VLSI IMPLEMENTATION OF CROSS-PARITY AND MODIFIED DICE FAULT TOLERANT SCHEMES

by

DANIEL RYAN BLUM

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

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

MAY 2004

To the faculty of Washington State University:

The members of the Committee appointed to examine the thesis of DANIEL RYAN BLUM find it satisfactory and recommend that it be accepted.

_____
Chair

_____

_____

ACKNOWLEDGEMENT

I would like to thank my advisor Dr. José Delgado-Frias for his leadership, guidance and patience. Working with Dr. Delgado has helped me to develop substantially as an engineer and as a person. I also appreciate the assistance of Dr. Valeriu Beiu and Dr. Jabulani Nyathi for their contributions to this thesis and my work at WSU in general.

I would also like to thank my mother, father and sister for providing constant support and encouragement through the years. My family has given me the confidence to pursue my goals.

Finally, I would like to extend my appreciation to the Washington State University School of Electrical Engineering and Computer Science for awarding me a teaching assistantship position. Without this support, I wouldn't have had the opportunity to write this thesis.

VLSI IMPLEMENTATION OF CROSS-PARITY AND MODIFIED DICE FAULT TOLERANT SCHEMES

Abstract

by Daniel Ryan Blum, M.S.
Washington State University
May 2004

Chair: José G. Delgado-Frias

Fault-tolerant approaches to digital system design are becoming increasingly important, in particular for mission critical systems. In this document, two implementations of a fault-tolerant cell for a reconfigurable DSP processor are described. This cell is centered around a 32-bit memory which is used as a lookup table inside the processor. Fault-tolerance is implemented through the use of a cross-parity scheme and a modifided DICE (Dual-Interlocked storage Cell) design. The cross-parity scheme is a system-level approach that computes and stores parity bits during write operations, and uses these bits during memory reads to identify errors in the system. One error can be corrected during every read operation of the cell. A prototype for this system has been fabricated in 0.5μm CMOS VLSI technology. The modified DICE design is a circuit-level approach that utilizes redundancy and feedback to quickly correct transient errors inside of individual memory latches. The benefits and drawbacks of both approaches will be compared and analyzed.

TABLE OF CONTENTS

LIST OF FIGURES

Page

Chapter 1

Introduction

The steady development of digital electronics in the last few decades has led to a number of useful applications, including personal computers, embedded systems, and digital signal processing (DSP). The performance of DSP exceeds that of traditional analog signal processing in a number of ways, which has opened up paths to greater functionality in many designs. DSP is a vital component in communications, multimedia, and space applications. Modern mobile and wireless DSP devices require high performance and low power consumption, which have traditionally been combined only in custom integrated circuits [1]. However, the rising cost of custom solutions and the increasing capabilities of advanced fabrication processes have made reconfigurable designs more attractive. A reconfigurable architecture can implement a number of different designs with substantial performance, at the cost of modest power consumption [2]. The stability of such a system can be enhanced by incorporating fault-tolerance into its architecture. This thesis focuses on the VLSI design and implementation of fault-tolerant components for a reconfigurable DSP processor.

1.1 Reconfigurable DSP Architecture

A reconfigurable DSP architecture has been designed to balance performance, power consumption, and versatility [3,4]. This architecture is made up of medium-grain cells, as opposed to the fine-grain components that make up field programmable gate arrays (FPGAs). For this application, FPGAs require excessive area and power to achieve an unnecessary level of

flexibility. Due to the regularity of most DSP algorithms, a medium-grain reconfigurable structure is sufficient. The structure featured in this section consists of an array of cells that perform 4-bit operations. Every cell is connected to its eight neighbors by sixteen 4-bit busses. Figure 1.1 is a high-level illustration of the reconfigurable DSP architecture.

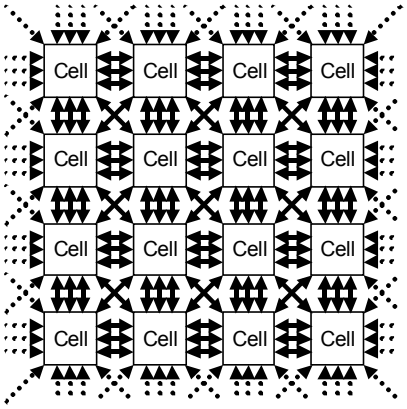

Figure 1.1: High-Level View of the Reconfigurable DSP Architecture

The processing core of each cell is made up of a 4x4 array of elements. Each element is a 16x2-bit lookup table that stores the truth table of a user-defined function. The array of elements can be arranged into a memory mode or mathematics mode configuration, which are displayed in figure 1.2. The memory mode arrangement turns the cell into a 64x8 bit random access memory, providing storage capability for the processor. In mathematics mode, the structure of the array of elements is similar to that of a carry-save multiplier. This facilitates the efficient implementation of many arithmetic functions used in DSP, including addition and multiply-accumulate.

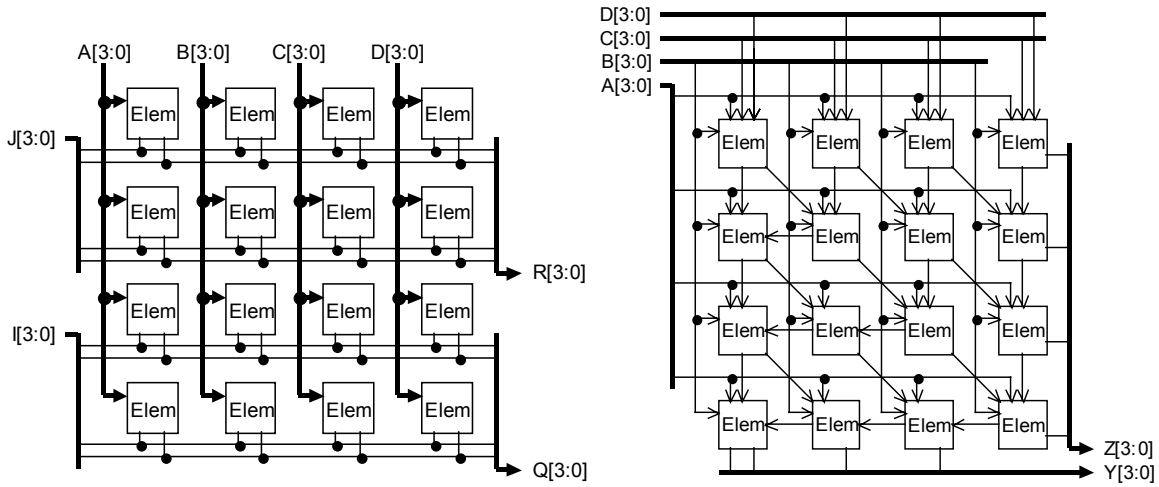

Figure 1.2: Array of Elements in the Memory Mode and Mathematics Mode Configurations

It is possible to implement functions of multiple word lengths with this reconfigurable architecture. Each cell manipulates 4-bit operands, and the cells can be cascaded to process longer data. Figure 1.3 shows four cells interconnected to implement an 8-bit multiply-accumulate function. Word lengths of 16, 32, 64 or even 128 bits can be achieved in this fashion. Many functions with long word lengths take multiple cycles to compute. To increase the throughput, pipeline latches are present in every cell, allowing a new operation to be initiated during every clock cycle [3,4].

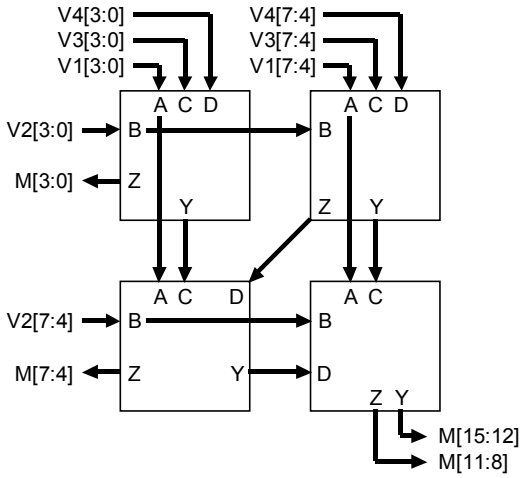

Figure 1.3: 8-Bit Multiply-Accumulate Function Implemented with Four Cells

1.2 Demand for Fault-Tolerance

Reliability and fault-tolerance are primary concerns in the design of digital systems, particularly when considering mission critical systems such as space communications. The failure of such a system to function correctly may result in undesirable consequences. Many digital systems, ranging from computers to cellular phones, and DVD players to satellites, rely on digital signal processing (DSP) to achieve their functionality. However, few of these DSP devices utilize any form of fault-tolerance to improve their reliability. None of them combine fault-tolerance with the flexibility garnered through the use of a reconfigurable architecture. Research in the area of fault-tolerance is becoming increasingly important because the reduction of integrated circuit (IC) feature size is resulting in circuits that are more fragile. Decreased gate capacitances allow charged particles to exert greater influence on transistor operation [5].

Many reconfigurable architectures utilize memory as a core component. In particular, this report focuses on a reconfigurable DSP processor that uses memory as lookup tables (LUTs) to implement user-programmable functions. Random access memories inside of digital systems are especially susceptible to error, as a voltage spike on a feedback line would be amplified by the latch inverters, which could easily change the state of the latch. A single erroneous value stored in this memory can continually disrupt the functionality of the entire processor.

1.3 Causes of Faults in Integrated Circuits

Latch-up, burn-out, oxide charging, reduction of carrier lifetimes, single-event upsets (SEUs), and electrical noise are some of the faults that affect ICs [6]. The first four faults cause permanent damage to a chip, and must be addressed through process enhancements or a software/hardware reconfiguration scheme that can bypass the damaged circuitry. SEUs are small particles that can momentarily change the voltage level of a node in an IC, which makes their effect similar to that of electrical noise. Environments that are not adequately shielded against radiation have high

rates of SEU occurrence, such as in space or at high altitude [7]. Different types of faults occur in different environments, so it is important to tailor individual fault-tolerant schemes to demands of the situation.

1.4 Overview of Fault-Tolerant Schemes

An efficient error correction system needs to be designed to protect a 32-bit memory inside of every element in the reconfigurable DSP chip. System-level schemes rely on components outside of the memory structure to perform calculations and correct errors that may be present in the memory. Circuit-level schemes exist entirely inside of the memory structure, meaning that fault-tolerance is incorporated directly into the design of the RAM latches [8]. Presented in this section are four of the major schemes to implement fault-tolerance in hardware. Cross-parity, Hamming code, and Triple Modular Redundancy (TMR) are system-level schemes, while the Dual-Interlocked storage Cell (DICE) approach is a circuit-level scheme.

Error detection and correction in the cross parity scheme is made possible by the storage of a parity bit for every row and column in a memory unit [9]. These parity bits are generated during writes to the memory. When a write occurs to the location corresponding to row *i* and column *j* of the memory, parity bits for *i* and *j* must be updated. If the same memory location is read at a later time, then the current parity of row *i* and column *j* is compared to the stored parity of *i* and *j*. Figure 1.4 illustrates the relationship between the parity bits and the rows and columns of the data memory. If the current parity is not consistent with the stored parity, then the memory location in question contains an incorrect bit (assuming there is no more than one error in the system). If this is not the case, then the bit is determined to be correct.

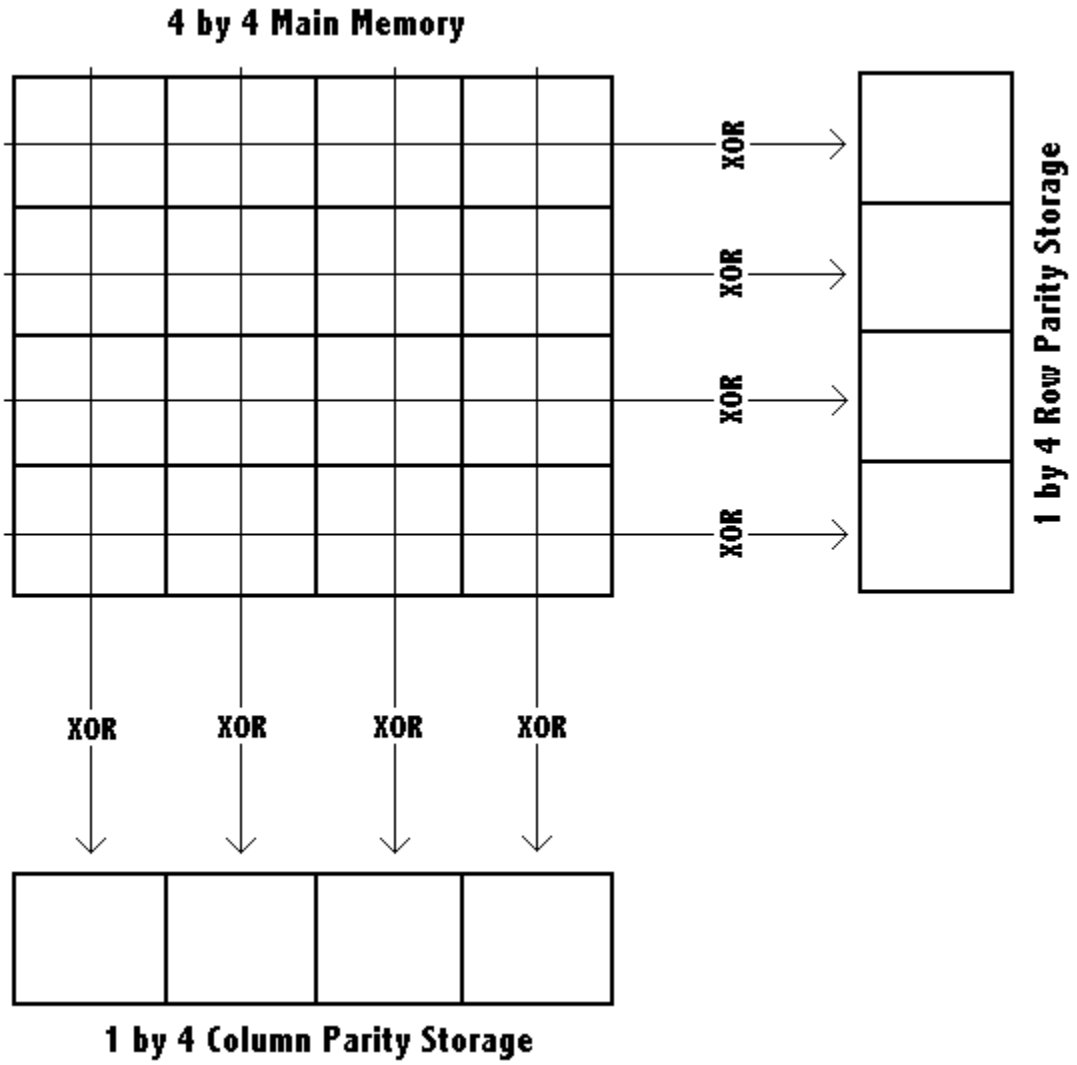

Figure 1.4: Overview of the Cross-Parity Scheme

The Hamming code approach divides the memory into multiple data words, and provides the ability to correct up to one error at a time in each data word. Encoded priority bits are inserted into these data words. An example of an 8-bit data word with four inserted Hamming parity bits is shown in figure 1.5. During a read operation, this encoded priority is decoded to give the position of an erroneous bit in the data word, if one exists. Accomplishing this requires an extensive XOR network to calculate the encoded priority, and a dedicated decoder to determine the position of the incorrect bit. In contrast, cross-parity requires a much smaller XOR

capability, and no dedicated decoder. Because of this, it is obvious that the cross-parity is more optimal in terms of both size and computational complexity. Therefore, Hamming code was ruled out for this project.

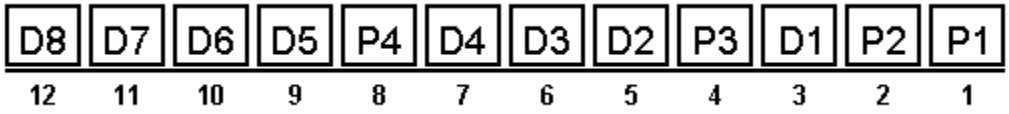

Figure 1.5: Hamming Parity Bits P1-P4 Inserted into an 8-Bit Data Word D1-D8

On the other hand, an evaluation of TMR does not reveal such obvious inadequacies. In a TMR-based system, three identical memory cells would be used, each storing the same data. When a read operation is performed, the three memory cells each send their version of the data. A voting circuit then passes on the data that was sent by the majority of the memory cells. A graphical depiction of this is shown in figure 1.6. Such a system continues to function correctly when multiple errors are present in its memory, which is not possible in a cross-parity system. However, this comes at the cost of 200% more memory cells, in addition to voting logic and other circuitry. Cross-parity only requires 50% more memory, in addition to XOR logic and control signal generation. Also, updating an incorrect bit in memory is more complicated in TMR, as the module containing the error must be identified and coordinated with the appropriate memory control signals. After weighing all of these factors, it was determined that TMR would be more effective in situations where burst errors are more probable, whereas cross-parity is more appropriate for errors that are separated by at least one read cycle. The latter case is considered in this report.

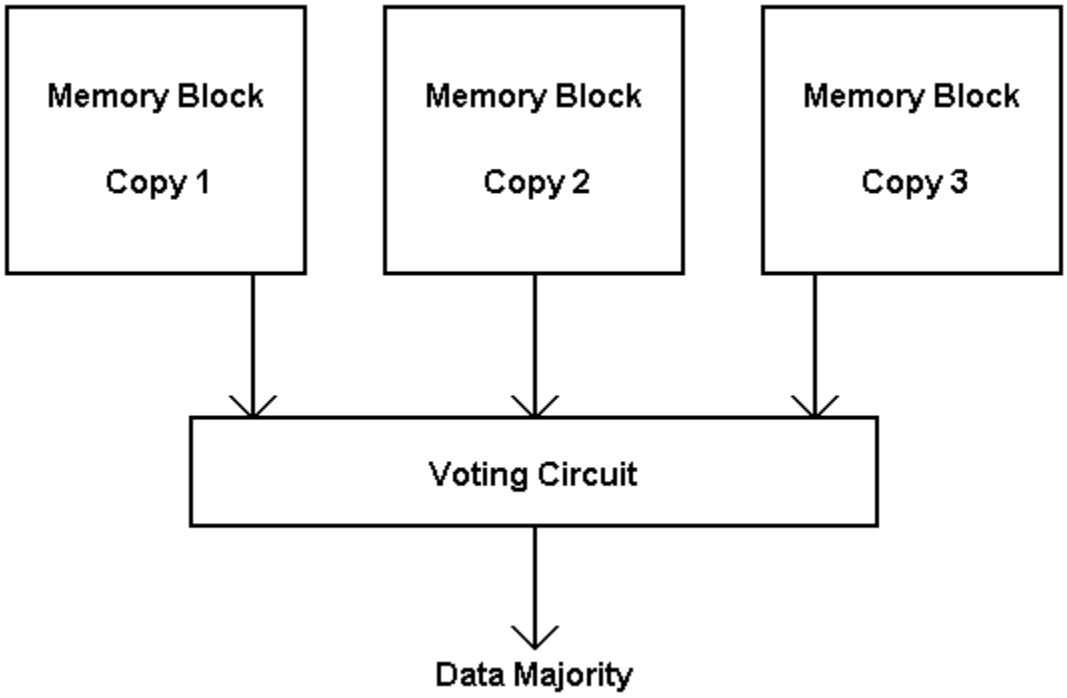

Figure 1.6: Block Diagram of a TMR System

Permanent damage to memory that is caused by faults must be detected and bypassed to insure the robust operation of a system. The software in a reconfigurable architecture can be designed to detect this damage and reconfigure the system to avoid compromised circuitry. In addition, a number of hardware-based methods can be used to accomplish this, including the system-level approaches described above. System-level approaches require significant area and delay overhead outside of the memory itself. This overhead may not be necessary in all circumstances.

