switches, pull up and pull down to be enabled as fast as possible, for
the BLS to have more time to settle

set_output_delay 0.15 -clock clk {get_ports {c*}}
set_output_delay 0.15 -clock clk [remove_from_collection [get_ports {c*}] clk]
set_output_delay 0.15 -clock clk [get_ports {p*}]
# Also, all the enable signals for decoders, sense amplifiers and replica array
should be set as fast as possible
set_output_delay 0.15 -clock clk [wt_replica_e]
set_output_delay 0.15 -clock clk [rd_replica_e]
set_output_delay 0.15 -clock clk [get_ports [add_decoder*]]
set_output_delay 0.15 -clock clk [get_ports {ref_decoder*}]
set_output_delay 0.15 -clock clk {san_e}

# Set input driver strengths
set_driving_cell -library f3d0a_a_generic_core_ttlp2v25c -lib_cell BUFX1
```
Multilevel Gain Cell Arrays for Fault Tolerant VLSI Systems

{address_msb re we data_in}
# reconiser this if flip-flop after SA has been introduced
set_driving_cell -library fsd0a_a_generic_core_ttlp2v25c -lib_cell BUFX1 {rbl_a rbl_b}
# set output capacitive load
set_load [load_of fsd0a_a_generic_core_ttlp2v25c/BUFX1] {data_out}
# reconiser loads of all outputs after parasitic extraction of layout
set_load 0.0005 [get_ports {c*}]
# extracted x2
set_load 0.0005 {wt_replica_e} set_load 0.0005 {rd_replica_e}
# extracted x2
set_load 0.002 [get_ports {add_decoder*}] set_load 0.002 [get_ports {ref_decoder*}]
# extracted x2
set_load 0.002 {san_e}
# rough estimate
set_load 0.002 [get_ports {p*}]
# rough estimate
# UNITS
#  of time [ns] and cap [pf]
# rise and fall time for pull up and pull down
set_min_delay 0.0 -fall_to [get_ports {pu*}]
set_min_delay 0.0 -fall_to [get_ports {pd*}]
set_min_delay 0.0 -rise_to [get_ports {pu*}]
set_min_delay 0.0 -rise_to [get_ports {pd*}]
set_max_delay 0.1 -fall_to [get_ports {pu*}]
set_max_delay 0.1 -fall_to [get_ports {pd*}]
set_max_delay 0.1 -rise_to [get_ports {pu*}]
set_max_delay 0.1 -rise_to [get_ports {pd*}]

# rise and fall time for bit line switches
set_min_delay 0.0 -fall_to [get_ports {cp*}]
set_min_delay 0.1 -fall_to [get_ports {cp*}]
set_max_delay 0.1 -fall_to [get_ports {cp*}]
set_max_delay 0.1 -fall_to [get_ports {cp*}]
set_min_delay 0.0 -rise_to [get_ports {cn*}]
set_min_delay 0.1 -rise_to [get_ports {cn*}]
set_max_delay 0.1 -rise_to [get_ports {cn*}]
set_max_delay 0.1 -rise_to [get_ports {cn*}]

# rise and fall time for decoders
set_min_delay 0.0 -fall_to [get_ports {add_decoder*}]
set_min_delay 0.1 -fall_to [get_ports {add_decoder*}]
set_max_delay 0.1 -fall_to [get_ports {add_decoder*}]
set_min_delay 0.0 -rise_to [get_ports {ref_decoder*}]
set_max_delay 0.1 -rise_to [get_ports {ref_decoder*}]
set_min_delay 0.0 -rise_to [get_ports {ref_decoder*}]
set_max_delay 0.1 -rise_to [get_ports {ref_decoder*}]

# rise and fall time for replica and sense amplifier
set_min_delay 0.0 -fall_to [get_ports {wt_replica_e}]
set_max_delay 0.1 -fall_to [get_ports {wt_replica_e}]
set_min_delay 0.0 -rise_to [get_ports {wt_replica_e}]
set_max_delay 0.1 -rise_to [get_ports {wt_replica_e}]
set_min_delay 0.0 -fall_to [get_ports {rd_replica_e}]
set_max_delay 0.1 -fall_to [get_ports {rd_replica_e}]
set_min_delay 0.0 -rise_to [get_ports {rd_replica_e}]
set_max_delay 0.1 -rise_to [get_ports {rd_replica_e}]
set_min_delay 0.0 -fall_to [get_ports {san_e}]
set_max_delay 0.1 -fall_to [get_ports {san_e}]
set_min_delay 0.0 -rise_to [get_ports {san_e}]
set_max_delay 0.1 -rise_to [get_ports {san_e}]

Multilevel Gain Cell Arrays for Fault Tolerant VLSI Systems

set_max_delay 0.1 -rise_to [get_ports {san_e}]

compile_ultra

# Save mapped design
write -hierarchy -format ddc -output DDC/mapped_n.ddc

# Report generation
# Reference report
report_reference -nosplit > reports/ref_n.rpt
# Area report
report_area > reports/area_n.rpt
# Timing report
report_timing > reports/timing_n.rpt

# save design constraints
write_sdc -nosplit SDC/digital_ctrl_n.sdc

#write out verilog netlist
change_names -h -rules verilog