In a number of situations, the only faults that will affect an IC are SEUs and other transient errors. Decreasing the feature size of a circuit increases the impact of an SEU by a power of two, which means that SEUs are becoming more of a problem as IC technology improves [5]. The system-level schemes listed above will protect against SEUs, but they are not optimal. First off, it would be preferable if the chosen solution reacted immediately to an SEU

instead of only when read operations are performed. Secondly, the capability to bypass affected hardware is unnecessary, as the system can simply pause until the glitch dissipates. Both of these objectives can be accomplished by adopting a circuit-level approach, where SEUs are prevented by modifying the latches themselves, instead of adding additional components outside of the memory cells. This approach can significantly reduce area and delay of the system, as the error-correcting scheme is integrated directly into the memory it is protecting [10].

The Dual Interlocked storage Cell (DICE) is a circuit-level design that has the capability to recover from transient faults at any of its feedback nodes [8]. The DICE memory cell is based off of four inverters which are connected in an unorthodox fashion. Arranging the latch interconnectivity this way assures that two separate inverters will restore the node to its original state via feedback. However, the unmodified DICE cell does not have the capability to recover from transient errors on its data or feedback lines. This report will detail a scheme that adds redundant data and control signals to the DICE cell, providing it with the ability to resist a single transient error at any location in the memory unit.

Two implementations of a fault-tolerant memory-based element have been designed for use in a medium-grain reconfigurable DSP array. Each element in the DSP array utilizes a 32-bit fault-tolerant memory unit as a lookup table to perform functional operations. The cross-parity scheme achieves fault-tolerance at the system level, while the modified DICE method provides fault-tolerance at the circuit level. A prototype of the cross-parity approach has been fabricated.

1.5 Thesis Outline

The remainder of this report is organized as follows: Section 2 contains an overview of the cross-parity scheme. Schematics and layouts for the cross-parity scheme are exhibited in Section 3. Simulations of the cross-parity scheme are analyzed in Section 4. Section 5 presents schematics and simulations of the modified DICE approach. Finally, conclusions are presented in Section 6.

Chapter 2

Cross-Parity Scheme

The cross-parity method of error correction was the first scheme selected for implementing fault-tolerance in the LUTs of the reconfigurable DSP processor. It was chosen because of its simplicity, in terms of both circuit size and computational complexity. These factors are of the utmost importance in a hardware scheme designed for a small memory.

2.1 Overview

2.1.1 Parity-Bit Calculation and Storage

Error correction is provided by a cross-parity scheme, which uses row and column parity calculations to determine if a bit is incorrect. When a bit is written to the main memory unit, an even-parity bit is calculated for the corresponding row and column. These even-parity bits are then stored in a parity memory unit. When a bit is read from main memory, the stored parity bits for the corresponding row and column are compared with newly calculated parity bits. If the newly calculated row and column parities are inconsistent with the stored versions, then the data bit from main memory is incorrect. If this is the case, it will be inverted in the Correction Unit. A maximum of one bit can be corrected at a time per memory group.

2.1.2 Cell Organization

This error correction/memory system consists of two main components, which are the memory

block and the error correction block. The memory block is made up of the main memory that stores the data, the row and column address decoders, and the memory signal generator, which creates control signals for the system. The error correction block contains memory for storing parity bits during write operations, XOR networks for calculating parities, switches and muxes for routing signals, and a Correction Unit for correcting corrupted bits. The entire system has been designed and simulated using Cadence schematic tools, and then implemented using the Cadence Virtuoso layout editor. A prototype of the system has been fabricated by MOSIS and tested for functionality.

During a write to a block, new parity bits are calculated for the row and column specified by the write address. This is accomplished by combining the data bit to be written with the corresponding row and column, and then XORing each set to generate the parity bits. The XOR of the four row bits is stored as the row parity bit, while the XOR of the column bits becomes the parity bit for that column. When a read is performed, new parity bits are calculated for the row and column indicated by the read address. The newly calculated parity bits are then compared to their stored counterparts. If an error is present at the location of the requested data bit, then the stored parity bits will not match newly calculated values for the corresponding row and column. When this happens, the corrupted bit is corrected and fed back into the memory, permanently fixing the problem.

Storing parity bits for every row and column in an element allows the detection of one error in each row and column. If there is only one error in the block, the location of that error will be identified when a read is performed on the corrupted memory location, as an error will be detected in its row and column. Knowing the location of the corrupt bit makes correction possible. The advantage of having the capability to correct errors in memory is that fault-tolerance is enabled "on the fly." A system that can only detect errors must halt and retransmit data to insure proper operation.

Since a parity bit must be stored for every row and column, eight parity bits are required

per block, and sixteen are required for the LUT. This leads to a 50% memory overhead, compared to the 200% overhead required for triple modular redundancy, in which three copies of each memory bit are kept.

The cross-parity error correction scheme divides the 32-bit element into two parallel 16-bit memory units (arranged in a 4x4 configuration). Two bits of data are sent to or received from an element during every read or write operation. The location of this data is specified by a four-bit address. Each of the parallel 4x4 memory units stores one of these data bits at the position specified by the address. Figure 2.1 shows the hardware organization of the error correction scheme in a memory unit. The error correction system is capable of fixing up to one error in each memory unit during every read operation. In this study, we will examine the operation of a single 4x4 memory unit for the sake of simplicity. An even-parity bit is stored for each row and column in a memory unit during write operations. When a memory read is performed, the parity bits of the selected row and column are XORed with the data in the row and column, which allows the detection of one error in each row and column. If an error is detected in both the row and column of a requested data bit, then the system has located the exact position of the error. Because the position is known, the error can be corrected. The system accomplishes this by inverting the data bit, sending it to the output line, and feeding it back to the memory. This feedback corrects the erroneous bit in memory, allowing multiple errors in the same unit to be fixed, as long as they are all separated by at least one read cycle.

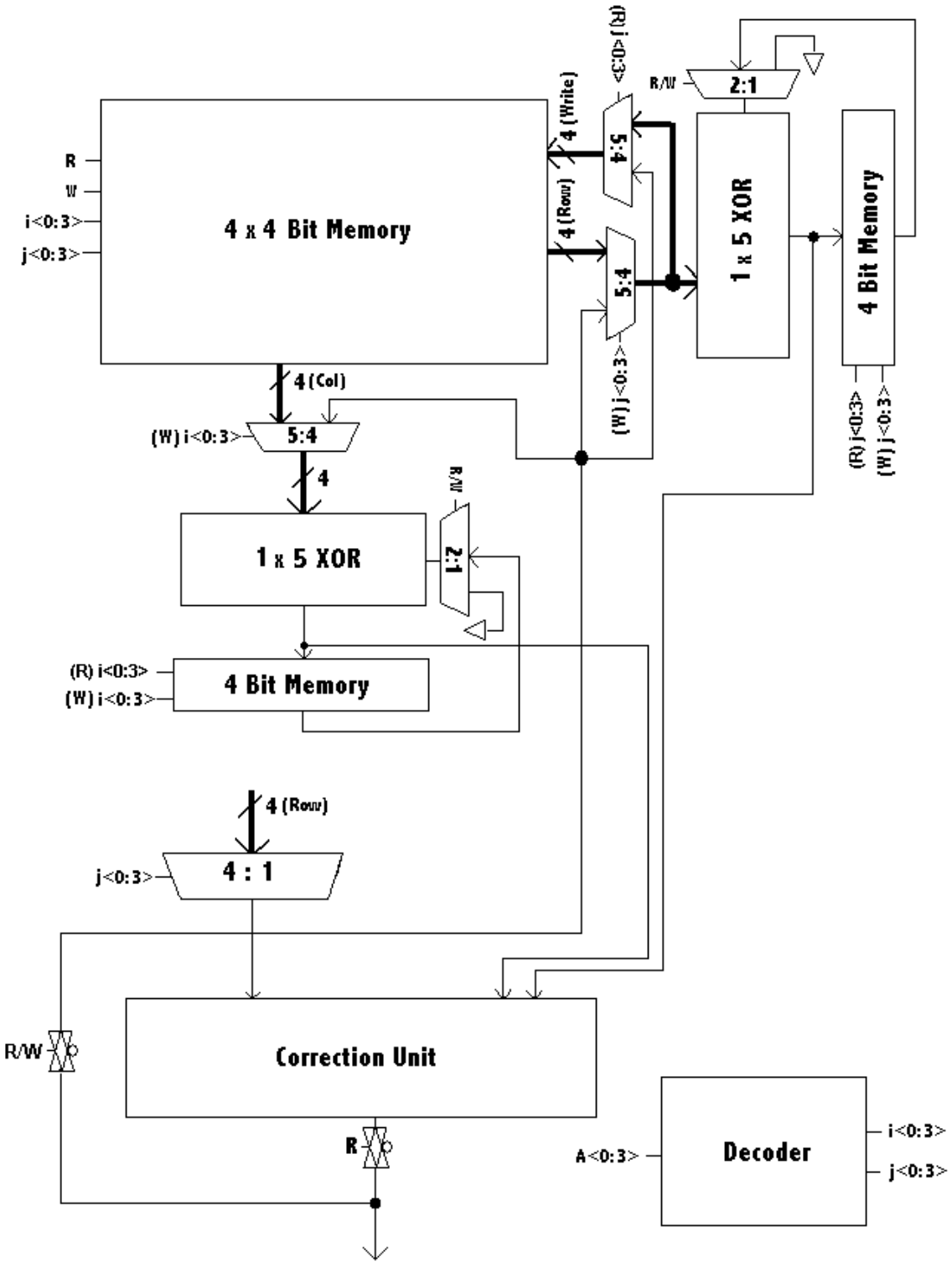

Figure 2.1: Block Diagram of the Cross-Parity System

Figure 2.2 shows the interaction of the hardware of a single fault-tolerant memory block during a read operation. The row and column data selected by the read address are directed to the XOR logic, where they are combined with their parity bits. The row bits are also sent to a multiplexor, which selects the data bit to be read based on the column address bits. If the row and column are inconsistent with the stored parity bits, then the correction unit inverts the data bit. If at least one of the parity bits is consistent, then the data bit is allowed to pass through unaltered. In any case, the output is now available, and is written back to the memory.

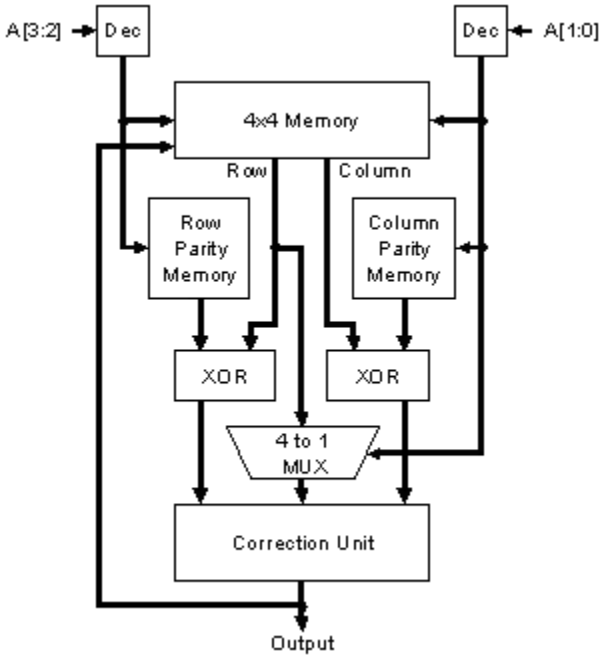

Figure 2.2: Flow Diagram of Hardware During a Read Operation

Figure 2.3 depicts the organization of hardware during a write to the memory. The data bit to be written is XORed with the row and column from memory that are specified by the write address. The outputs of these XOR operations are then stored as the parity bits for that row and column. Also, the bit to be written is stored at the desired location in the memory unit.

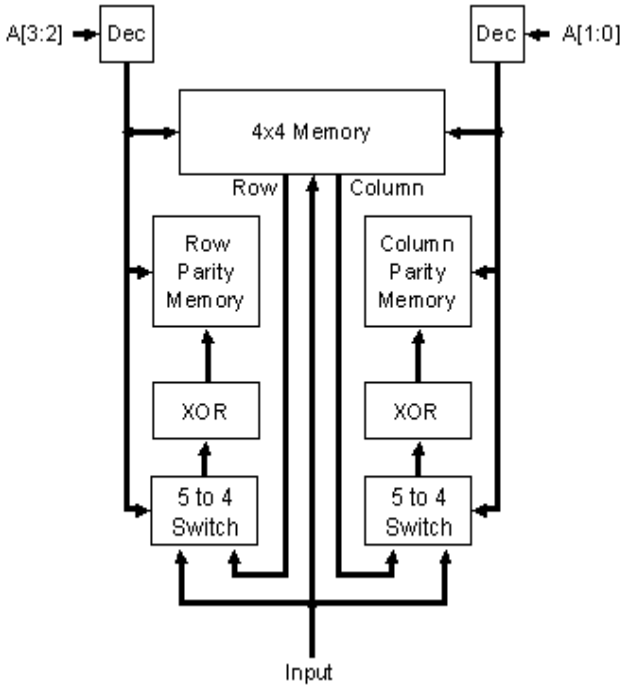

Figure 2.3: Flow Diagram of Hardware During a Write Operation

2.2 Data Path

If a write operation is initiated, a data bit must be supplied to the input of the memory/correction unit. This bit is first sent to three separate five-to-four switches, which combine the data bit with its corresponding row and column. One of these switches is used to route the new column to calculate the column parity bit, and the other switch is used for the row parity bit. The column parity switch receives the data bit and the four bits of the column specified by the write address as inputs. The switch combines the bit to be written with the three column bits that are not in its row. This combination of four bits is sent to an XOR network, which calculates the column parity and writes it to the corresponding location in parity memory. In a similar manner, the row switch also receives the four bits of the corresponding row as inputs. The bit to be written is combined with the three row bits that are not in its column. The resulting four bits are XORed, and then written to main memory and parity memory in parallel.

During a read operation, the decoders decode the address into row and column enable outputs that correspond to the location of the bit being read. The assertion of these outputs cause the entire row and column of the desired bit to be read out of memory, and into the XOR networks. The XOR networks compare the parity bits generated by the row and the column with the bits stored in parity memory. The outputs from these comparisons are sent to the Correction Unit. The four bits of row data that was pulled from main memory are sent to a 4:1 mux, which selects the data bit to be read based on the decoded column information. This bit is then sent to the Correction Unit, which will correct the bit if the parity data indicates that this is necessary. The correct bit is then fed back into main memory via the write bus. If the bit was corrected, the change will be written into the correct cell.

2.3 System Timing

The goal of the system clock approach was to keep it as simple as possible. A diagram of the system timing is presented in figure 2.4. The address decoders evaluate while the clock is low, and their outputs are latched during a high clock. The memory and error correction circuitry evaluate when the clock is high, and have valid outputs latched at the end of a high clock. Correcting an erroneous bit is accomplished through feedback. Corrupted bits are corrected during the read cycle, but are not written back to memory until the following decode cycle because of the problems associated with performing simultaneous memory reads and writes. This forced the adoption of a timing approach that separated these activities. Memory is still read during the memory evaluation phase, which is a high clock. But memory writes have been moved to the next clock pulse, which is a low clock. This was accomplished by delaying the decoded write signals and latching the corrected data for one clock pulse.

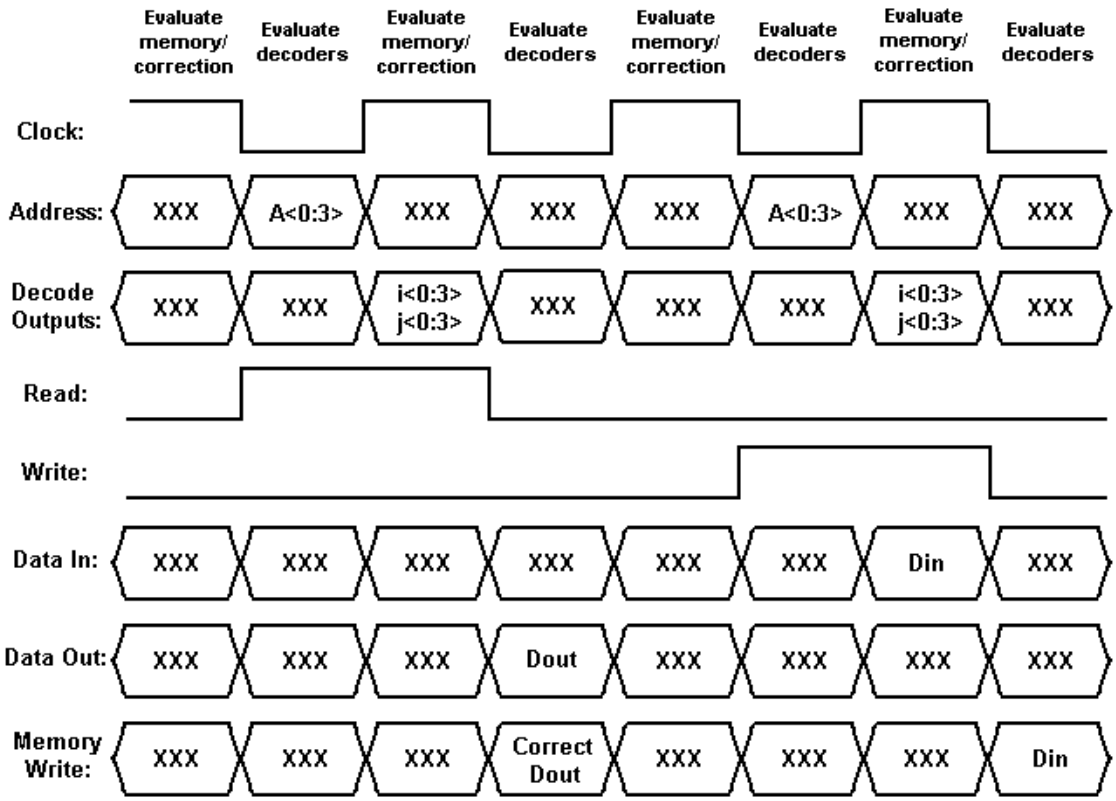

Figure 2.4: Cross-Parity Timing Diagram

Chapter 3

Circuits and Layouts for the Cross-Parity Scheme

CMOS technology has been used to construct the circuitry for this fault-tolerant memory system. The major components in the system include a five-to-four switch, five-input XOR, correction unit, main memory cell, parity memory cell, muxes, address decoder, and a memory signal generator.

In this section, the term M_{ij} is used to represent a bit in row *i* and column *j* of memory. A row of bits in memory is therefore bits M_{i0} M_{i1} M_{i2} $M_{i3,}$ whereas bits M_{0j} M_{1j} M_{2j} M_{3j} represent a column.

3.1 XOR Circuits

Two 5-input XOR circuits perform the parity calculations for the rows and columns in memory. During a write to memory location $M_{ij,}$ these circuits calculate the new parity bits that are to be stored in parity memory. The row circuit calculates $M_{i0} \oplus M_{i1} \oplus M_{i2} \oplus M_{i3}$ for row parity bit *i*, while the column circuit performs $M_{0j} \oplus M_{1j} \oplus M_{2j} \oplus M_{3j}$ for column parity bit *j*. The fifth input of each circuit is tied to ground in this case, as it is not used. When a read of M_{ij} is performed, the current parity of the row *i* and column *j* must be compared with their stored parities. This is accomplished by XORing the four bits from row *i* with stored parity bit P_i, and XORing the bits from column *j* with P_j. This equates to $M_{i0} \oplus M_{i1} \oplus M_{i2} \oplus M_{i3} \oplus P_i$ for the row circuit, and $M_{0j} \oplus M_{1j} \oplus M_{2j} \oplus M_{3j} \oplus P_j$ for the column circuit. A result of zero for either calculation means that no error was detected in the corresponding row or column. If one of the results is one, then an error

has been detected in the corresponding row or column. In any case, the results of both circuits are sent to the correction unit, which determines if it is necessary to correct the bit.

These XOR circuits are constructed out of four two-input XOR gates. These gates are implemented with NMOS pass transistors. The required internal inverse signals are generated by EQV gates, which produce a result opposite to that of an XOR. Inverters are used to restore the voltage levels of the signals passed through the transistors. Figure 3.1 is a graphical depiction of the circuit, while figure 3.2 is the layout. Each XOR gate is formed with a pair of pass transistors. One transistor out of a pair has its drain connected to the first input and its gate connected to the inverse of the second input. The other transistor has its drain connected to the inverse of the first input, and its gate connected to the second input. The EQV gates that generate the internal inverse signals are also formed in a similar manner, but with the gate inputs switched.

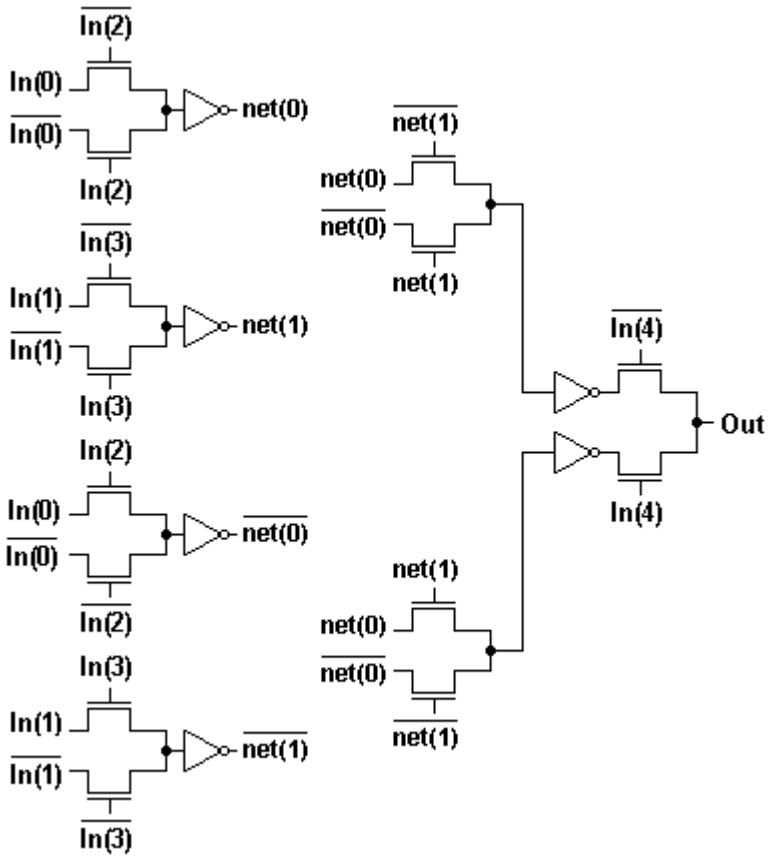

Figure 3.1: Five-Input XOR Circuit

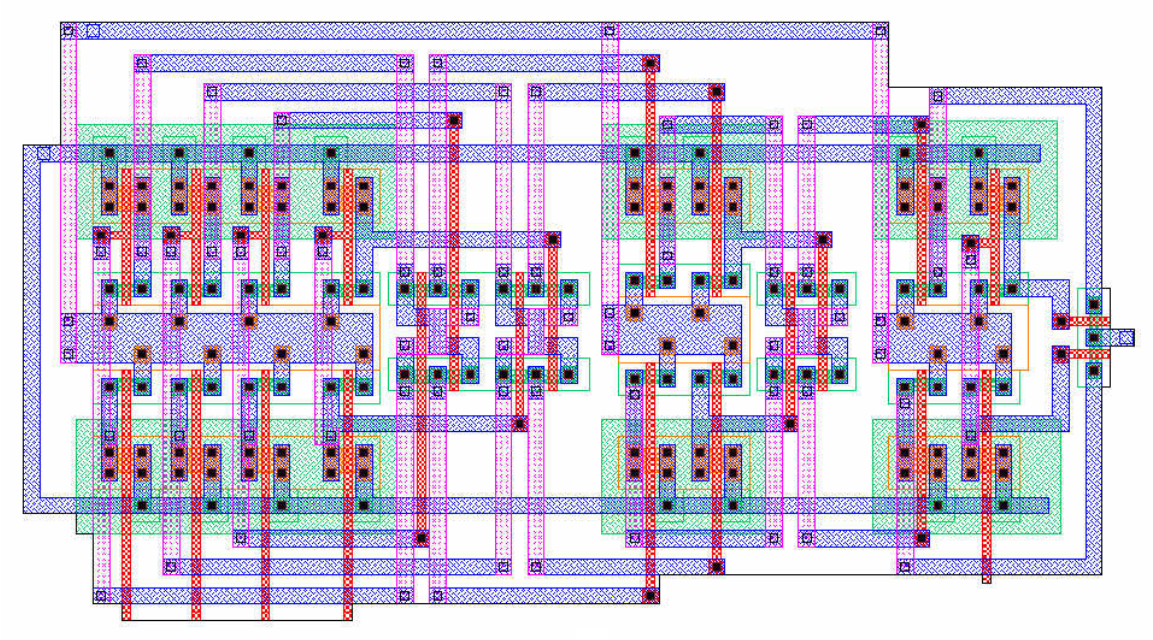

Figure 3.2: Layout for Five-Input XOR Circuit

Although XOR gates are often considered equal to AND/OR gates in circuit complexity analysis, it is difficult to make this estimation a reality. This is due to the fact that XOR is the most complicated binary function. Knowing this, it follows that discovering an optimal VLSI construction of an XOR structure is not trivial.

3.2 Main Memory Cell

Data memory in this system has been arranged into a block with four rows and four columns. This translates into sixteen total memory locations, each of which contains a single-bit main memory cell. Each of these cells must have the capability to read their data whenever its row or column is selected in the data address, as row and column parity bits must be calculated simultaneously. This means that separate row and column read busses are necessary for this system. Every cell in a column is connected to the same column read bus, and every cell in a row is connected to the same read bus. This adds up to four row read busses and four column read

busses. For a given data address (A_3 A_2 A_1 A_0) representing memory position M_{ij}, bits (A_3 A_2) select row *i* and bits (A_1 A_0) select column *j*. When a read of M_{ij} is requested, bits M_{i0}, $M_{i1,}$ M_{i2}, and M_{i3} are presented on the row read busses, while bits M_{0j}, M_{1j}, M_{2j}, and M_{3j} appear on the column read busses. The main memory cells also have separate write busses, which allows a corrected bit to overwrite corrupted data via feedback.

The circuit for a main memory cell appears in figure 3.3, and the layout is in figure 3.4. Its core is a standard pseudo-static latch consisting of two inverters and a feedback path controlled by a pass transistor [11]. Three additional pass transistors connect the cell to the data busses. The transistor that provides the connection to the row read bus has the row's decoded address signal (AD_i) connected to its gate. By the same token, the transistor connecting the cell to the column read bus is controlled by the column's decoded address signal (AD_j). A connection to a read bus is activated whenever a read or a write is performed on row *i* or column *j*. Finally, the transistor that connects the cell to the write bus is activated by W_i, which is a signal from the address decoder that is asserted during writes to row *i*.

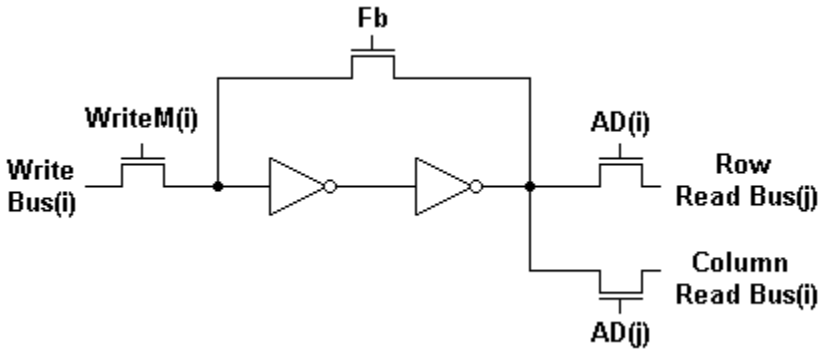

Figure 3.3: Main Memory Cell

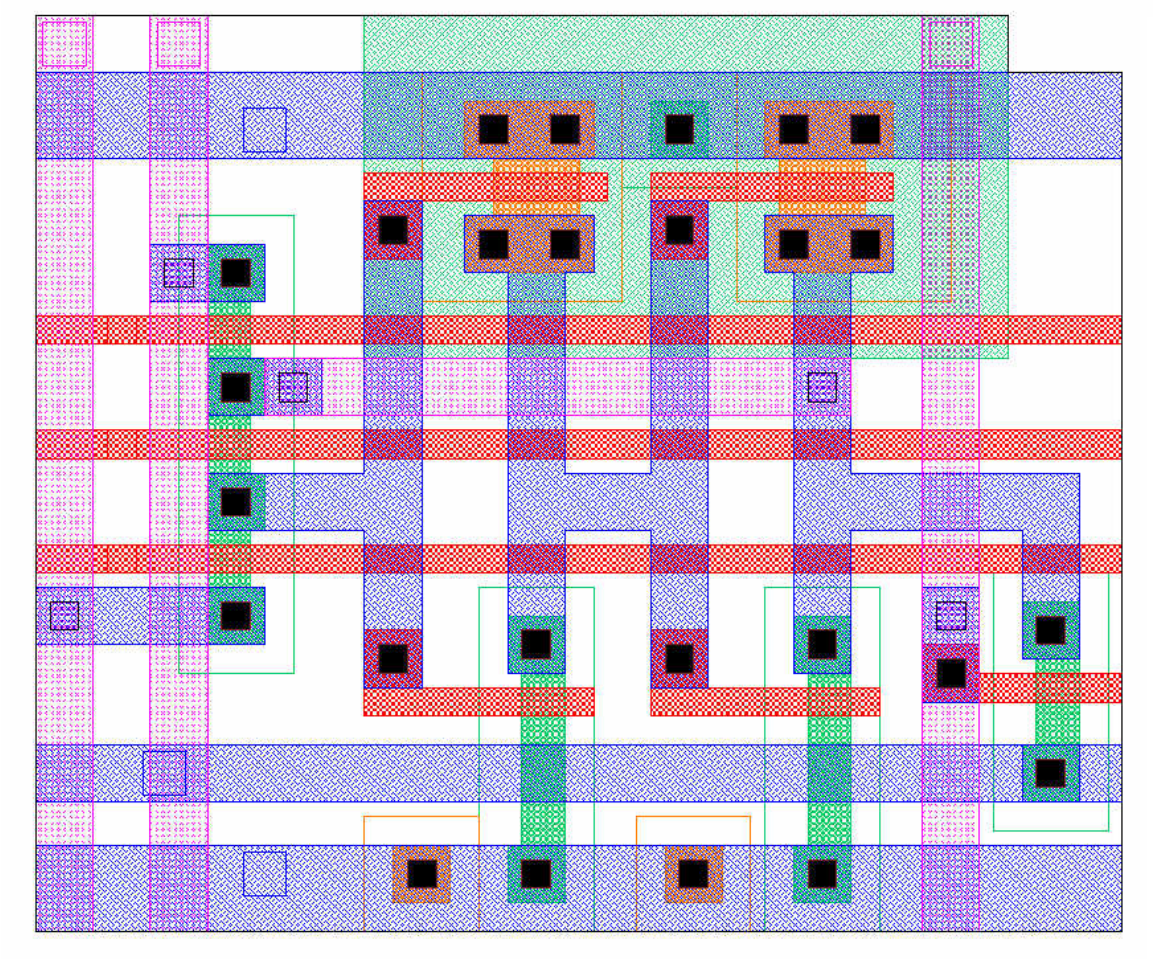

Figure 3.4: Layout for the Main Memory Cell

3.3 Parity Memory Cell

Parity bits are calculated and stored for row *i* and column *j* during a write to position M_{ij} of main memory. This storage capability is provided by two sets of four parity memory cells. One set is provided for the rows in memory, and another is provided for the columns. The parity memory circuit uses only one read bus, instead of the two used in the main memory circuit. This makes the circuits for the parity memory cells slightly simpler than those of their main memory counterparts. The core of the parity memory cell is the same pseudo-static latch. Figure 3.5 depicts the schematic of the parity memory cell. The layout is shown in figure 3.6.

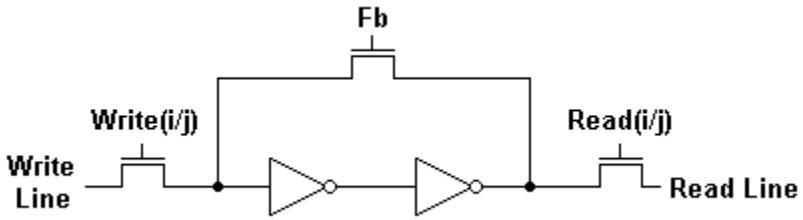

Figure 3.5: Parity Memory Cell

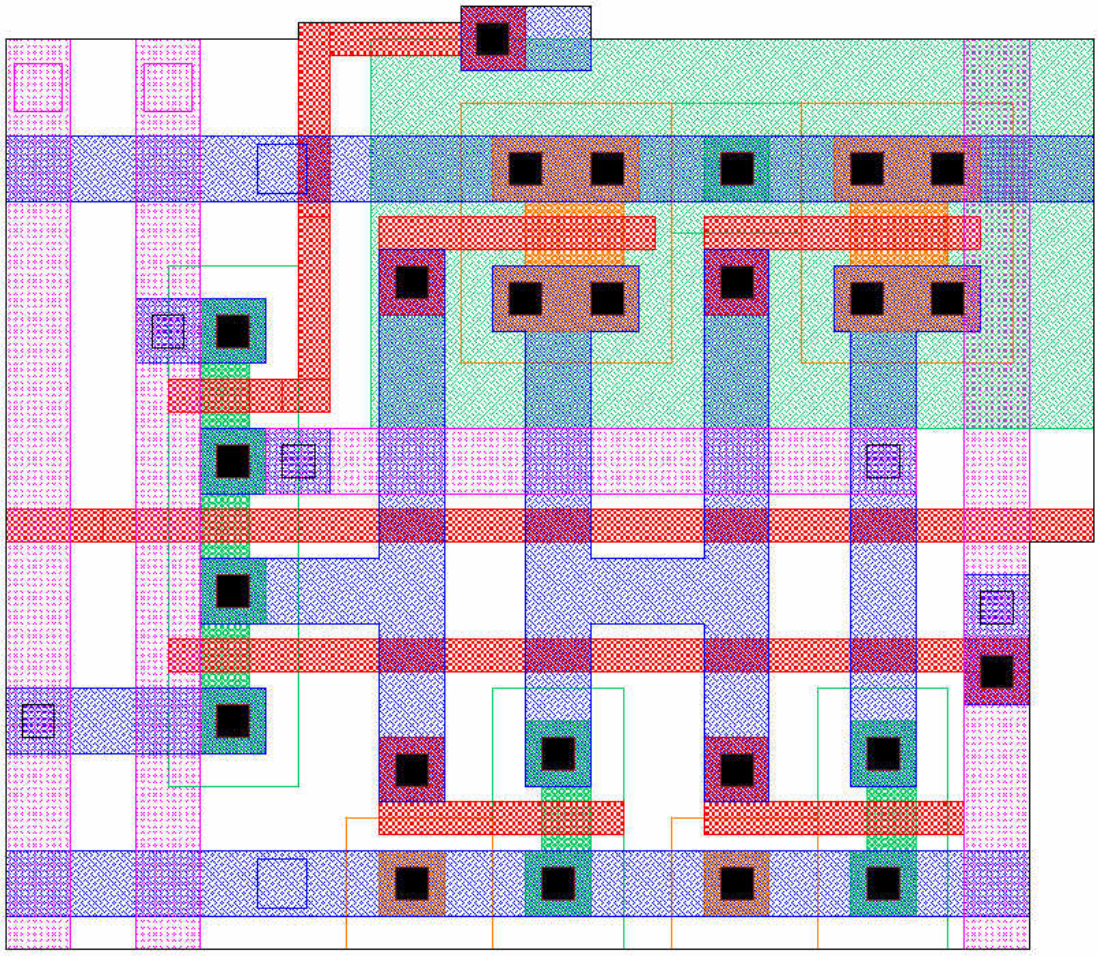

Figure 3.6: Layout of the Parity Memory Cell

3.4 Correction Unit

If a single error is present in the memory unit, its location will be identified when a read is attempted on the incorrect bit. During such a read, both XOR circuits will detect inconsistency between the current parity of the row and column and their stored parity counterparts. When this is the case, it is the job of the correction unit to invert the erroneous bit. The correction unit inverts a data bit M_{ij} only when $(M_{i0} \oplus M_{i1} \oplus M_{i2} \oplus M_{i3} \oplus P_i)$ AND $(M_{0j} \oplus M_{1j} \oplus M_{2j} \oplus M_{3j} \oplus P_j)$ is equal to one. In other words, an error is detected in row *i* and column *j* if both XOR terms are true. Assuming only one error exists in the memory, then the location of the error must be M_{ij}. If both XOR terms are not true, then M_{ij} is deemed correct and allowed to pass through unaltered. In any case, the result of the correction unit is fed back into memory and presented as the output of the entire system.

Figure 3.7 is the schematic for the correction unit, and figure 3.8 is the layout. This circuit receives three inputs: The data bit M_{ij} from memory, and the outputs of the two XOR circuits. The output is the corrected data bit. Bit M_{ij} is connected to the drains of two pass transistors. Both of these transistors have their source connected to the output of the circuit. The results from the row and column XOR circuits are supplied to the inputs of a NAND gate. The output of this gate will be zero when an error has been detected in position M_{ij}, and one otherwise. This signal controls the two pass transistors. There is an inverter between the NAND gate and the pass transistor on the right, assuring that only one of the transistors will be on during steady state operation. The leftmost transistor allows M_{ij} to pass unaltered, while the transistor on the right lets the inverse of M_{ij} through. This scheme controls data bit correction. When M_{ij} is found to be incorrect, the NAND gate turns on the transistor on the right, causing the inverted version of M_{ij} to appear as the output. When this is not the case, the transistor on the left is activated, making the unaltered version of M_{ij} the output.