write -h -f verilog -o netlists/dig_ctrl_n.v

C. MatLab files for Memory Model and Stimuli Generation

Memory Model

MatLab code for memory model.

function do=mldram(rst, re, we, addr, di)
%
--- parameters of the considered MLDRAM ---------------------------------
%- latency (nb. of cycles from applying input data until it is stored in the cell and could be read back)
rlat=4; % read latency
addrw=8; % address width
dataw=16; % data width (width of do bus)
wordw=32; % word width (16 2-bit gain cells)
% Note that the address pool of the memory is from 0 to 2^addrw-1, while it % is from 1 to 2^addrw for MatLab
%
% assume that all stimuli come cycle-true over the same number of cycles
ncycl=size(rst,1); % number of simulation cycles
mem=2*ones(2^addrw,wordw); % memory array; '2' means unknown, which would % translate to a don't care condition if it % were read out.
do = 2*ones(ncycl,dataw); % '2' means don't care
skip=0; % for cycles which need to be skipped (and read or write operation % has been initiated, and we must wait for it to complete)

for i=1:ncycl % i is the current cycle
  if skip>0
    % wait (do nothing)
    skip=skip-1;
  elseif rst(i)==0 % active low reset
    % do nothing
    % the reset operation of the address and data registers is not % modeled here
  elseif we(i)==1 && re(i)==0 % initiate a write operation % during cycle i+wlat-1, mem(addr(i))=di(i), and in the meanwhile % nothing else can happen (no other operations can be initiated).
    % That's why we can assign the new memory content already now, and % then we just wait for the required number of cycles.
  elseif wlat>0 % latency
    % wait (do nothing)
    wlat=wlat-1;
  elseif we(i)==1 && re(i)==0 % initiate a write operation
    % during cycle i+wlat-1, mem(addr(i))=di(i), and in the meanwhile % nothing else can happen (no other operations can be initiated).
    % That's why we can assign the new memory content already now, and % then we just wait for the required number of cycles.
  else
    % do nothing
  end
end
Multilevel Gain Cell Arrays for Fault Tolerant VLSI Systems

Multilevel Gain Cell Arrays for Fault Tolerant VLSI Systems

MatLab code for generation of stimuli used to simulate memory.

% Purpose: generate stimuli and expected responses for multilevel% gain-cell-based DRAM
 %Full Spectre simulations will be performed, and the stimuli/exp resps% must be in such format as to be readable by the 'vpwlf' source.% One source for each input/output.% The same stimuli should also be provided as bit patterns to be used on% the tester.%
clear;
% Verify the following scenarios:
% 1) ALL1
%all1=true; % Must be always included for tester!
% Write '1' to all addresses in increasing order (0 to 255)% (Important note: write '1' to all addresses with VDD2=1.2 should be% performed on the tester before VDD2 is increased to 1.9V for real% operation.)% Read '1' from all addresses
% 2) RANDOM
random=true;
% Write random data to deterministic addresses% Read back data from all addresses
% 3) MULTILEVEL CHECKER BOARD (MLCB)
%mlcb=true;
% Write data 00110011.... (11001100....) to all even (odd) addresses in% increasing order% Read back data from all addresses in increasing order
% 4) STEP
%step=true;
% 0 1 2 3 2 1 0 .
% 1 2 3 2 1 0 1 .
% 2 3 2 1 0 1 2 .
% 3 2 1 0 1 2 3 .
% 2 1 0 1 2 3 2 .
% 1 0 1 2 3 2 1 .
% 0 1 2 3 2 1 0 .
% . . . . . .
% 5) LEAKDOWN
%leakdown=true;
Multilevel Gain Cell Arrays for Fault Tolerant VLSI Systems

% Write highest level (3) to all cells, % keep WBLs at lowest level, % read back all cells after X cycles, % where X is increased until wrong data is read out.
% 6) LEAKUP
% leak=true;
% Write lowest level (0) to all cells, % keep WBLs at highest level, % read back all cells after X ns, % where X is increased until wrong data is read out.

% ---- Configuration part =================================================
% Memory -------------------------------------------------------------