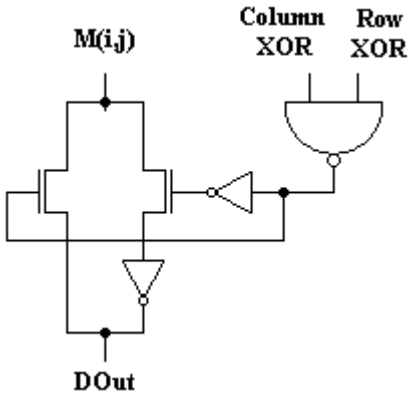

Figure 3.7: Correction Unit

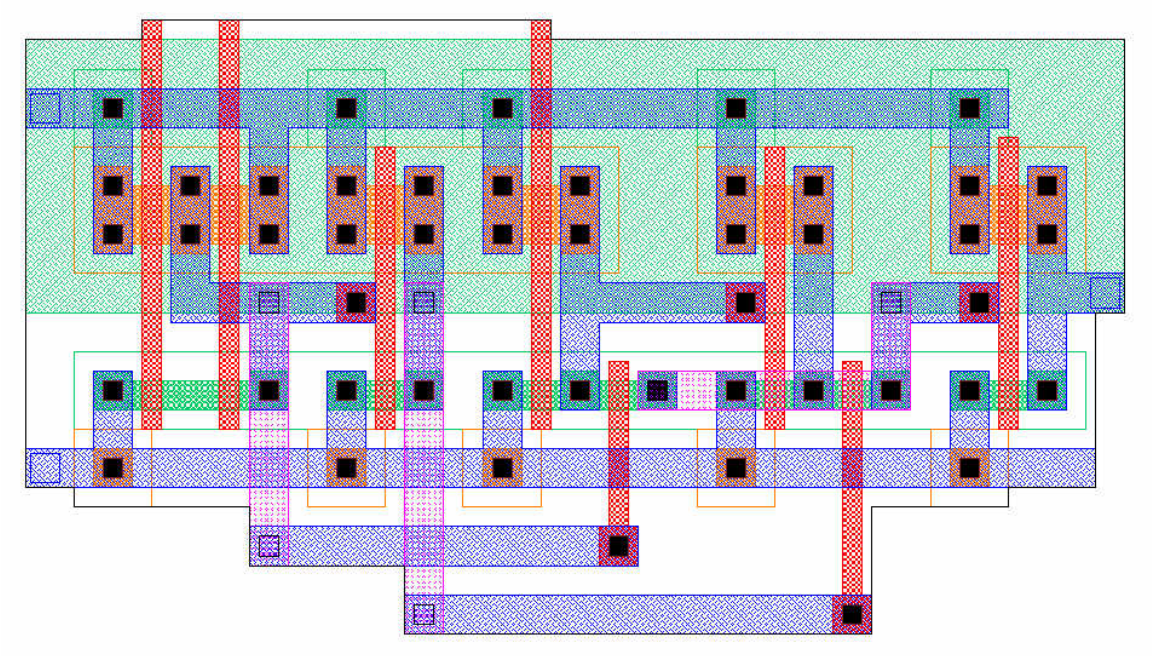

Figure 3.8: Layout of the Correction Unit

3.5 Five-to-Four Switch

Data routing during write operations is facilitated through the use of two five-to-four switches. These switches are transparent during read operations. When a data bit DIn is to be written to location M_{ij} in the memory unit, that bit must be combined with the bits in row *i* and column *j* to allow for calculation of the new parity bits. This is the purpose of these five-to-four switches.

One switch is associated with row data, and it receives bits DIn, M_{i0}, $M_{i1,}$ M_{i2}, and M_{i3} as inputs. The four outputs are DIn (which takes the place of M_{ij}), and the three memory row inputs that are not in the *j*th column. Similarly, the switch associated with column data receives DIn, M_{0j}, M_{1j}, M_{2j}, and M_{3j} as its inputs, and sends out DIn and the three memory column inputs that are not in the *i*th row. The outputs of the switches form the new row and column data, and are used to calculate new parity bits for row *i* and column *j*.

The circuit used for these switches is in figure 3.9, and the layout is in figure 3.10. Pairs of NMOS pass transistors are used to select a signal for each of the four outputs. One transistor out of each pair has its drain connected to Din, while the other transistor is connected to the appropriate memory input signal. Each transistor connected to the DIn signal is controlled the Write(i) or Write(j) signals from the memory signal generator. Conversely, the transistors connected to the inputs from the memory unit are controlled by the inverses of these control signals. When write operations are performed, Din is allowed to pass in place of M_{ij}, while the other memory unit inputs to pass to their respective outputs. During read operations, the switch is transparent, allowing all of the memory unit inputs to proceed to the outputs.

A third five-to-four switch is used during read operations to combine the corrected data bit with the other bits in the selected row, if necessary. This switch receives the column bits M_{i0}, $M_{i1,}$ M_{i2}, and M_{i3} from memory, and also the output of the correction unit, which takes the place of the DIn signal in the figure. The Read(j) control signals are used to identify the bit position that DIn should replace. If the data bit M_{ij} that is requested from memory is correct, DIn will be equivalent to M_{ij} and this switch will effectively perform no operation. However, if M_{ij} is found to be incorrect, DIn will be the opposite of M_{ij}, and this corrected bit will be written back into memory along with the rest of row *i*.

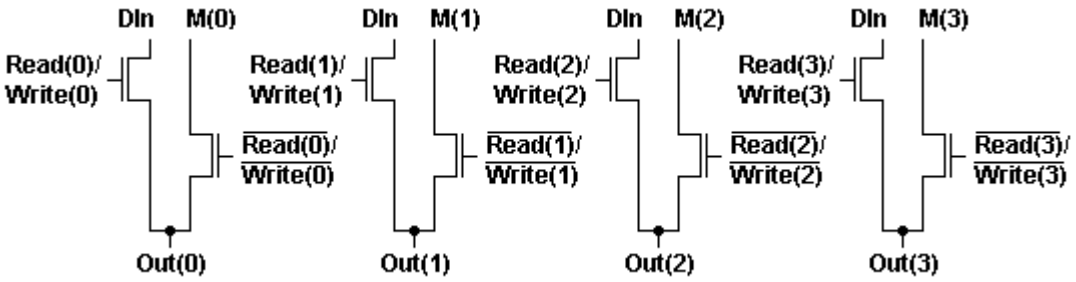

Figure 3.9: Five-to-Four Switch Circuitry

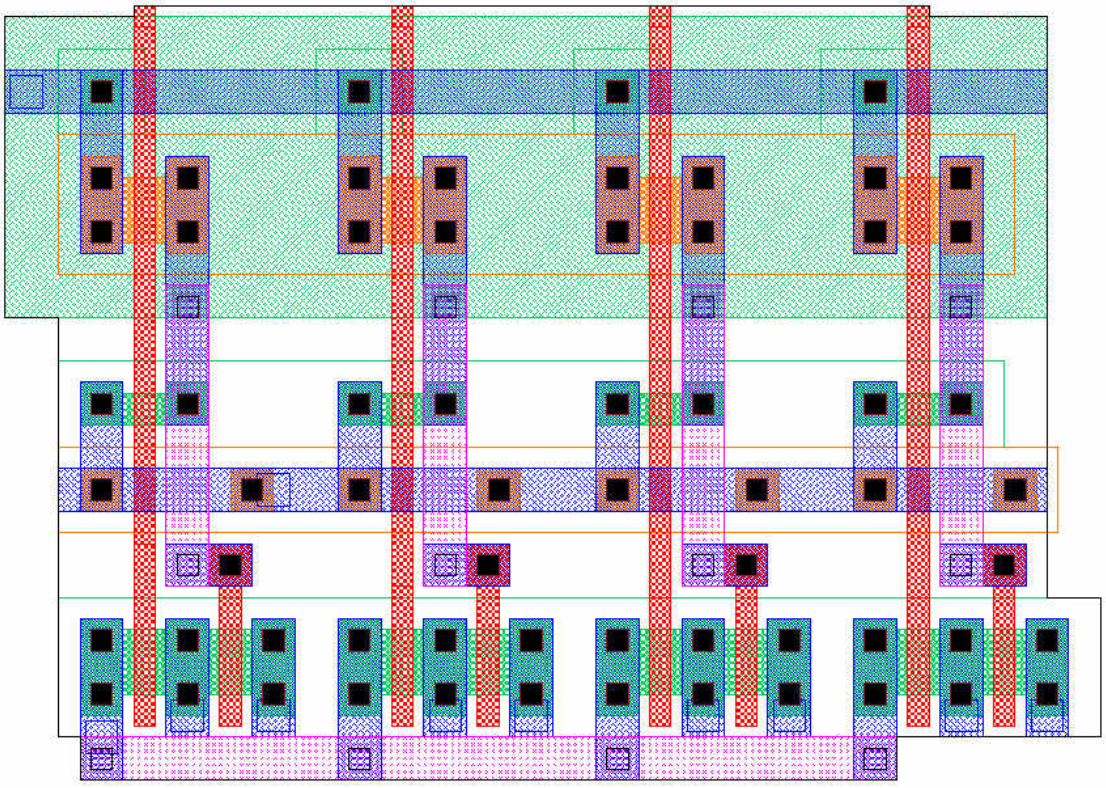

Figure 3.10: Five-to-Four Switch Layout

3.6 Muxes

Additional data routing is facilitated by the use of a four-to-one mux and two-to-one muxes. During read operations, the four-to-one mux receives the row data specified by the read address. It passes on the data bit that is desired by the read operation, which is determined by column read address. The two-to-one muxes are used to pass a logic zero to the fifth input of each XOR network during write operations. This is necessary because only the four bits of a row/column need to be XORed during write operations, whereas the stored parity bit for the row/column must also be XORed during read operations. Passing a zero to the fifth input of the XOR networks effectively takes it out of the circuit. If a read operation is requested, the two-to-one muxes will pass the stored parity bit for the desired row/column. All muxes in this system are implemented with pass transistors. The schematics of the muxes are shown in figure 3.11.

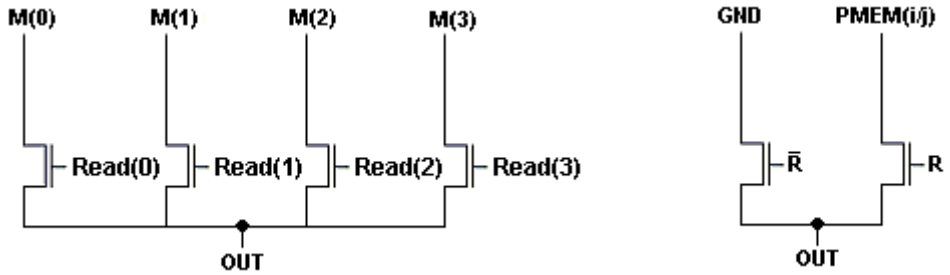

Figure 3.11: Four-to-One and Two-to-One Muxes

3.7 Decoder and Memory Signal Generator

Control signals must be delivered to the desired components during the correct clock cycle. Master-slave latches and a two-phase clocking scheme are responsible for the timing in this system. This setup allows signals to be saved and used a number of clock cycles later. Neighboring stages in a master-slave latch operate with different clock phases, which ensures that a signal can not skip any stage of the latch. The two-phase clocking scheme is generated by

applying a 50% duty cycle pulse to a cross-coupled NAND gate pair. Only one phase is active at a time. A small period of inactivity separates changes in active states. This inactivity prevents active state overlap, which can be caused by clock skew. Master-slave latches fail when active states in the two phases overlap.

The address decoders used in this system are two-to-four NOR decoders. The same decoder circuitry is used for both the row decoder and the column decoder. They are implemented with CMOS logic. NOR gates are used in the decoders instead of AND gates because their delay is an inverter length shorter. The assertion level of each input to a NOR gate is the opposite of the assertion level that would be used in an AND decoder. The schematic of the address decoders is illustrated in figure 3.12.

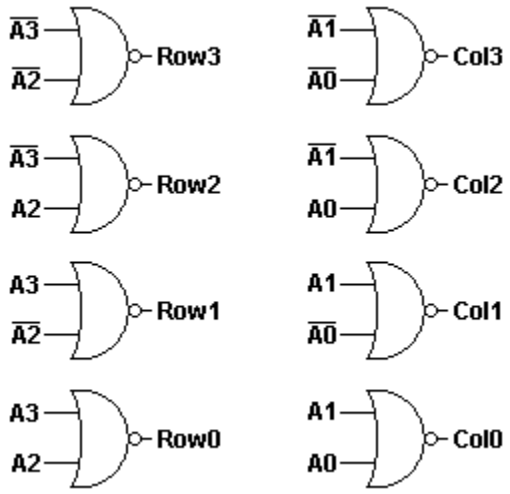

Figure 3.12: Row and Column Decoders

The memory signal generator (MSG) generates the control signals for the cross-parity system. The MSG receives the outputs of the row and column decoders, and calculates the signals needed to operate the circuitry in the system's data path. The main-memory cells require memory write signals (WriteM(i)), decoded row (AD(i)) and column (AD(j)) signals, and a feedback signal

(Fb). For the parity-memory cells, write row (Write(i)) or write column (Write(j)) signals, read row (Read(i)) or read column (Read(j)) signals, and the feedback signal (Fb) are required. For data routing, the two five-to-four switches used during write operations need the write row (Write(i)) or write column (Write(j)) signals. The third five-to-four switch, which is used during read operations, receives the read column (Read(j)) signals. The four-to-one mux requires the read column signals (Read(j)), and the two-to-one muxes need the read (R) signal. Figure 3.13 displays the schematic of the memory signal generator.

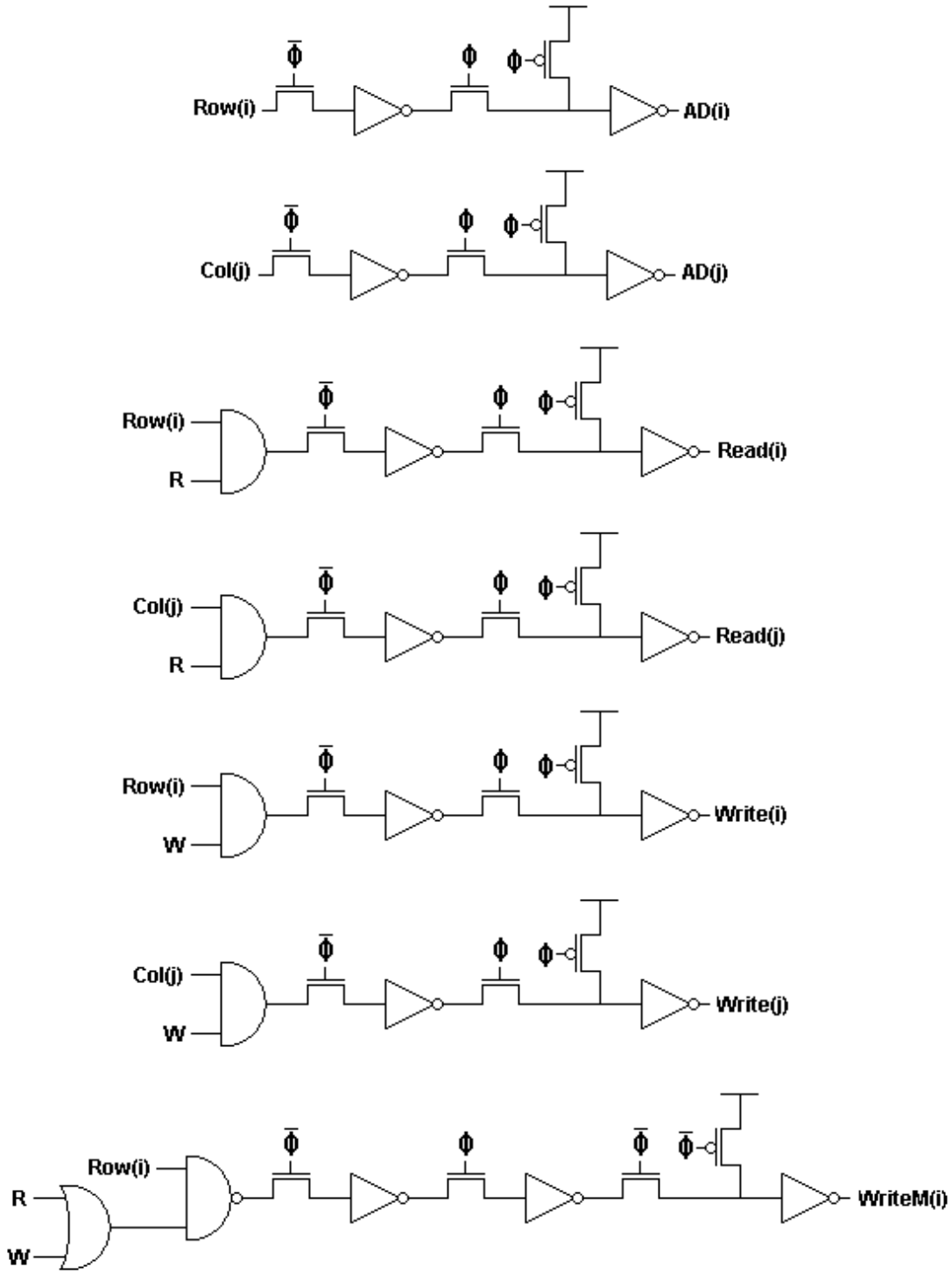

Figure 3.13: Memory Signal Generator Schematic

Chapter 4

Simulation of the Cross-Parity Scheme

All of the components of the cross-parity system have been designed in Cadence and simulated in Spectre using a .5μm process. The simulation results will be analyzed in this section.

4.1 XOR Network

An exhaustive simulation of the XOR network has shown that it operates as expected. Every possible combination of the five inputs (In0, In1, In2, In3, In4) has been included in the simulation. The "Out" signal is the output. It goes high whenever an odd number of inputs are asserted, and is low otherwise. The highest propagation delay of this circuit was found to be 686ps. Glitches can be observed in the outputs during input transitions, but they do not cause any detrimental effects in the overall system, as the output is passed on after all glitches have subsided. The simulation of the XOR network is shown in figure 4.1.

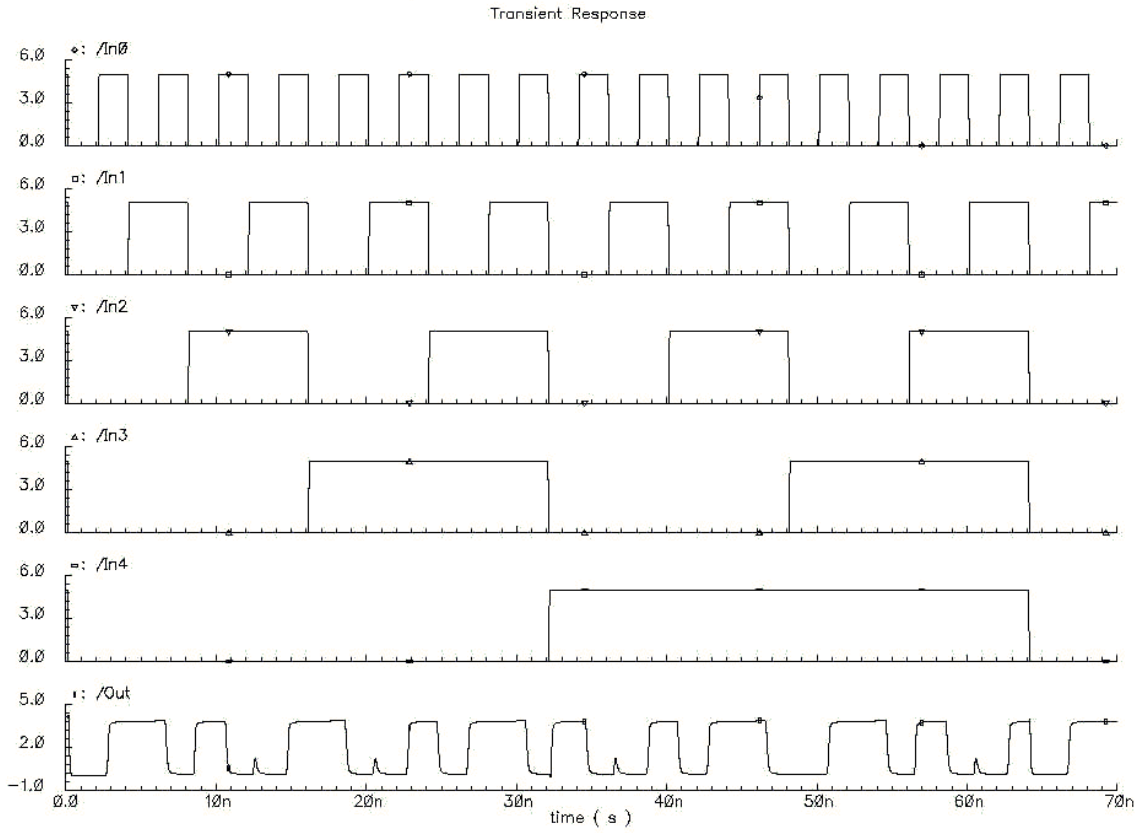

Figure 4.1: XOR Network Simulation

4.2 Main Memory

The main memory cell has separate row read, column read, and row write buses, as well as row read, column read, write and feedback inputs. The inputs shown on the simulation are the row and column read signals (ReadRow, ReadCol), the write signal (Write), the feedback signal (Fb), and the write bus input (WBus). The outputs are the row and column read busses (RRowBus, RColBus). Stage one of this simulation shows a write of a logic one to the memory. The next portion of the simulation shows the reading of the stored value onto the row and column read busses. After this, the memory write bus is forced low in the simulation, and the write signal is asserted. The row and column read signals are individually asserted in the following clock cycles. The memory cell responds to this properly in the simulation, as it stores the new bit value

and then sends it to the read busses. It took 38ps to propagate a read onto both busses. Figure 4.2 shows the simulation of the main memory cell.