wlat=1; % write latency (nb. of cycles from applying input data until it is % stored in the cell and could be read back)
rlat=4; % read latency is 4 cycles
addrw=8; % address width
dataw=16; % data width (width of do bus)
wordw=32; % word width (16 words storing 2-bits per gain cell)
% Is is assumed here that one can perform either a write or read operation, % but not both of them at the same time.
% Waveform control ---------------------------------------------------
T=3e-9; % target clock period is 3 ns
f=1/T; % target frequency is 333.33 MHz
appli=0; % stimuli application time
acqui=2.8e-9; % response acquisition time
tran=100e-12; % transition time of input signals (100ps)
v1=0.0; % low voltage
vh=1.2; % high voltage
vu=0.6; % undefined, or, in the context of exp resp, don't care
% The retention time under worst case leakage conditions at 85 deg % centigree is expected to be around 140 us.
%X=ceil(140e-6/T);

% ---- Interface =========================================================
% Signals (and memory pre-allocation) -----------------------------------
nvect=100000; % estimated number of test vectors (for memory pre-allocation)
% CLKxCI can be produced with a pulse source
% reset (1 bit)
% re (1 bit)
% we (1 bit)
% address (8 bits)
% data_in (32 bits)
% data_out (16 bits)
% WBLCtrlxSI (2 bits)
% wblctrl=[0 0] => WBLs are floating (from both sides) while memory is % in the idle state (default)
% wblctrl=[0 1] => WBLs are driven low when memory is in idle state
% wblctrl=[1 0] => WBLs are driven high when memory is in idle state
%wblctrl=[0 0]; % default value
% Records
% reset re re address data_in
% stimuli(1 2 3 4-11 12-43)
global stimuli;
stimuli = zeros(nvect,1+1+addrw+wordw);

% Initialization
global k;
k=1;
% Reset
append([0 0 0 zeros(1,addrw) zeros(1,wordw)]);
% stimuli while waiting until new data can be accepted
global waitvec;
Multilevel Gain Cell Arrays for Fault Tolerant VLSI Systems

waitvec=[1 0 0 zeros(1,addrw) zeros(1,wordw)];
% multilevel checker board (mlcb) vector and its complement
% mlcbvec=zeros(1,wordw);
% mlcbvecb=zeros(1,wordw);
% for i=3:4:wordw
%    mlcbvec(1,i:i+1)=[1 1];
% end
% for i=1:4:wordw
%    mlcbvecb(1,i:i+1)=[1 1];
% end
% all1=false;
% random=false;
% mlcb=false;
% %step=false;
% leakdown=false;
% leakup=false;

if all1==true
% write '1' to all addresses in increasing order
for addr = 0:2^addrw-1
    append([1 0 1 int2bin(addr,addrw) ones(1,wordw) wblctrl]);
    % wait for 'wlat-1' until new data can be accepted
    wait(wlat-1);
end
% read back from all addresses
for addr = 0:2^addrw-1
    append([1 1 0 int2bin(addr,addrw) zeros(1,wordw) wblctrl]);
    % wait for 'rlat-1' until new data can be accepted
    wait(rlat-1);
end
end

if random==true
% Write random data to deterministic addresses in increasing order
for addr = 0:2^addrw-1; %address generation%
    append([1 0 1 int2bin(addr,addrw) randi([0,1],[1,wordw])]);
    % wait for 'wlat-1' until new data can be accepted
    wait(wlat-1);
end
% Read back data from all addresses in decreasing order
for addr = 2^addrw-1:1:0;
    for x=1:rlat;% to keep the read enable and address signals stable for 4
        cycles
        append([1 1 0 int2bin(addr,addrw) zeros(1,wordw)]);
        % wait for 'rlat-1' until new data can be accepted
        wait(rlat-4);
    end
end
end

if mlcb==true
% Write data 00110011.... (11001100....) to all even (odd) addresses
for addr = 0:2^addrw-1
    if mod(addr,2)==0
      % even address
      append([1 0 1 int2bin(addr,addrw) mlcbvec wblctrl]);
    else
      % odd address
      append([1 0 1 int2bin(addr,addrw) mlcbvecb wblctrl]);
    end
    % wait for 'wlat-1' until new data can be accepted
    wait(wlat-1);
end
% Read back data from all addresses
for addr = 0:2^addrw-1
end

Muhammad Umer Khalid Telecommunication Circuits Laboratory, EPFL
Multilevel Gain Cell Arrays for Fault Tolerant VLSI Systems

```matlab
append([1 1 0 int2bin(addr,addrw) zeros(1,wordw) wblctrl]);
% wait for 'rlat-1' until new data can be accepted
wait(rlat-1);
% end
% end
% if step==true
% % Write the following pattern
% 0 1 2 3 2 1 0 ... @ addr 0
% 1 2 3 2 1 0 1 ... @ addr 1
% 2 3 2 1 0 1 2 ... @ addr 2
% 3 2 1 0 1 2 3 ... @ addr 3
% 2 1 0 1 2 3 2 ... @ addr 4
% 1 0 1 2 3 2 1 ... @ addr 5
% 0 1 2 3 2 1 0 ... @ addr 6
% . . . . . . .
% basicstevec=[0 0 1 1 0 1 1 0 0 1];
% !!! CAUTION: not generic !!!
% stepvec32=[basicstevec basicstevec basicstevec(1,1:8)];
% %addrseq=[127];
% %addrseq=[0 1 21 127 131 254 255];
% %addrseq=[2 70 81 88 90 106 121 141 150 166 173 186 239]; % Passed!
% %addrseq=[5 75 80 89 91 105 122 140 153 161 169 174 189 242]; % Passed!
% %addrseq=[81 173];
% %addrseq=[0 255];
% %addrseq=[0 127 128 255];
% for i = 1:2 %0:2^addrw-1 %!!!!!!INPUT HERE THE ADDRESS TO WRITE
% addr=addrseq(i);
% vec=rotate(stepvec32,2*mod(addr,6));
% append([1 0 1 int2bin(addr,addrw) vec wblctrl]);
% waitvec=[1 0 0 int2bin(addr,addrw) vec wblctrl];
% % wait for 'wlat-1' until new data can be accepted
% wait(wlat-1);
% end
% % Read back from all addresses
% for i = 1:2 %0:2^addrw-1 %!!!!!!INPUT HERE THE ADDRESS TO READ
% addr=addrseq(i);
% append([1 0 1 int2bin(addr,addrw) zeros(1,wordw) wblctrl]);
% waitvec=[1 0 0 int2bin(addr,addrw) zeros(1,wordw) wblctrl];
% % wait for 'rlat-1' until new data can be accepted
% wait(rlat-1);
% end
% % end
% if leakdown==true
% wblctrl=[0 1]; % Drive WBLs low when memory is in idle state
% waitvec=[1 0 0 zeros(1,addrw) zeros(1,wordw) wblctrl];
% % write '1' to all addresses in increasing order
% for addr = 0:2^addrw-1
% append([1 0 1 int2bin(addr,addrw) ones(1,wordw) wblctrl]);
% % wait for 'wlat-1' until new data can be accepted
% wait(wlat-1);
% end
% % wait for X cycles
% wait(X);/home/khalid/Desktop
% % read back from all addresses
% for addr = 0:2^addrw-1
% append([1 1 0 int2bin(addr,addrw) zeros(1,wordw) wblctrl]);
% % wait for 'rlat-1' until new data can be accepted
% wait(rlat-1);
% end
% wblctrl=[0 0]; % set it back to default value
% waitvec=[1 0 0 zeros(1,addrw) zeros(1,wordw) wblctrl];
% % end
% if leakup==true
```
Multilevel Gain Cell Arrays for Fault Tolerant VLSI Systems

% wblctrl=[1 0]; % Drive WBLs high when memory is in idle state
% waitvec=[1 0 0 zeros(1,addrw) zeros(1,wordw) wblctrl];
% % write '1' to all addresses in increasing order
% for addr = 0:2^addrw-1
%     append([1 1 int2bin(addr,addrw) zeros(1,wordw) wblctrl]);
%     % wait for 'wlat-1' until new data can be accepted
%     wait(wlat-1);
% end
% % wait for X cycles
% wait(X);
% % read back from all addresses
% for addr = 0:2^addrw-1
%     append([1 1 int2bin(addr,addrw) zeros(1,wordw) wblctrl]);
%     % wait for 'rlat-1' until new data can be accepted