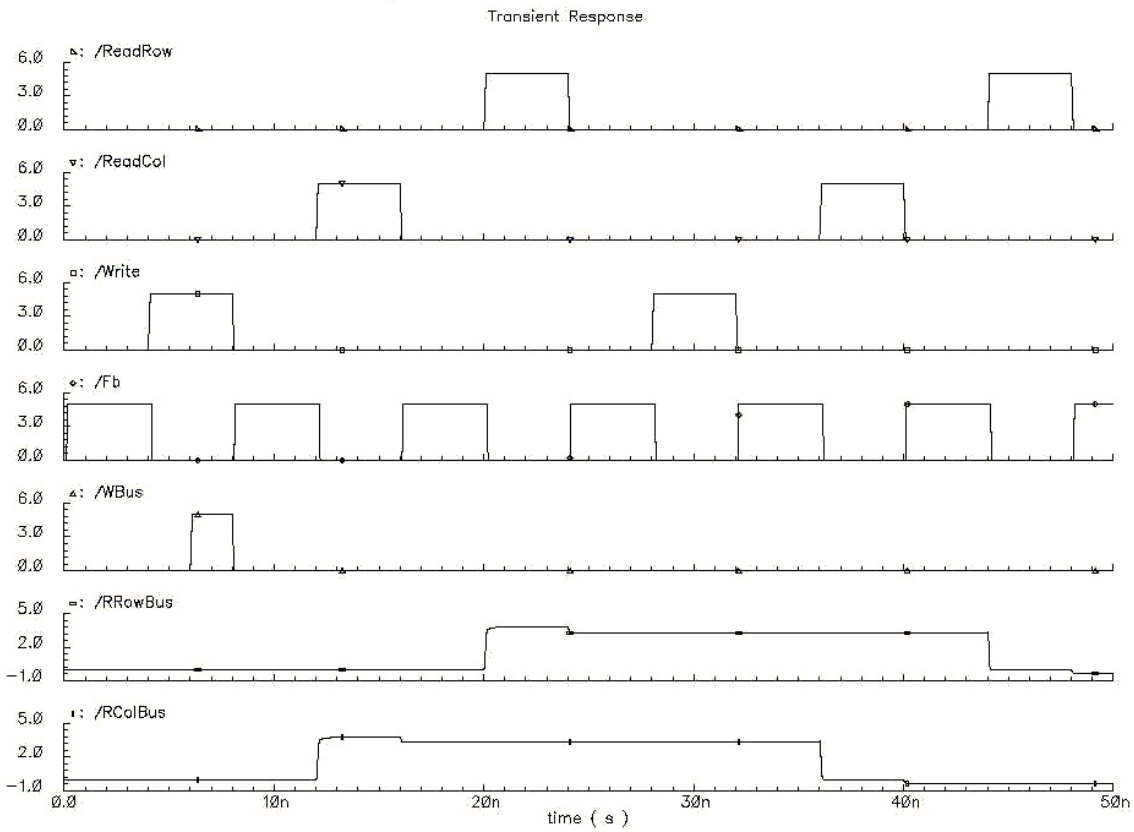

Figure 4.2: Main Memory Cell Simulation

4.3 Parity Memory

The parity memory cell simulation was similar to that of the main memory cell, but without the extra features. The inputs are read and write signals (Read, Write), a feedback signal (Fb), the write bus (WBus). The output is the read bus (RBus). The layout of this cell uses separate read and write busses to allow for more control of the circuit during simulation. A value of logic one is written to the cell first, and it is read back successfully. A zero is written to the cell after this, and is also read back without failure. The largest read propagation delay was found to be 32ps.

Figure 4.3 depicts the simulation of the parity memory cell.

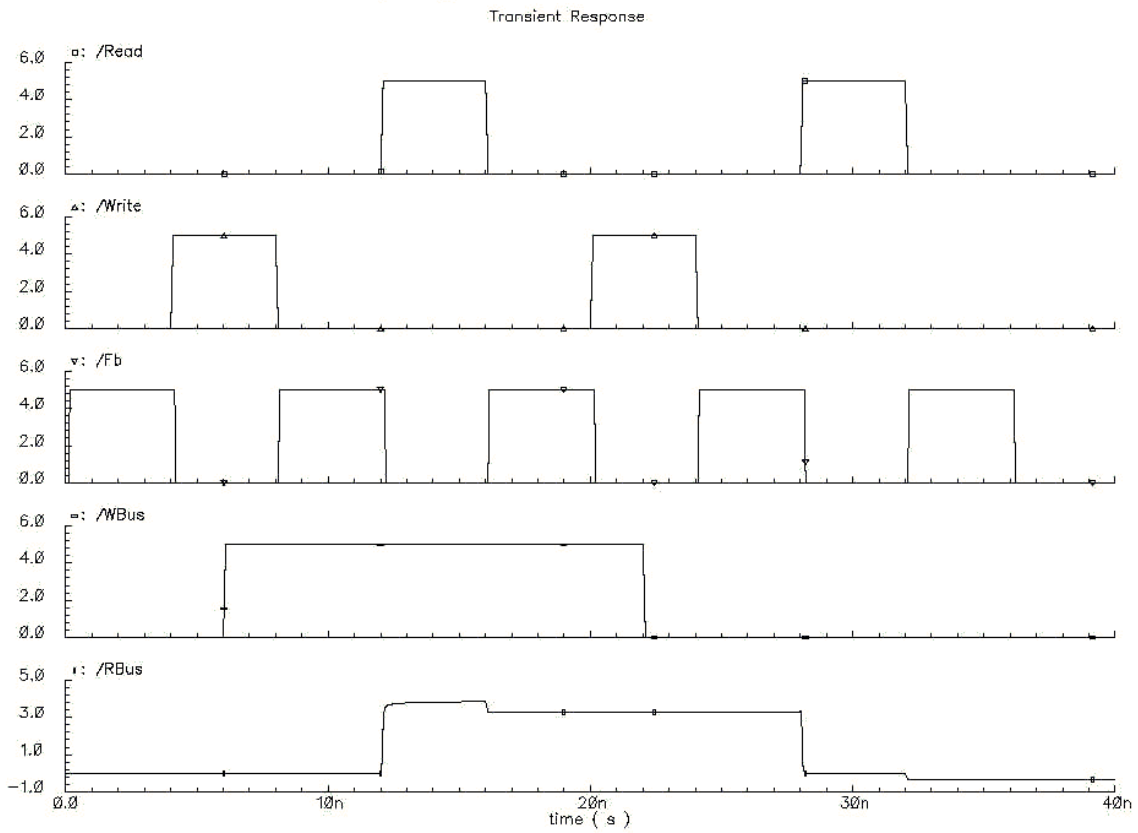

Figure 4.3: Simulation of the Parity Memory Cell

4.4 Correction Unit

The Correction Unit simulation shows the data input signal (D_In), the row and column parity input signals (Row_P and Col_P), and the data out signal (Out). The data input is the bit being read from memory, and the parity signals come from the outputs of the XOR network. If the row or column in main memory that corresponds to the data bit fails a parity check, then a bit in that row or column was found to be incorrect. If this is the case, then the row or column parity bit will be high. If the row and column both fail their parity check, then the Correction Unit decides that the input data has been corrupted, and so it corrects it by inverting it. If at least one of the

parity bits is not high, then the Correction Unit allows the bit to pass unaltered. The simulation shows that this circuit functions as it should, as the data bit is corrected only if both parity bits are high. The longest propagation delay was 373ps. The simulation of the correction unit is shown in figure 4.4.

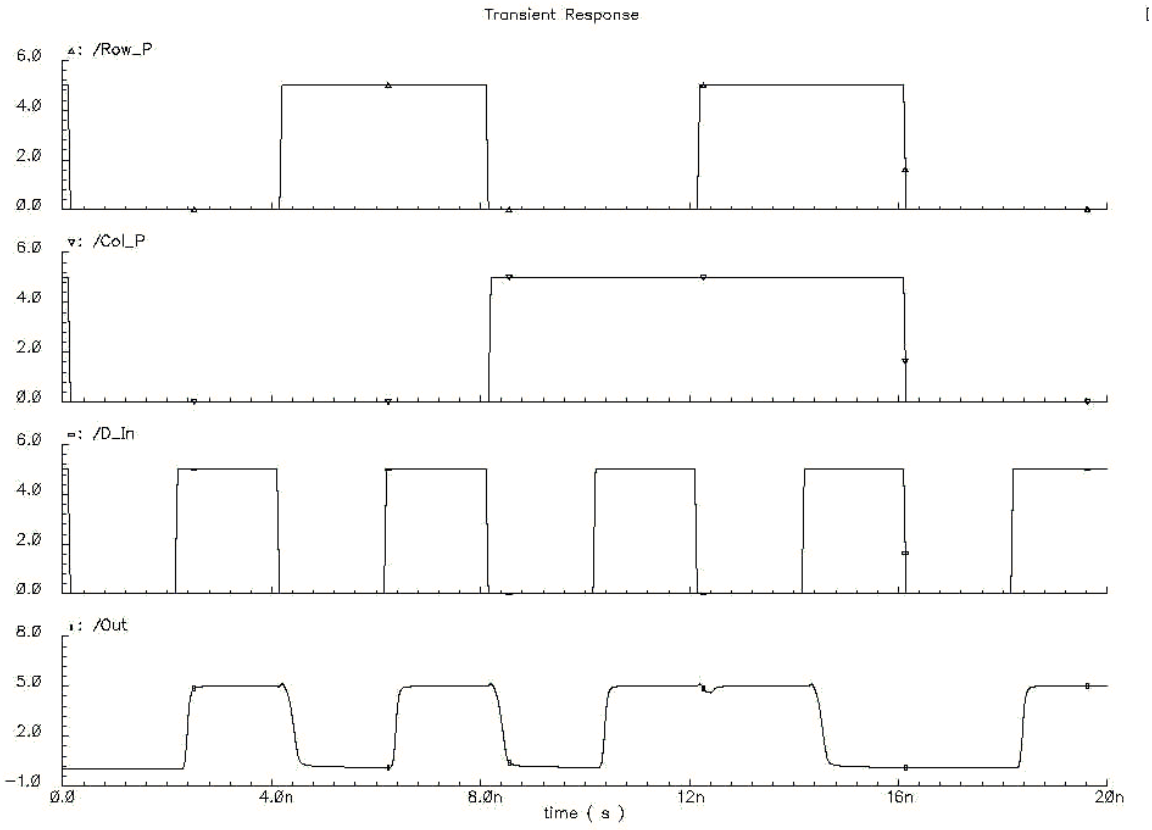

Figure 4.4: Correction Unit Simulation

4.5 Five-to-Four Switch

The five-to-four switch is used during write operations and memory correction, where it combines the data bit to be written into main memory with the three other bits in the corresponding row/column. The simulation for this switch includes an input signal for the new data bit (Dnew), four inputs for the corresponding row/column from main memory (Dmem1, Dmem2, Dmem3, Dmem4), four decoded row/column input signals (S1, S2, S3, S4), and four

outputs (Out1, Out2, Out3, Out4). The new data input signal pulsed throughout the simulation, while the four inputs from main memory are set at different logic levels, which allows them to be distinguished from one another. Dmem1 is 0, Dmem2 is 1, Dmem3 is 0, and Dmem4 is 1. Each decoded row/column signal is asserted individually. While one of these select signals is high, Dnew is allowed to pass through to the corresponding output line, while the respective Dmem signal is tri-stated. All of the other Dmem signals are connected to their corresponding outputs. In the simulation, the period of the Dnew signal is equal to the time that each select signal is asserted. This allows the detection of Dnew in the output signals, as all of the Dmem inputs remain constant. It can be observed from the simulation that Dnew is always passed to the correct output line, while all other outputs remain connected to their corresponding Dmem input pins. The longest propagation delay of this circuit was 93ps. Figure 4.5 presents the simulation of the five-to-four switch.

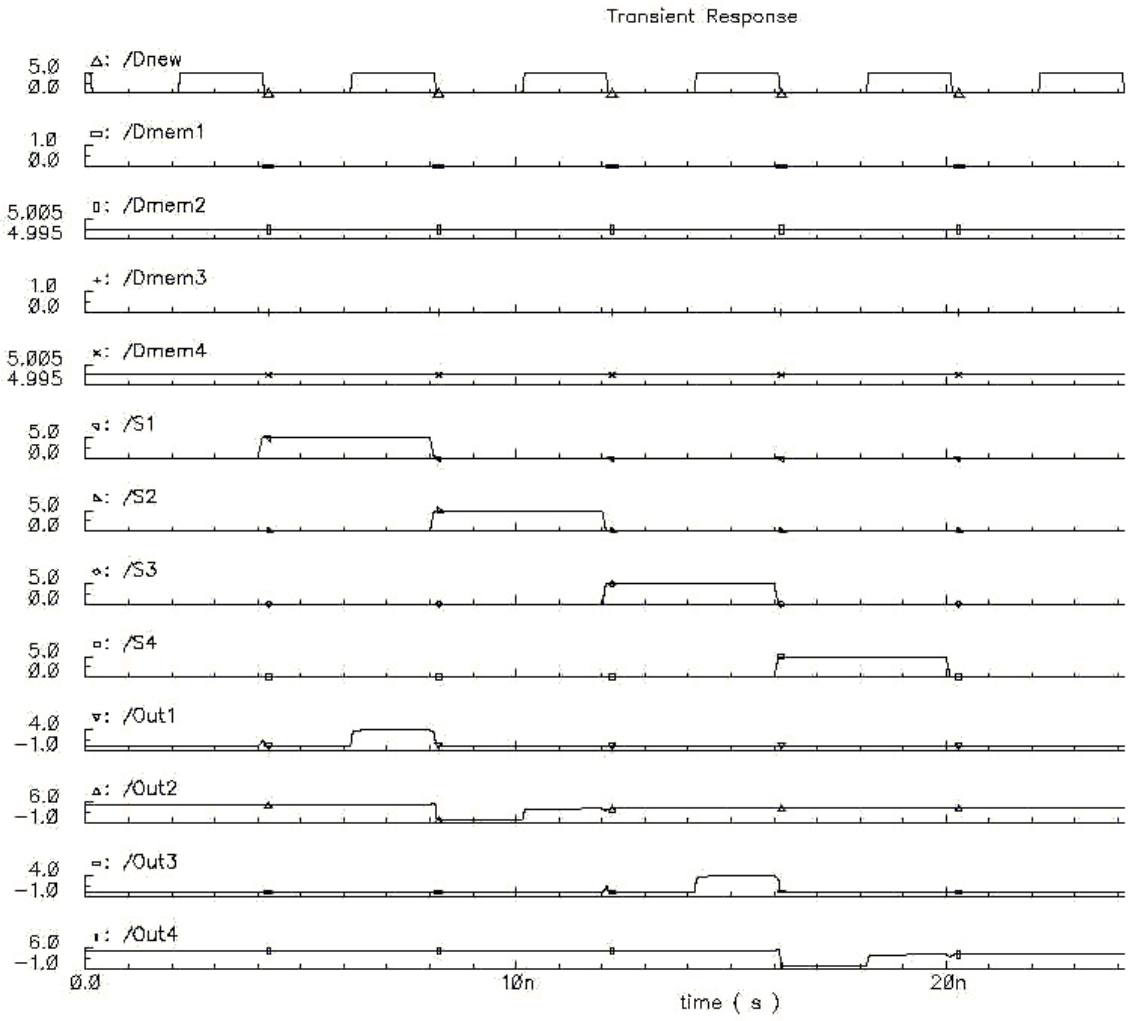

Figure 4.5: Simulation of the Five-to-Four Switch

4.6 Muxes

Because of the similarity between the four-to-one mux and the two-to-one mux, only the four-to-one mux will be examined here. In the simulation of the four-to-one mux, each of the four decoded column signal inputs (s1, s2, s3, s4) is asserted individually. The four data input signals (In1, In2, In3, In4) are initialized to logic zero. These data inputs are allowed to pass through the mux when their corresponding select signal is asserted. Only one select signal is allowed to be high at a time. When selected, a data input signal stays low for half of its selection window, and

then goes high for the other half. The output of the mux shows that the data signals were correctly selected as directed by the column input signals. The longest propagation delay was found to be 24ps. Figure 4.6 shows the simulation of the four-to-one mux.

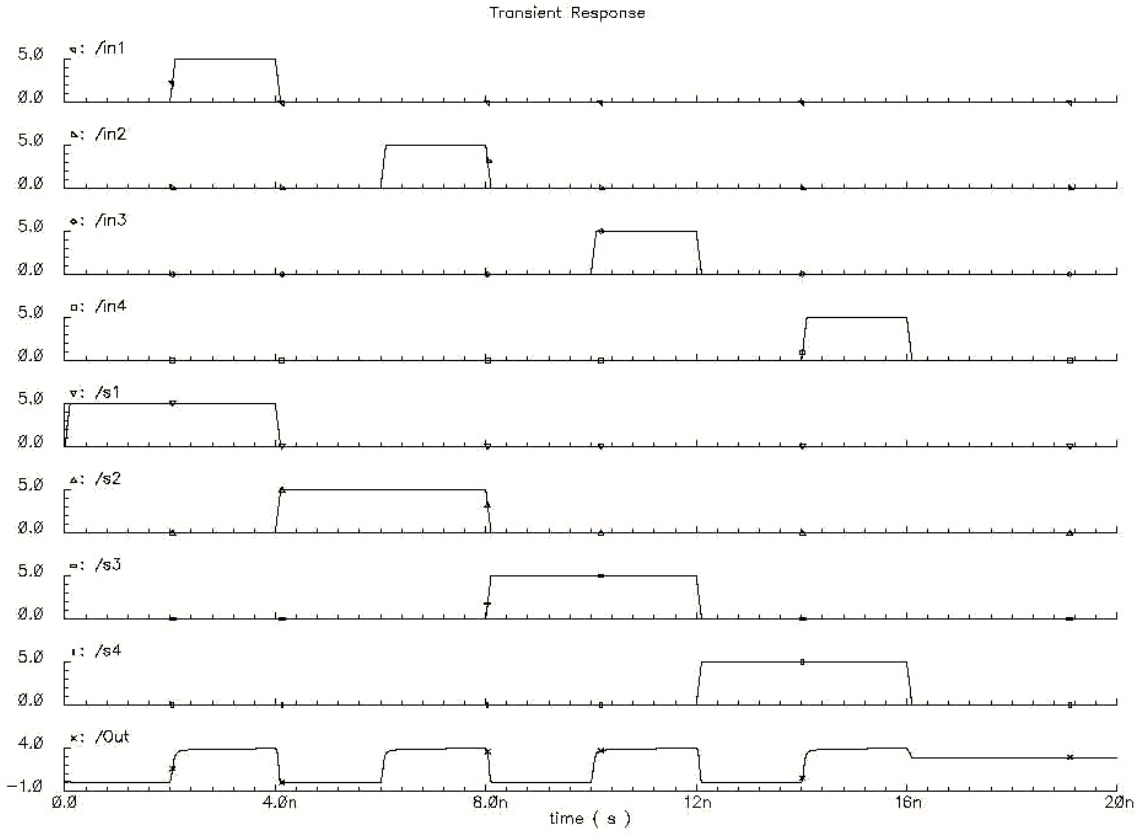

Figure 4.6: Simulation of the Four-to-One Mux

4.7 Decoder and Memory Signal Generator

Layouts for the address decoders and memory signal generator have been combined into one unit in this report. The address decoders decode the row and column address bits, and the memory signal generator creates control signals for the memory and error correction systems. Two simulations are included in this report: One for the main memory control signals, and another for the control signals of the error correction system. The clock (Clk), row address (A3, A2), and

read/write signals (Read, Write) are the inputs that have been included in both simulations. The outputs for the memory control simulation are the row read signals (Row3, Row2, Row1, Row0), and the row write signals (WMRow3, WMRow2, WMRow1, WMRow0). This unit also has column read outputs, but they are not included in this simulation for the sake of simplicity. Both the row and column read signals tell the rows and columns of main memory to read their data onto a bus, while the write signals tell the rows to write data. If selected, the read signals become active during the next high clock phase, while the write signals are activated during the following low clock phase. The simulation of the control outputs for error correction system includes row read (RRow 3, RRow 2, RRow 1, RRow 0), and row write (WRow3, WRow2, WRow1, WRow0) outputs. Column control outputs for the error correction system exist, but they were omitted from this simulation. The row and column read and write signals control the switches, muxes, and parity memory in the error correction system. Without these signals, the system would not know which bit in data memory it is trying to correct.

An observation of both simulations shows that all the output signals go high during the appropriate clock phase. All possible combinations of row input signals are analyzed in the simulations. The largest propagation delay of the decoder was 360ps. Figure 4.7 is a simulation of the main memory control signals generated by the decoder and MSG. A simulation of the control signals for the error correction system is shown in figure 4.8.

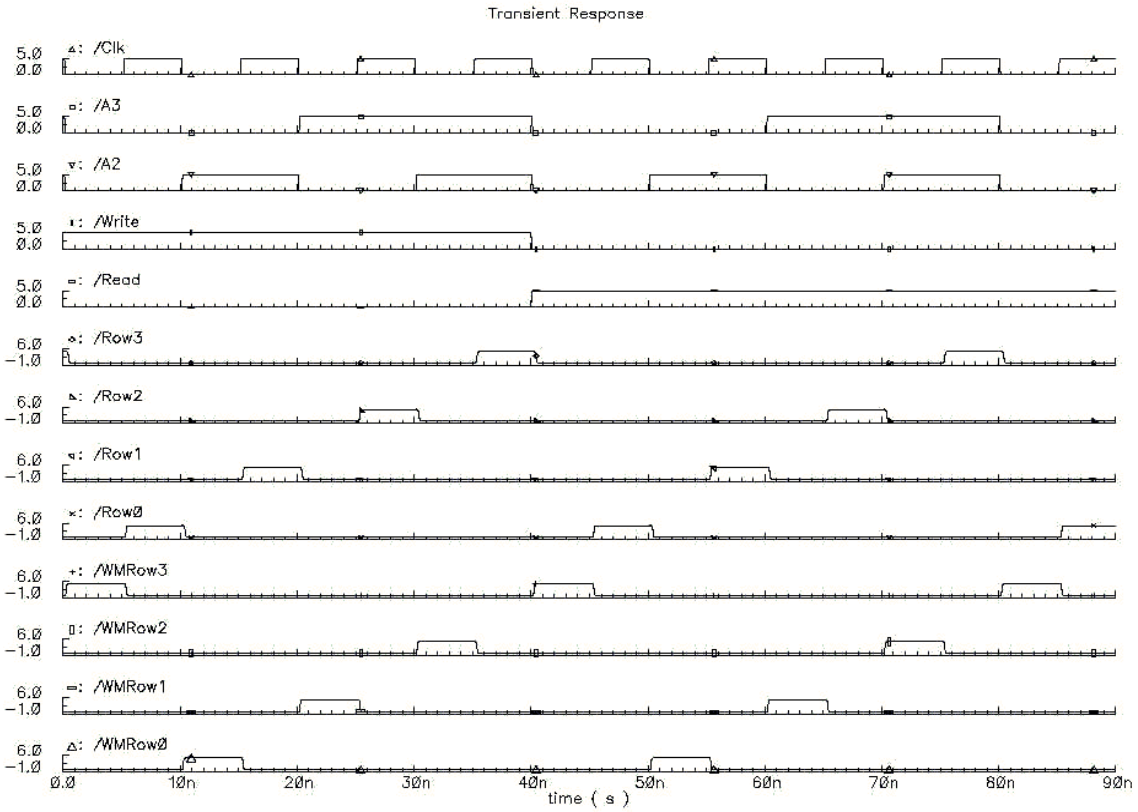

Figure 4.7: Simulation of the Main Memory Control Signals Generated by the Dec and MSG

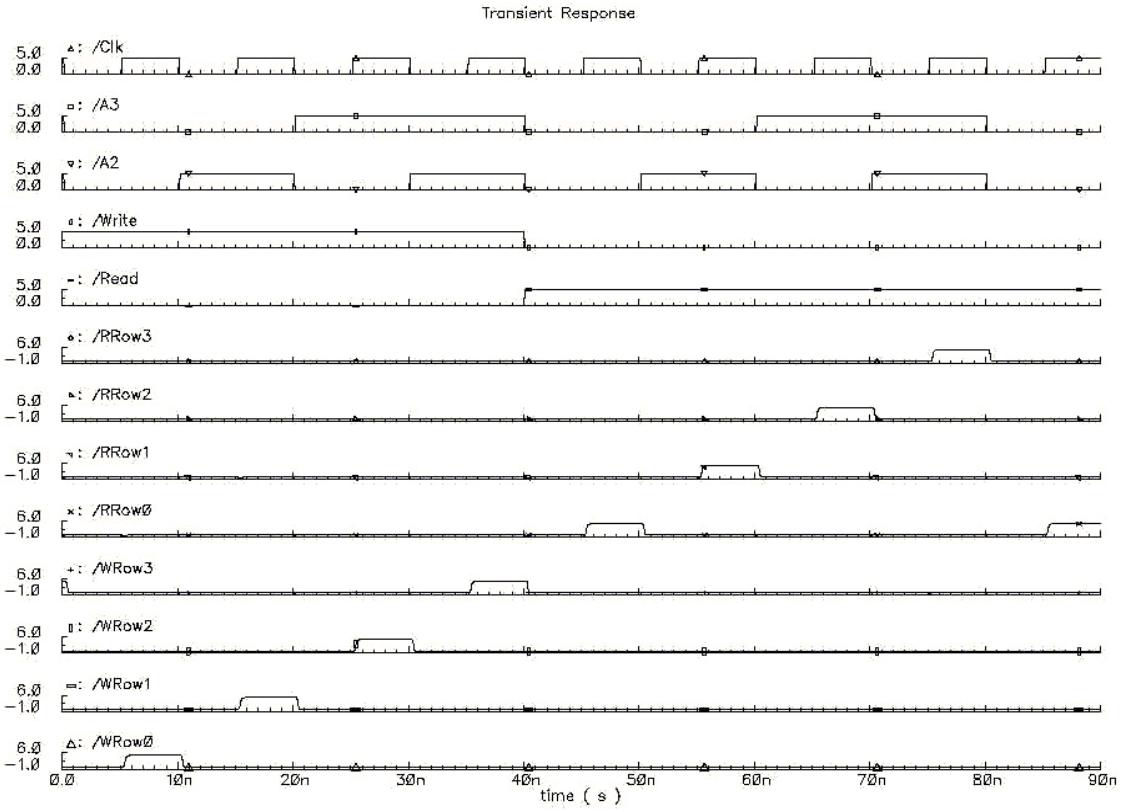

Figure 4.8: Simulation of the Error Correction Control Signals Generated by the Dec and MSG

4.8 System Simulations

Four simulations of the entire cross-parity correction system are included in this section. The first simulation analyses performance, while the others verify the functionality of the system. In the first simulation, the worst-case read and write delays are compared to the best-case read delay. The second simulation shows writes to all sixteen cells in the memory, followed by reads of those cells. The third simulation shows two separate memory writes to cell (0,0), each followed by an error, and then corrected memory reads. Finally, the fourth simulation depicts reads and writes to every cell in a layout of two memory/error correction systems (which forms a full element LUT). The results will be analyzed in the following paragraphs.

The performance of the cross-parity system is detailed in the first simulation, which is shown in figure 4.9. Writing data to memory was found to be the operation that required the most

time to complete. In the worst case, the propagation delay of a write was 4.13ns. This propagation delay was measured from the 50% point of the rising clock edge to the 50% point of the valid bit appearing in parity memory [11]. The performance of writes suffers because of the number of sequential functions they require. When bit DIn is to be written to location M_{ij} of memory, row *i* and column *j* must be read from memory. After this, DIn, row *i* and column *j* must be combined in the five to four switches. The XOR circuits then calculate new parity bits with the outputs of the switches. During this time, DIn is stored at location M_{ij} in memory. Finally, the new parity bits are stored in parity memory.

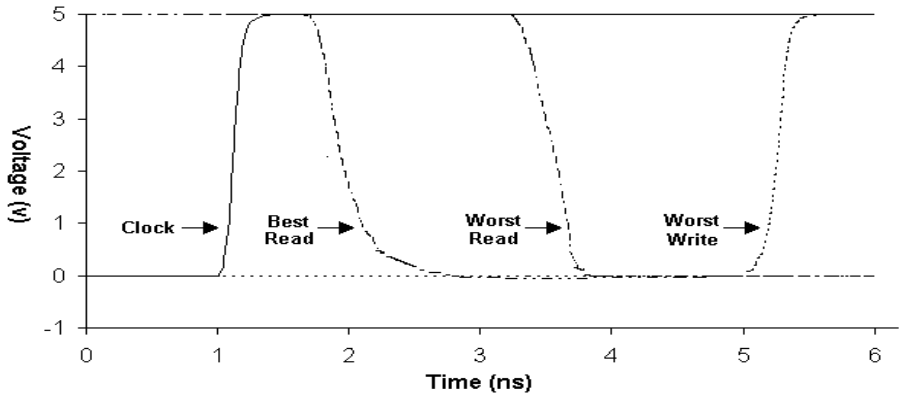

Figure 4.9: Simulation of the Cross-Parity System Highlighting Propagation Delays

Read operations execute more quickly than writes because their critical path is shorter. The largest propagation delay of a read was found to be 2.42ns. This occurred when a read of bit M_{ij} revealed that it was corrupted. First, row *i* and column *j* were read from memory and passed to the XOR circuitry. At the same time, P_i and P_j were read from parity memory and sent to the XOR circuits as well. Next, the outputs of these XOR calculations were sent to the correction unit, along with bit M_{ij} (extracted from row *i* by the 4:1 mux). Finally, the correction unit determined that M_{ij} was corrupted, and so it inverted it and sent it to the output. There is more parallelism present here than during a write operation, and that is why the performance is

superior.

In the best case, a read from memory had a propagation delay of only 790ps. This happened when no error was detected in M_{ij}, the bit being read. The critical path in this case was simply the read of row *i* and column *j* from memory, the extraction of M_{ij} from row *i*, and the transmission of M_{ij} through one pass transistor in the correction unit. The XOR circuits detected no errors, so the parity inputs of the correction unit remained at zero during the whole read cycle. Because of this, the output was valid before the XOR calculations completed, which effectively eliminated their delay in this instance.

Even with the modest .5μm CMOS technology this circuit is exhibits high performance because of its very simple datapath. The number of serial functions has been reduced to the minimum for all necessary operations.

The second simulation shows writes and reads to every cell in main memory, and it is shown in figure 4.10. It displays the following inputs: The clocks (Clk, _Clk_),the four address bits (A3 & A2 for the row, and A1 & A0 for the column), read and write signals (Read, Write) and the data bit to be written (Din). The outputs are the row and column parity bits generated by the XOR networks (XORRow and XORCol), and the data bit that is read (DOut). The column of the cell being analyzed increments every clock cycle, while the row increments every four cycles. Writes to these cells were performed first. Data values of logic one are written to cells (0,0), (0,1), followed by zero to (0,2), (0,3) one to (1,0), (1,1), zero to (1,2), (1,3), one to (2,0), (2,1) zero to (2,2), (2,3), one to (3,0), (3,1), and zero to (3,2), (3,3). The simulation shows that consistent parity values are generated during memory writes, and the correct bit is transmitted during memory reads. Also, the parity values are low during memory reads, which shows that the system does not deem any of the stored bits to be incorrect, which is as expected.

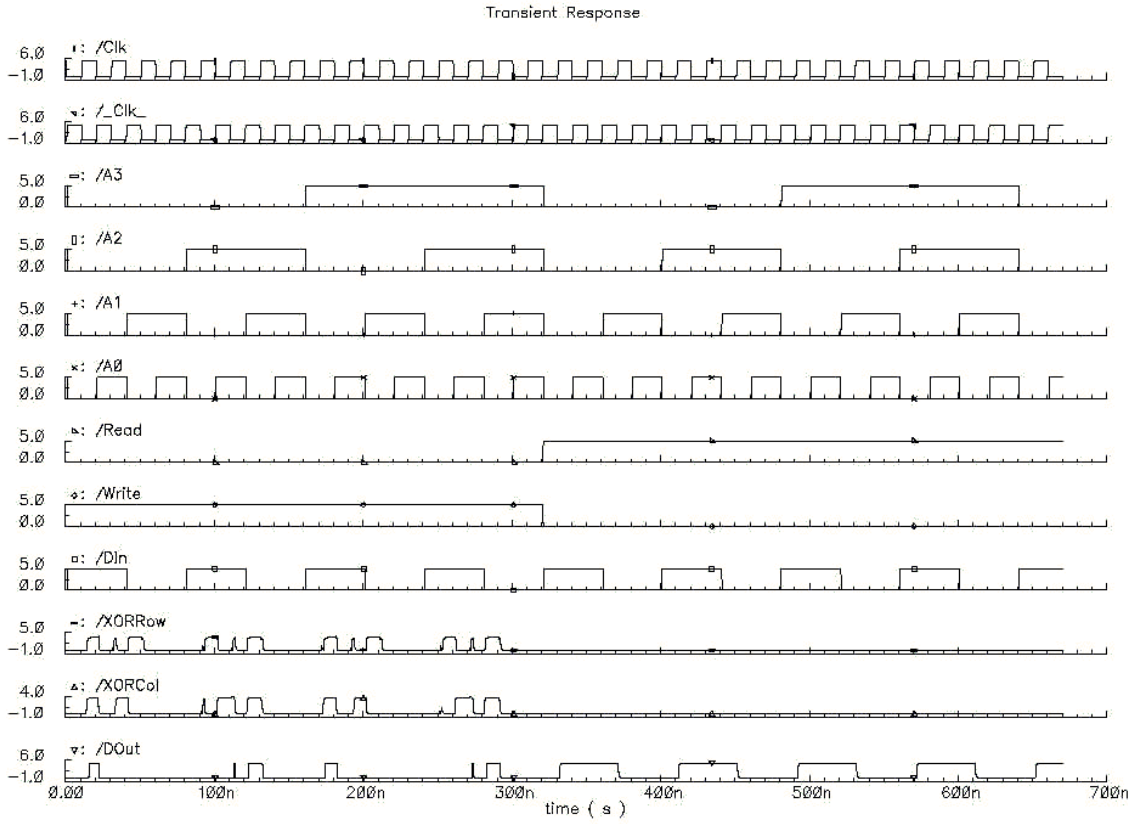

Figure 4.10: Cross-Parity System Simulation Showing Reads and Writes to Every Cell in Memory

Simulation three is displayed in figure 4.11, and it depicts the correction of an error in the system. First, a data value is written to cell (0,0). During the next clock cycle, an error changes the value stored in the cell. After this, the memory cell is read, and the error is fixed. The inputs shown on the simulation are the clock (Clk), address bits (A3, A2, A1, A0), read and write signals (Read, Write), data bit to be written (DIn), and error control bits (Error00 changes the value in cell (0,0) to Ebit00). The outputs are the row and column parity outputs (XORRow and XORCol), and the data output (DOut). During the first cycle, a logic zero is written to cell (0,0). In cycle two, the value in the cell is changed to a one by an error. A logic zero is read from the cell in cycle three, which shows that the bit was corrected. During this cycle, the row and column parity values are both at logic one, which means that the bit in memory was found to be in error. This causes the Correction Unit to invert the bit, which is then sent back to memory. And finally,

the cell is read again in cycle four. This time, the row and column bits are zeros, which shows that the bit in memory is accurate, which means that the feedback path functioned correctly. The entire algorithm is repeated in the second half of the simulation, except a one is written to the cell. Again, the system functions as desired.

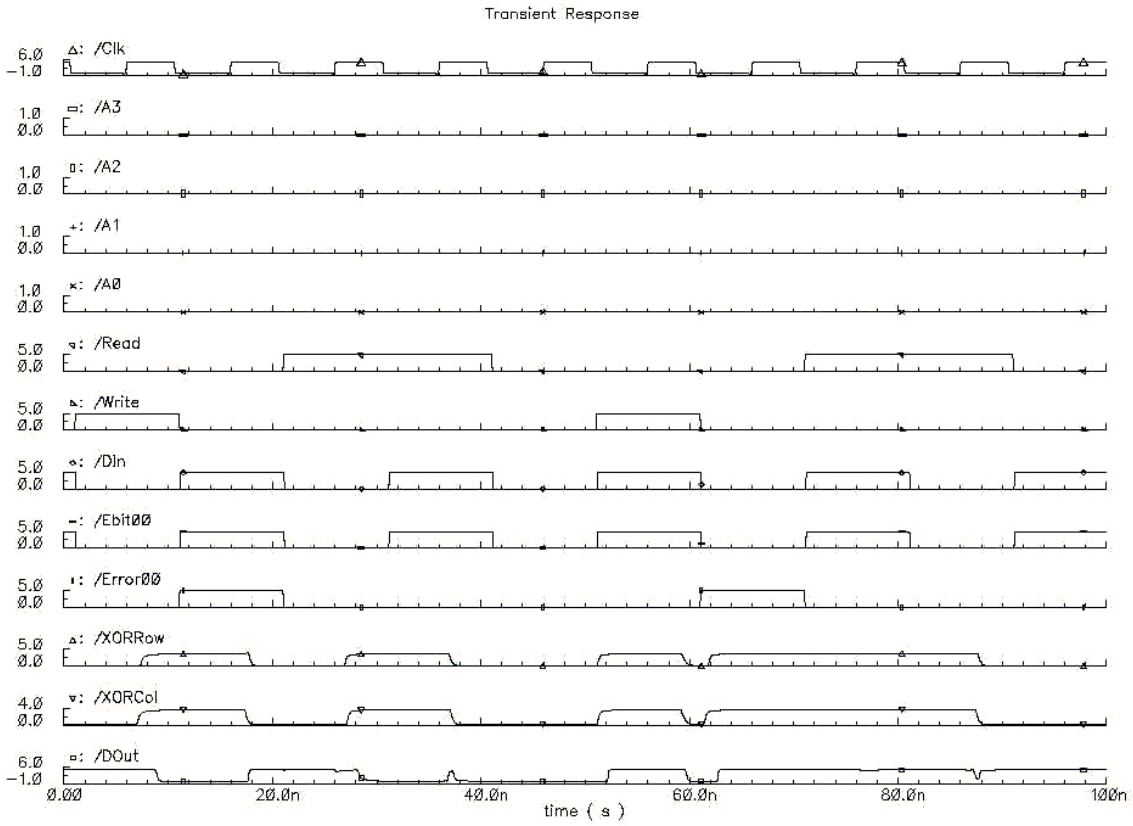

Figure 4.11: Cross-Parity System Simulation Depicting the Correction of an Error

The fourth simulation is similar to the second simulation, except two memory/error correction systems are examined in parallel. This was done to test the functionality of a full lookup table inside of an element in the reconfigurable DSP processor. Both systems are given their own input data bit patterns, causing them produce distinct outputs. The inputs in this simulation are the clock (Clock), address bits (A3, A2, A1, A0), and data inputs (Din1, Din2). The outputs come from the XOR networks (XorR1, XorC1, XorR2, XorC2), and from the data

outputs (DOut1, DOut2). As in the first simulation, data is written to every cell initially, and then each cell is read to show correct operation of the systems. Logic zero is written to every cell in column zero of the first system, while one is written to column one, zero to column two and one to column three. In system two, logic one is written to row zero, zero is written to row one, one to row two, and zero to row three. It is evident in the simulation that the data outputs in the read section (160ns to 320ns) match the corresponding data inputs from the write section (0ns to 160ns). According to the simulator, the entire layout functions correctly.