%     wait(rlat-1);
% end
% wblctrl=[0 0]; % set it back to default value
% waitvec=[1 0 0 zeros(1,addrw) zeros(1,wordw) wblctrl];
% end
% Add 1 more cycle due to latency of output register
% append(waitvec);

% get expected responses from mldram model
rst=stimuli(:,1);
re=stimuli(:,2);
we=stimuli(:,3);
addr=stimuli(:,4:addrw+3);
di=stimuli(:,addrw+4:wordw+addrw+3);
wbl=stimuli(:,wordw+addrw+4:size(stimuli,2));
expresp = mldram(rst, re, we, addr, di); % exp resp doesn't depend on
% WBLCtrlxSI
% Transient simulation time
transsim=T*(k-1)+1e-9;
disp(['Vector generation has completed. Transient simulation time is ' ...
     num2str(transsim)]);

% First prepare bit pattern files, then translate them to be used with
% the 'vpwlfl' source.
stm = fopen('stimuli.asc','w');
exp = fopen('expresp.asc','w');
for i=1:k-1
    % print stimuli
    fprintf(stm, '\n %u %u %u ', rst(i), re(i), we(i));
    for j=1:addrw
        fprintf(stm,'%u',addr(i,j));
    end
    fprintf(stm, '\n');
    for j=1:wordw
        fprintf(stm,'%u',di(i,j));
    end
    fprintf(stm,' %u%u
',wbl(i,1), wbl(i,2));
    % print exp resp
    for j=1:dataw
        if expresp(i,j)==2
            fprintf(exp,'-');
        else
            fprintf(exp,'%u',expresp(i,j));
        end
    end
    fprintf(exp,'\n');
end
fclose(stm);
fclose(exp);
Multilevel Gain Cell Arrays for Fault Tolerant VLSI Systems

% Note how indices in VHDL (used always downto) and Matlab differ.
% Example 'addr':
% VHDL: 7 6 5 4 3 2 1 0
% Matlab: 1 2 3 4 5 6 7 8

% 'vpwlf' sources
frst=fopen('stm/rst','w');
fre=fopen('stm/re','w');
ewe=fopen('stm/we','w');
% fwb1=fopen('stm/wblctrl_1','w');
% fwb0=fopen('stm/wblctrl_0','w');
% with cycle 'c', we associate rst(c) which is applied 'appli' seconds
% after the active clock edge.
printpwl(rst,frst,T,appli,tran,vl,vh,vu); % advanced the signal for setup
printpwl(re,fre,T,appli,tran,vl,vh,vu); % advanced the signal for setup
% printpwl(wbl(:,1),fwb1,T,appli,tran,vl,vh,vu);
% printpwl(wbl(:,2),fwb0,T,appli,tran,vl,vh,vu);
fclose(frst);
fclose(fre);
fclose(ewe);
% fclose(fwb1);
% fclose(fwb0);
for i=1:addrw
% addr
fid=fopen(['stm/addr_' int2str(addrw-i)],'w');
printpwl(addr(:,i),fid,T,appli,tran,vl,vh,vu);
fclose(fid);
end
for i=1:wordw
% di
fid1=fopen(['stm/di_' int2str(wordw-i)],'w');
printpwl(di(:,i),fid1,T,appli,tran,vl,vh,vu);
fclose(fid1);
end
for i=1:dataw
% do
fid2=fopen(['exp/do_' int2str(dataw-i)],'w');
% no delay for exp resp
printpwl(expresp(:,i),fid2,T,0.0,tran,vl,vh,vu);
fclose(fid2);
end
fid=fopen('ocean.inc','w');
fprintf(fid,['transsimend=' num2str(transsim) '\n']);
fprintf(fid,['period=' num2str(T) '\n']);
fprintf(fid,['ncycl=' num2str(k-1) '\n']);
fprintf(fid,['acqui=' num2str(acqui) '\n']);
fprintf(fid,['vl=' num2str(vl) '\n']);
fprintf(fid,['vh=' num2str(vh) '\n']);