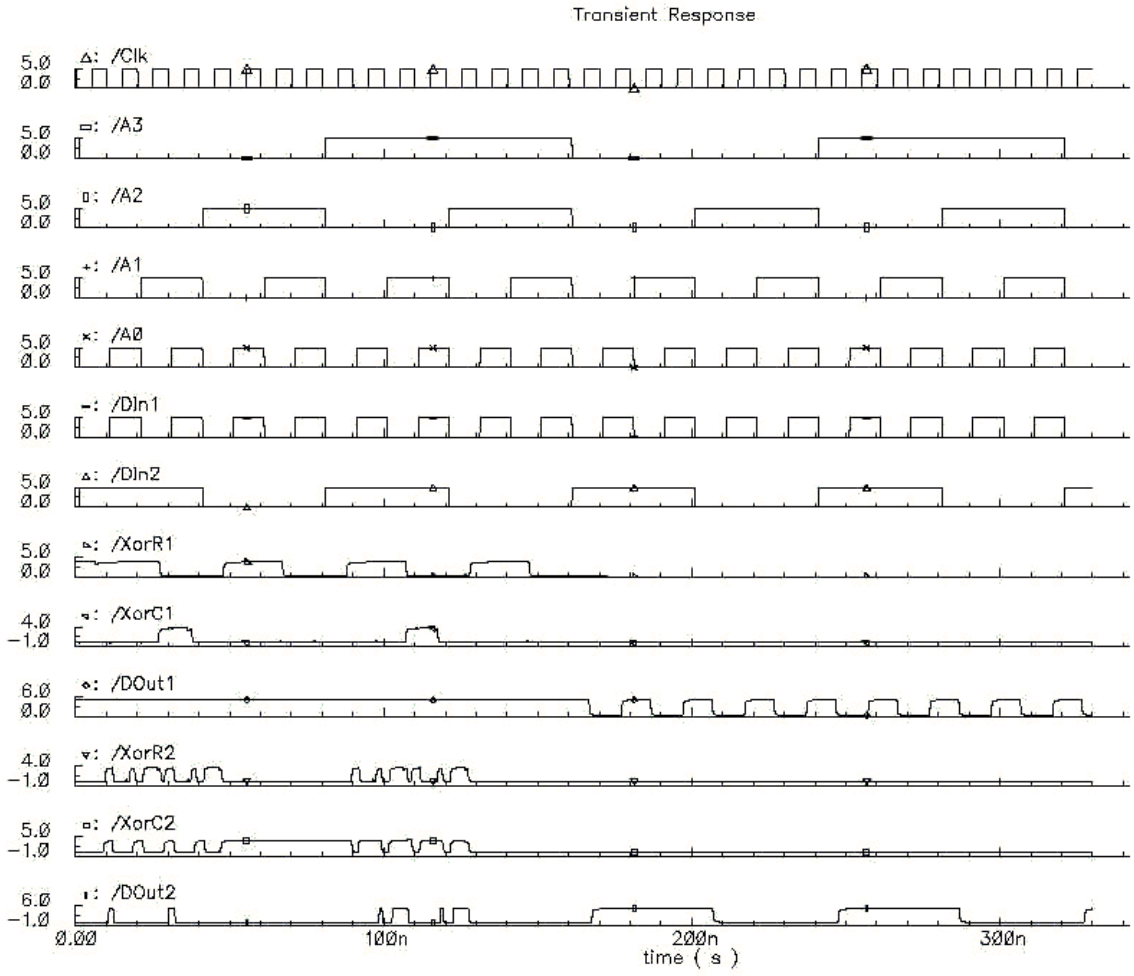

Figure 4.12: Cross-Parity System Simulation Showing Reads and Writes to Two Parallel Blocks

4.9 Chip Testing Approach

The design for this system has been implemented and simulated with Cadence layout tools, and fabricated by MOSIS. After the fabricated chip was received, it was tested to verify proper functionality. Computer simulations of the system layout are not sufficient, as the simulator can not detect problems such as latch-up or excessive power dissipation. Therefore, physical tests of the chip itself need to be performed.

The chip was tested by using a "Brain Box" to control the input pins while the outputs were observed with a logic analyzer. A simple test pattern was sent to the chip, and the output of the logic analyzer is included as figures 4.13 and 4.14. In the cases presented, the chip functions as expected.

Figure 4.13 is an output from the logic analyzer that demonstrates correct memory functionality. An alternating bit sequence is written into the memory during the first half of the simulation. This sequence is accurately read back during the second half of the simulation.

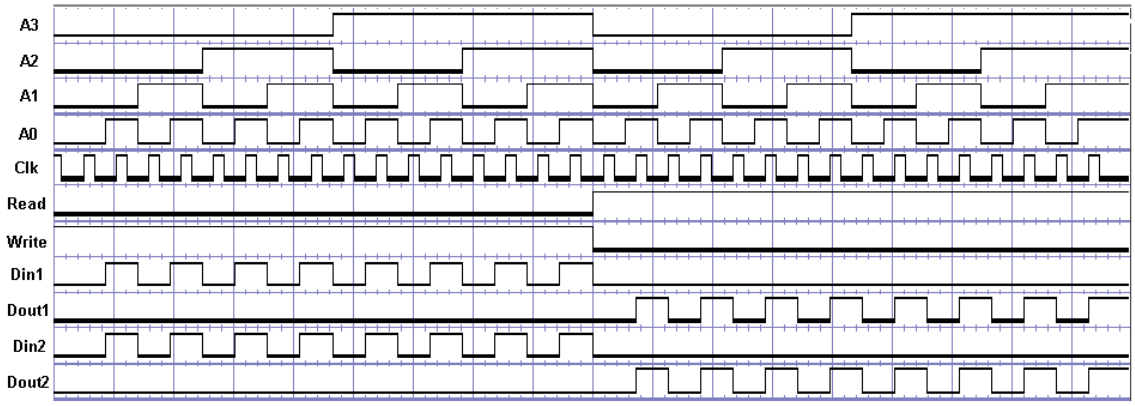

Figure 4.13: Logic Analyzer Output Showing Reads and Writes to All Memory Locations

Error correction is performed in the logic analyzer output displayed in figure 4.14. A value of logic 0 is initially written into cell (0,0). After this, the cell is erroneously forced to a logic 1. A read operation is then performed on the cell, which fixes the error. The correct value

of logic 0 is eventually present on the output. It can be seen that both the row and column parity bits are at logic 1 while the erroneous value is stored in the cell.

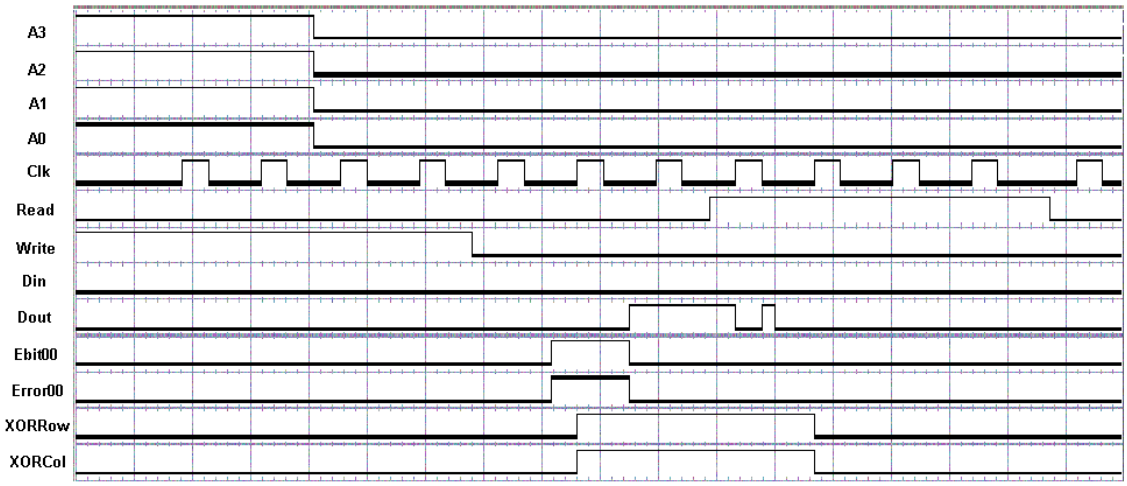

Figure 4.14: Logic Analyzer Output Depicting the Correction of an Erroneous Bit

Chapter 5

Modified DICE Scheme

The cross-parity method of fault-tolerance described above is a system-level scheme that is capable of correcting one bit in a memory block per read operation. This scheme can handle any type of fault affecting a memory cell, whether the damage is permanent or temporary. This robust capability comes at the cost of circuit area and delay, however. In applications where only one type of fault may occur, it would be beneficial to use a more elegant approach to meet the demands of the situation. For example, in radioactive environments, transient errors dominate. Small particles may impact a node in an IC, causing a transient error to occur at that location. It is possible to recover from these errors quickly using a compact circuit-level approach, as opposed to the more cumbersome system-level approaches described previously. System-level approaches such as TMR and cross-parity utilize circuitry outside of the memory to correct errors. In contrast, circuit-level approaches implement error correction inside of each memory unit. A typical circuit-level approach achieves fault-tolerance by incorporating redundancy and feedback into each memory latch [8]. This results in area and delay specifications that are considerably lower than those of a system-level approach, and even comparable to the specifications of unprotected memory.

5.1 Single Event Upsets

High concentrations of charged particles are often present in radioactive environments. When a charged particle strikes an IC, a Single Event Upset (SEU) may occur. Unwanted electrical

signals can be initiated in transistors when SEUs are triggered in unprotected circuitry. The effect of these unwanted signals increases as semiconductor feature size decreases. Reducing the feature size of an IC results in smaller node capacitances, which are more susceptible to injected charge [8].

SEUs occur most often as a result of alpha particle strikes [12]. Alpha particles are heavy ions that are created in space and other radioactive environments. When an alpha particle passes through the substrate of an IC, it may create an ionized trail that initiates a burst of charge [13]. If this charge collects at the drain or source of a transistor, it could initiate a current through the channel of that transistor. This potential current is highly undesirable, as it could cause a number of unwanted effects in a circuit. Most notably, it might change the state of a memory latch, which could affect the operation of the IC for a long period of time. Memory is a core element in many reconfigurable architectures, as it forms LUTs and controls interconnect configuration. The corruption of memory inside of such a circuit could cause an entire system to fail.

5.2 Efficiency of Circuit-Level Approach

A significant amount of research on the development of radiation-hardened ICs focuses on preventing SEUs, as these transient errors are perhaps the most significant electrical problem caused by radioactive particles. SEUs can easily cause unprotected circuits to fail, especially memory elements that utilize feedback and charge storage. As it turns out, there is a high probability that only one SEU will affect an IC at a time [8]. Because of this, a circuit that has the capability to recover from a single transient fault at any of its nodes would have a very high resistance to SEUs. Ideally, such a circuit would recover from SEUs quickly, and have low complexity. These goals could be achieved if the fault-tolerant circuitry was incorporated directly into every memory cell. This concept is referred to as a "circuit-level" approach [8].

As an example, redundant memory could be incorporated into each memory cell. If an

SEU affected the original memory bit, then the corresponding redundant memory bit could restore the original bit to the desired value. This process would occur quickly, as the redundant memory would be directly corrected to the original memory. Also, the complexity would be relatively low, as only one redundant memory bit is needed per original bit of memory. All things considered, it is possible to efficiently and effectively protect static RAM from SEUs with a circuit-level approach.

5.3 Basic DICE Cell

The Dual Interlocked storage Cell (DICE) memory cell described in [8] is possibly the most robust circuit-level approach available that has the capability to protect the internal nodes of a latch from SEUs. It requires only twice the circuitry of a standard SRAM latch, and it recovers from transient faults quickly. Also, this cell can be implemented in any CMOS process, and it does not require precise transistor ratioing. Figure 5.1 depicts the basic DICE cell.

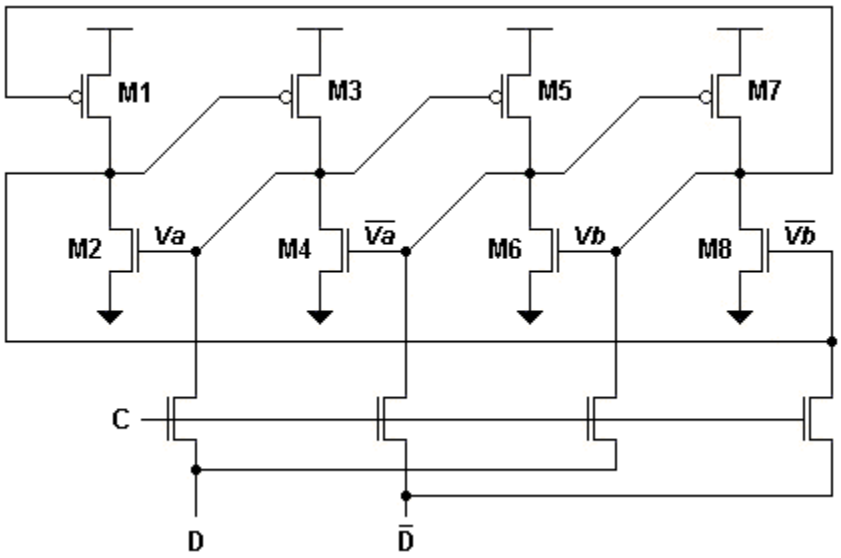

Figure 5.1: Basic DICE Cell

The DICE cell is based off of four inverters which are connected in an unorthodox fashion. There are four internal nodes in this latch, each of which are controlled by the output of one inverter. Nodes *Va* and *Vb* hold the logic value stored in the latch, while nodes $\overline{Va}$ and $\overline{Vb}$ hold the inverse of this value. A logic 0 is stored in the latch when *Va,* $\overline{Va}$*,Vb, and* $\overline{Vb}$ are at logic 0101, and a logic 1 is stored when the nodes are at logic 1010. The gates of the NMOS and PMOS transistors of each inverter are connected to separate nodes. Arranging the latch interconnectivity in this fashion assures that two separate inverters will restore an altered node to its original state via feedback.

Figure 5.2 is a simulation that shows the effect of an SEU on node *Va* of the DICE cell. This simulation was performed in 0.25µm technology with VDD = 2.5V. Initially, the internal nodes are at logic 0101, which implies that a logic 0 is stored in the latch. 0.25ns into the simulation, an SEU strikes *Va* and causes it to rise from GND to slightly above VDD. The SEU is modeled as a current pulse of 50mA amplitude with 0.05ns rise time and 0.2ns fall time. The SEU turns on M2, which affects the voltage at $\overline{Vb}$. $\overline{Va}$ and *Vb* remain unaffected, which allows them to restore the state of the latch after the SEU dissipates at 0.75ns. Effectively, the DICE cell recovered from a 0.25ns SEU within 0.5ns. SEUs that occur at $\overline{Va}$, *Vb*, or $\overline{Vb}$ have similar results.

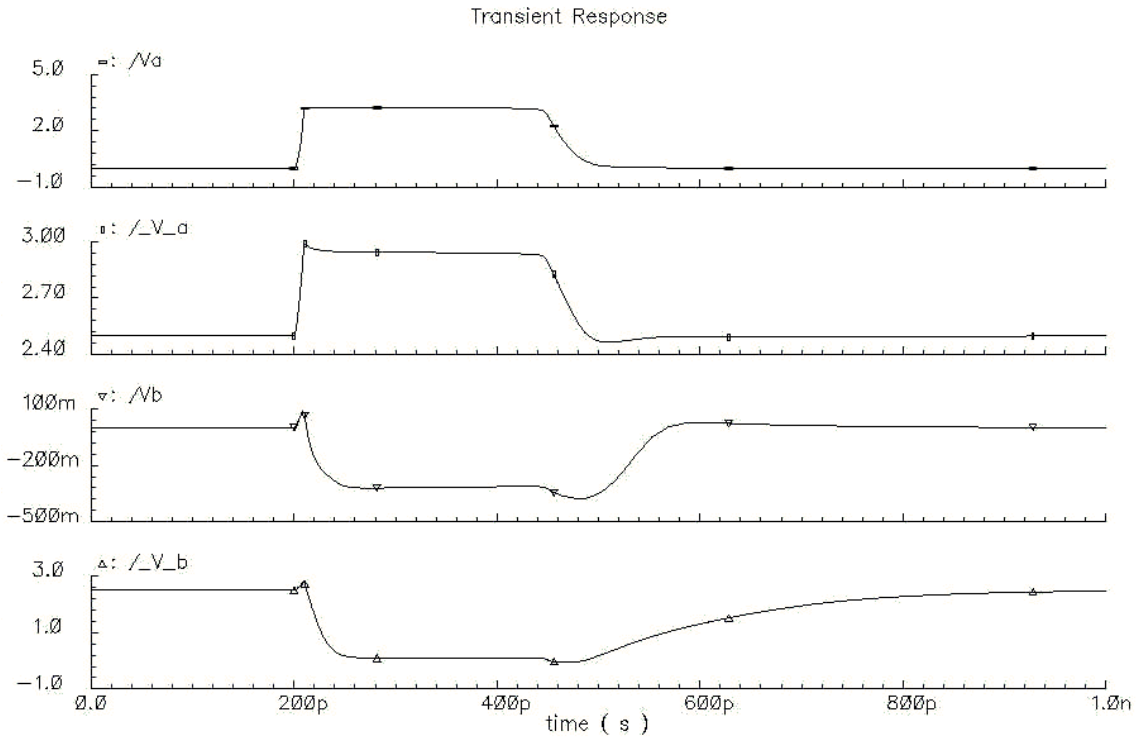

Figure 5.2: Basic DICE Cell Recovering from an SEU at an Internal Node

5.4 Problems With the Basic DICE Cell

It was demonstrated in the previous section that the basic DICE cell can recover from an SEU at any of its four internal nodes. However, the basic DICE cell has no protection against SEUs that may impact its data lines or the gates of its write transistors. It is possible for an SEU to initiate a transient current in the active region of the combinational logic that drives the data lines or the gates of the write transistors. This indirect effect is known as a Single-Event Transient (SET) [14]. The designers of the original DICE cell implied that SEUs do not carry enough charge to change the state of a latch from these locations. However, this is no longer the case with modern process technologies, as the masking of faults in combinational logic decreases with shrinking feature sizes [15]. Nodes with smaller capacitance are more sensitive to injected charge, which means that a SET can change their voltage levels to a greater degree. Therefore, the effect of SETs must be considered if a high level of fault-tolerance is desired.