fprintf(fid,['vu=' num2str(vu) '\n']);
fclose(fid);
disp('All files have been written out.');
Multilevel Gain Cell Arrays for Fault Tolerant VLSI Systems

Append

MatLab code to append data.

function append(a)
global stimuli;
global k;
stimuli(k,:)=a;
k=k+1;

Binary to Integer

MatLab code to convert binary to integer number

function z=bin2int(x)
z=0;
for i=1:size(x,2)
j=size(x,2)-i;
z=z+x(i)*2^j;
end

Integer to Binary

MatLab code to convert integer to binary number.

function z=int2bin(x,n)
a=dec2bin(x,n);
z=zeros(1,size(a,2));
for i = 1:size(a,2)
z(1,i)=str2double(a(1,i));
end

Parallel to Serial

MatLab code to convert parallel data to serial data.

function [vec1, vec2] = par2ser(vec)
% Parallel to serial conversion
% vec:  z1 z0 y1 y0 ... a1 a0 (original vector)
% vec1: z0 y0 ... a0 (to be applied in 1st cycle (LSB))
% vec2: z1 y1 ... a1 (to be applied in 2nd cycle (MSB))
n=size(vec,2); % assume that the length of vec is an even number
vec1=zeros(1,n/2);
vec2=zeros(1,n/2);
for i=1:n/2
vec1(1,i)=vec(1,2*i);
vec2(1,i)=vec(1,2*i-1);
end

Serial to Parallel

MatLab code to convert serial data to parallel data.

function vec=ser2par(vec1, vec2)
% Serial to parallel conversion
% vec1: z0 y0 ... a0 (applied in 1st cycle (LSB))
% vec2: z1 y1 ... a1 (applied in 2nd cycle (MSB))
n=size(vec1,2); % both vec1 and vec2 must have the same size
vec=zeros(1,2*n);
for i=1:n
vec(1,2*i-1)=vec2(i);
vec(1,2*i)=vec1(i);
end
Rotate

MatLab code to rotate data.

function z=rotate(x,n)
% rotate the vector x to the left by n positions
% Example:
% x1 x2 x3 x4 x5 x6 ... xo xp, rotate by 2
% x3 x4 x5 x6 ... xo xp x1 x2
% z1 z2 z3 z4 ...
% s=size(x,2);
% z(1,1:s-n)=x(1,n+1:s);
% z(1,s-n+1:s)=x(1,1:n);

Wait

MatLab code to produce delay.

function wait(n)

global waitvec;
for j = 1:n
    append(waitvec);
end

Print PWL

MatLab code to print data for Cadence file.

function printpwl(data, fid, T, delay, tran, vl, vh, vu)

global k;
% initialization (assume V=0 at t=0)
fprintf(fid,'%10.9e %3.2e
', 0.0, 0.0);
if delay>0.0
    fprintf(fid,'%10.9e %3.2e
', delay, 0.0);
end
for i=1:k-1
    if data(i)==0
        % value which needs to be set up (regardless of what it was before)
        fprintf(fid,'%10.9e %3.2e
', delay+T*(i-1)+tran, vl);
        % keep this value for 1 cycle
        fprintf(fid,'%10.9e %3.2e
', delay+T*i, vl);
    elseif data(i)==1
        fprintf(fid,'%10.9e %3.2e
', delay+T*(i-1)+tran, vh);
        fprintf(fid,'%10.9e %3.2e
', delay+T*i, vh);
    else
        % data is undefined
        fprintf(fid,'%10.9e %3.2e
', delay+T*(i-1)+tran, vu);
        fprintf(fid,'%10.9e %3.2e
', delay+T*i, vu);
    end
end
D. OceanScript for Comparison

OceanScript to compare the results of Spectre simulation with expected responses generated by MatLab.

```
load("/home/khalid/memrep/stimuli/ocean.inc")
; Includes the following variables used for the stimuli generation
; transsimend
; period
; ncycl
; acqui
; vl
; vh
; vu

ocnWaveformTool( 'wavescan' )
simulator( 'spectre' )
design( "/scratch/khalid/cds/simulation/memory_tb/spectre/config/netlist/netlist")
resultsDir( "/scratch/khalid/cds/simulation/memory_tb/spectre/config" )
modelFile( 
'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/../../Models/Spectre/L90_25IO_NVT_V021.lib.scs" "tt")

'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/../../Models/Spectre/L90_25IO_V111.lib.scs" "tt")

'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/../../Models/Spectre/L90_NCAP25_V113.lib.scs" "typ")

'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/../../Models/Spectre/L90_varmis_25_rf_V011.lib.scs" "typ")

'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/../../Models/Spectre/L90_33IO_GOX52_VT21.lib.scs" "tt")

'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/../../Models/Spectre/L90_25IO_RF_V021.lib.scs" "tt")

'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/../../Models/Spectre/L90-104251-resistor-control-V041.scs" "")

'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/../../Models/Spectre/L90_BJT_V111.lib.scs" "tt_bip")

'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/../../Models/Spectre/L90_DIODE_V101.mdl.scs" "")

'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/../../Models/Spectre/L90_LL12_RF_V021.lib.scs" "tt")

'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/../../Models/Spectre/L90_LLVT12_RF_VTAB.lib.scs" "tt")

'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/../../Models/Spectre/L90_LL12_V102.lib.scs" "tt")

'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/../../Models/Spectre/L90_LLHVT12_V101.lib.scs" "tt")

'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/../../Models/Spectre/L90_LLLVT12_V101.lib.scs" "tt")

'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/../../Models/Spectre/L90_LLLVT12_V011.lib.scs" "tt")
```

Multilevel Gain Cell Arrays for Fault Tolerant VLSI Systems

'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/..Models/Spectre/L90SP_NCAP10_V112.lib.scs" "typ")
'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/..Models/Spectre/L90_NCAP12_L_L_V102.lib.scs" "typ")
'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/..Models/Spectre/L90_SP10_V061.lib.scs" "tt")
'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/..Models/Spectre/L90_SPHVT10_V111.lib.scs" "tt")
'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/..Models/Spectre/L90_SPNVT10_V011.lib.scs" "tt")
'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/..Models/Spectre/L90_mimcaps_20f_kf_V011.lib.scs" "typ")
'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/..Models/Spectre/L90_momcaps_V041.lib.scs" "typ")
'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/..Models/Spectre/L90_vardiop_rf_V011.lib.scs" "typ")
'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/..Models/Spectre/L90_varmis_2.lr rf_V021.lib.scs" "typ")
'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/..Models/Spectre/momcaps_array_vp3_rfvcl_V011.lib.scs" "typ")
'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/..Models/Spectre/momcaps_array_vp4_rfvcl_V011.lib.scs" "typ")
'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/..Models/Spectre/rnhr_rf_V011.lib.scs" "typ")
'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/..Models/Spectre/rnppo_rf_V011.lib.scs" "typ")
'("/dkits/umc/lms90_b15pb/Designkits/Cadence/umc90nm/..Models/Spectre/L90_SP10_RF_V021.lib.scs" "tt")
)
analysis('tran ?stop transsimend )
option( ?categ 'turboOpts 'uniMode "Turbo"
)
 temp( 27 )
run()

; report file
load("/home/khalid/memrep/stimuli/ocean.inc")
p=outfile("/home/khalid/Desktop/sim.rpt" "w")
printf(p "Time\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\tt\t\t

Muhammad Umer Khalid  Telecommunication Circuits Laboratory, EPFL

76
actwf=makeTable("actual waveforms")
expwf=makeTable("expected waveforms")
for( i 1 16
    j=16-i
    actwf[j]=VT(strcat("/do" concat(j) ">"))
    expwf[j]=VT(strcat("/do_" concat(j) "_exp"))
)
for( k 0 nycl-1
    s=acqui+k*period ; sampling instant
    fprintf(po "%10.9e \t%10d \t" s k)
    ; compute the y values only once and store them
    ; (this code is not much faster than 'compare1.ocn')
    ; at the same time, can print the actual response
    act=makeTable("actual values")
    exp=makeTable("expected values")
    for( i 1 16
        j=16-i
        act[j]=value(actwf[j] s)
        exp[j]=value(expwf[j] s)
        ; print actual responses
        if( act[j]>vl+0.95*(vh-vl) then
            fprintf(po "1")
        else
            if( act[j]<vl+0.05*(vh-vl) then
                fprintf(po "0")
            else
                fprintf(po "U")
        )
    )
    fprintf(po "\n")
    ; make the comparison
    correct=t ; assume act resp is correct
    for( i 1 16
        j=16-i
        if( exp[j]==vh then
            if( act[j]<vl+0.95*(vh-vl)
                correct=nil
            )
        else
            if( exp[j]==vl
                if( act[j]>vl+0.05*(vh-vl)
                    correct=nil
                )
            )
    )
    if( !correct then
        fprintf(po "Error: expected was \t\t\t\t")
        for( i 1 16
            j=16-i
            if( exp[j]==vh then
                fprintf(po "1")
            else
                if( exp[j]==vl
                    if( act[j]>vl+0.05*(vh-vl)
                        correct=nil
                    )
                )
        )
    )
Multilevel Gain Cell Arrays for Fault Tolerant VLSI Systems

wt_energy = (integ(IT("/V47/PLUS") 3e-09 771e-09) * 1.2)
fprintf(po "Write energy = %1.15f \n" wt_energy)
rd_energy = (integ(IT("/V47/PLUS") 771e-09 3.843e-06) * 1.2)
fprintf(po "Read energy = %1.15f \n" rd_energy)
close(po)