The failure of the basic DICE cell to tolerate a SET affecting its write enable (*C*) input is illustrated in figure 5.3. The internal latch nodes are initially at logic 0101, and the *C* signal is at logic 0. The *D* and $\overline{D}$ data lines are at logic 1 and logic 0, which would be the case if a logic 1 was being read onto these shared data lines by another memory cell. 0.25ns into the simulation, an SET charges the *C* input above VDD. This turns on all four write transistors, allowing current to pass from the data lines into the latch. At 0.5ns, the latch state has flipped, and the internal nodes are at 1010. This is obviously not a desireable effect, as an SET has changed the value stored in the latch without the consent of the system. A similar effect may occur if an SET affects the *D* or $\overline{D}$ line at the end of a write operation.

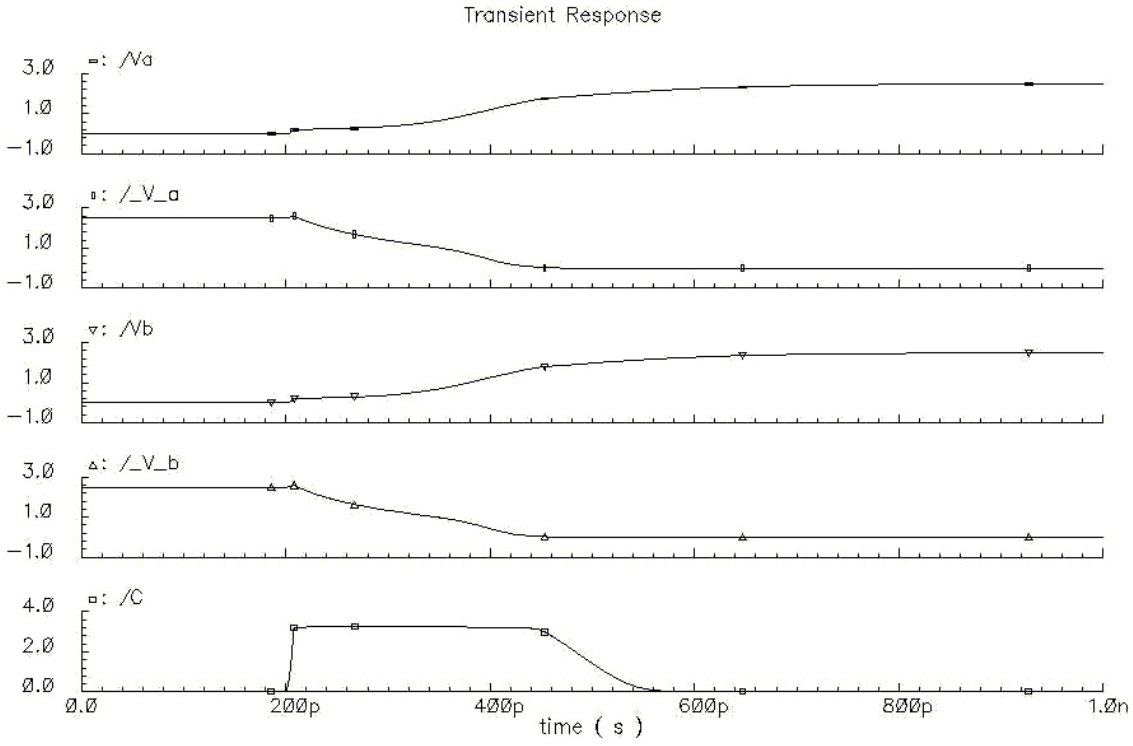

Figure 5.3: Failure of the Basic DICE Cell Caused by a SET Affecting Node C

5.5 Enhanced DICE Cell

This section presents and analyzes a modified version of the DICE cell that was developed to protect the memory from SETs affecting its write enable and data inputs. Figure 5.4 depicts the enhanced circuit. The four data lines have been separated into two read lines, $\overline{Qa}$ and $\overline{Qb}$, and two write lines, *Da* and *Db*. These data lines are connected to the latch via four transmission gates. Each transmission gate requires two control signals. *Ra*, $\overline{Ra}$, *Rb*, and $\overline{Rb}$ are used for the read lines, and *Wa*, $\overline{Wa}$, *Wb*, and $\overline{Wa}$ control the write lines. This setup incorporates redundancy to insure that an SET affecting any of the write enable or data inputs will not change the state of the latch.

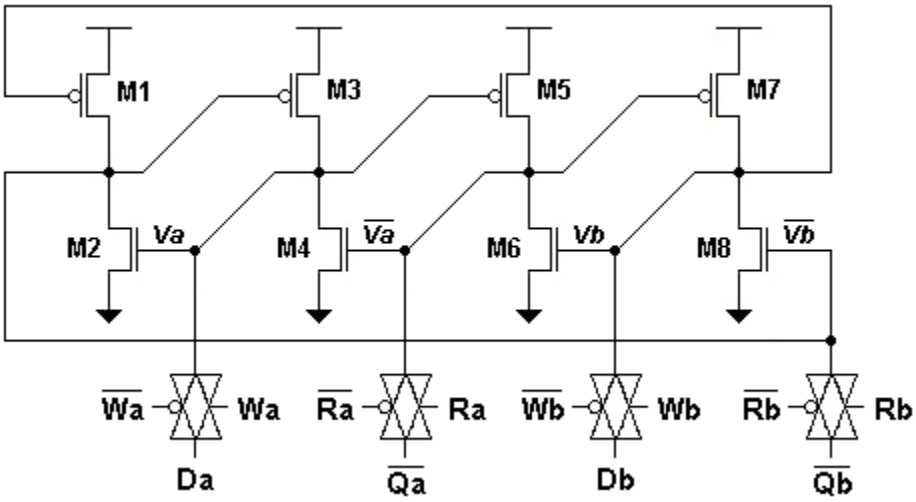

Figure 5.4: Enhanced DICE Cell

Each control signal is dual redundant, consisting of one signal with an *a* suffix and another with a *b* suffix. The *a* and *b* signals are generated independantly to insure that an error affecting one path does not affect the other. Because of this, a SET on *Wa*, *Wb*, *Da* or *Db* can only affect one internal node of the latch. This is acceptable, as the DICE cell was designed to

tolerate a single-node upset.

Figure 5.5 is a simulation of the enhanced DICE cell. The initial state of the latch is logic 0101, and *Da* and *Db* are at logic 1. A SET strikes *Wa* at 0.2ns, which causes *Wa* to go high and $\overline{Wa}$ to go low. This connects Da to Va, which increases the voltage at this node. Va controls the gates of M2 and M5, so increasing this voltage turns M2 on and begins to turn off M5. $\overline{Vb}$ is pulled down slightly because of this, but not enough to flip the latch state. The value stored in the latch returns to logic 0101 at 0.7ns.

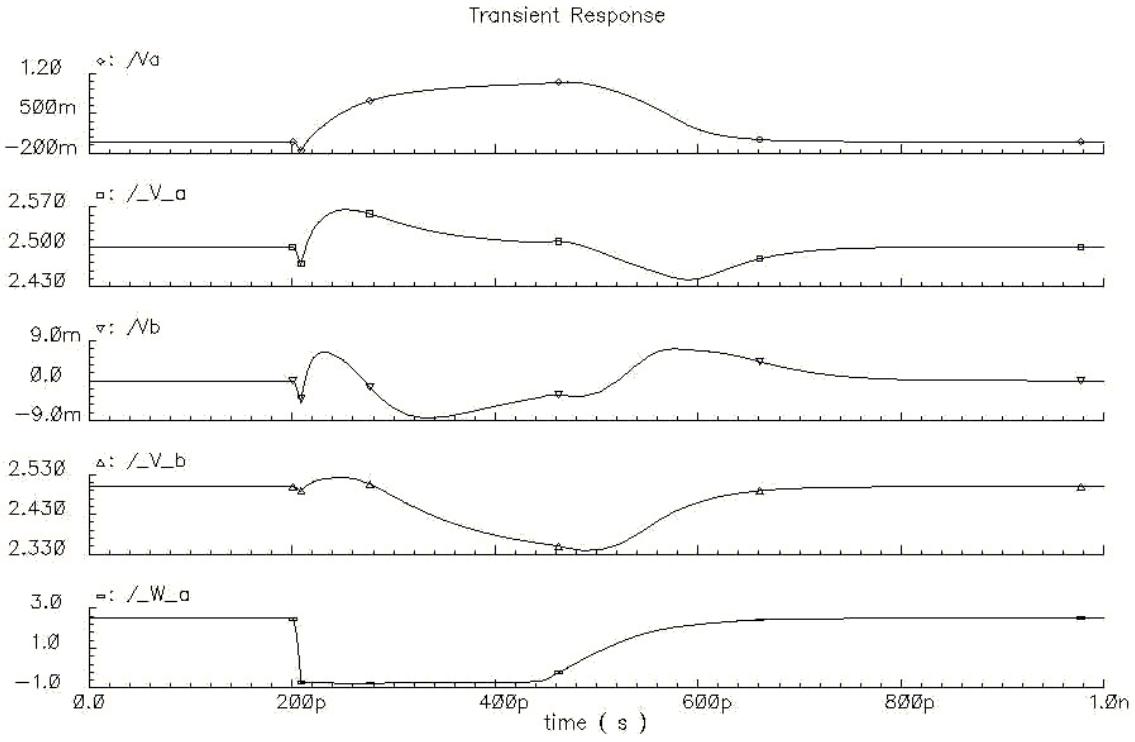

Figure 5.5: Enhanced DICE Cell Recovering from a SET on a Write Enable Input

5.6 Possible Failure of Enhanced DICE Cell During Read Operations

Although the enhanced memory cell is immune to isolated transient faults, normal read and write operations may create problems under certain conditions. The original DICE memory cell writes to all four internal nodes at once, whereas the enhanced memory cell only writes to nodes *Va* and *Vb*. Because of this, the four transistors driving *Va* and *Vb* (M3, M4, M7, and M8) must be made about one-half as strong as the other transistors. Otherwise, normal write operations will not be able to change the state of the cell. Changing transistor sizes to accommodate write operations impacts the ability to recover from transient faults at *Va* and *Vb*. If a glitch occurs at *Va* or *Vb* while no operation is being performed on the cell, the circuit will recover without any problems. However, if the glitch occurs at the beginning of a read operation, and the $\overline{Qa}$ and $\overline{Qb}$ lines are charged to the opposite logic levels as the internal nodes, the latch will change state if the bus capacitance is high enough.

Figure 5.6 depicts this SEU in the enhanced memory cell. The internal nodes initially have logic values 0101, but a transient spike pulls up *Vb* at 0.3ns. The $\overline{Qa}$ and $\overline{Qb}$ lines both store logic 0, and pull down $\overline{Va}$ and $\overline{Vb}$ after the read enable signals are asserted at 0.45ns. This combination of events causes the cell to flip state, as the internal nodes settle to logic values 1010 by the end of the simulation. Rectifying this problem would require additional transistors in the enhanced memory cell. Two solutions will be presented in the following sections that are optimized for particular applications. These optimizations substantially decrease or eliminate the likelihood of this type of SEU, while removing excess input signals.

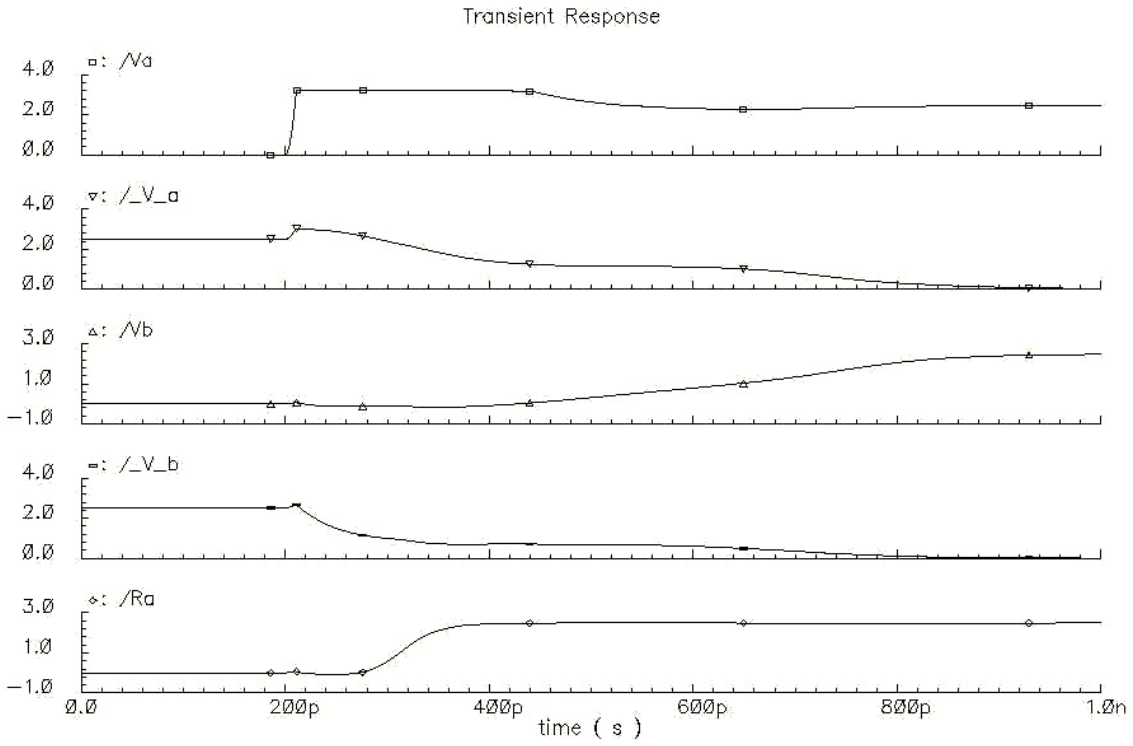

Figure 5.6: Failure of the Enhanced Cell to Tolerate an SEU at the Start of a Read Operation

5.7 Buffering Read Lines

The enhanced memory cell in figure 5.4 fails when stored charge in $\overline{Qa}$ and $\overline{Qb}$ affects nodes $\overline{Va}$ and $\overline{Vb}$ while a transient fault changes either *Va* or *Vb*. Simultaneously altering three nodes in this fashion changes the state of the latch. One way to correct this problem is to replace the pass transistors on the read lines with buffers, as shown in figure 5.7. This change prevents the data lines from affecting the internal nodes during read operations. Because of this, an SEU cannot combine with charge sharing from the data lines to flip the state of the latch. This buffered DICE cell is capable of robust performance when faced with SEUs and SETs that affect one node at a time. It is ideally suited for pipeline latches, since it continuously outputs the stored data on the read lines.

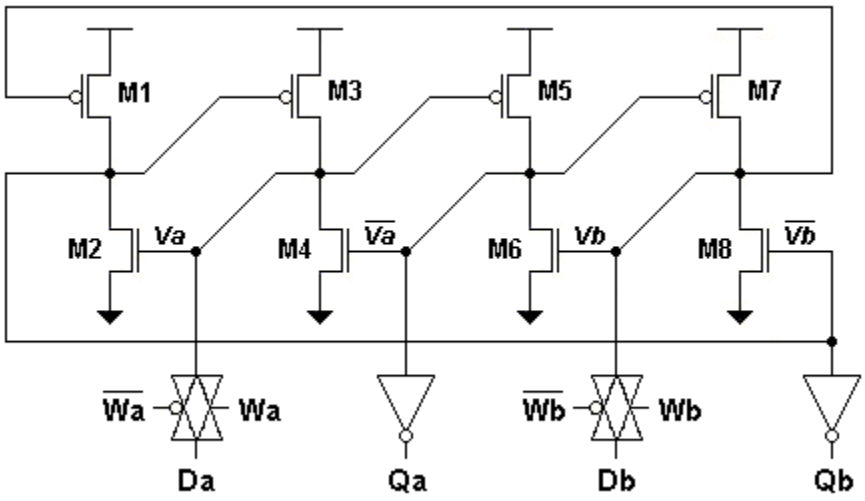

Figure 5.7: Buffered DICE Cell

Since only two data lines are utilized during write operations, transistor ratios must be set appropriately to insure correct functionality. The sizes of the inverters driving *Da* and *Db* and the write transmission gates must be larger than M3, M4, M7 and M8. In addition, all write enable signals must be active, and *Da* and *Db* must be equal for a write operation to occur. Figure 5.8 depicts a write to the buffered DICE cell. Initially, the latch state is logic 0101. The write enable signals are activated at about 0.1ns, and the latch settles to logic 1010 at about 0.7ns. This operation is not particularly fast, which is due to the fact that only two nodes receive the input data.

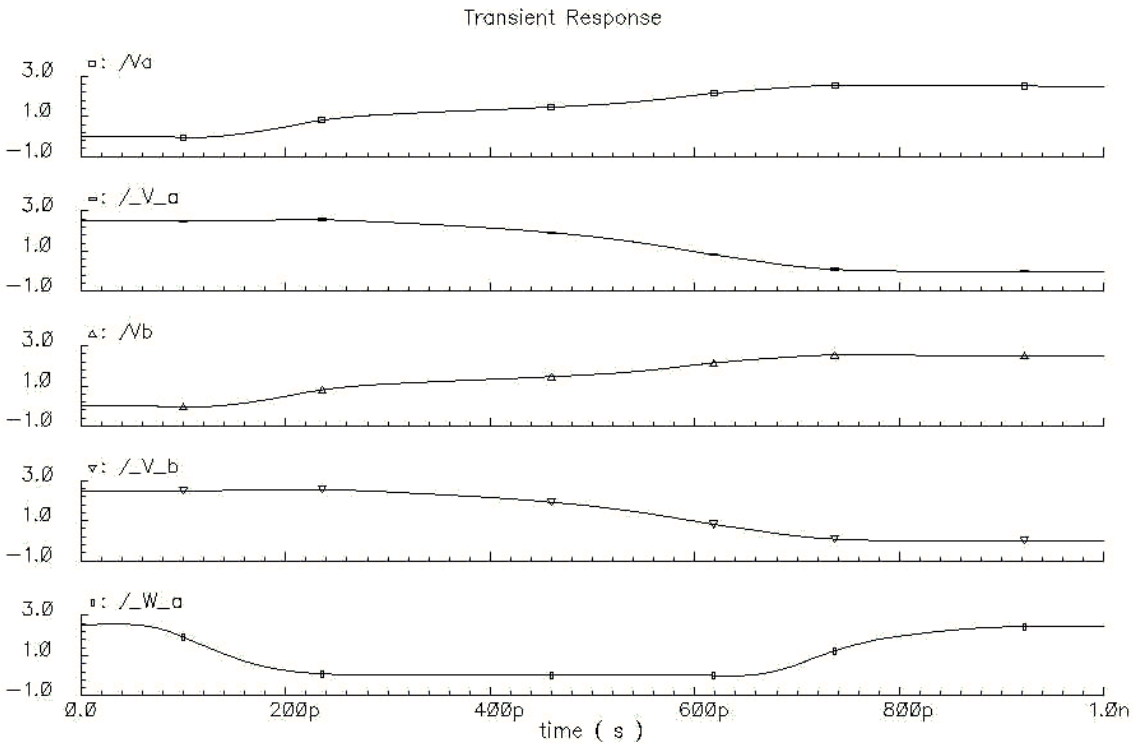

Figure 5.8: Simulation of a Write to the Buffered DICE Cell

Figure 5.9 depicts an SET that impacts *Wa* on the buffered DICE cell. The latch stores logic 0101, and *Da* and *Db* are logic 1. *Wa* and $\overline{Wa}$ are both activated around 0.2ns. The transmission gate connected to *Wa* and $\overline{Wa}$ is turned on, allowing *Da* to charge up *Va.* This turns on M2, which pulls down $\overline{Va}$ slightly. However, the state of the latch is preserved, and as it returns to logic 0101 by 0.7ns.

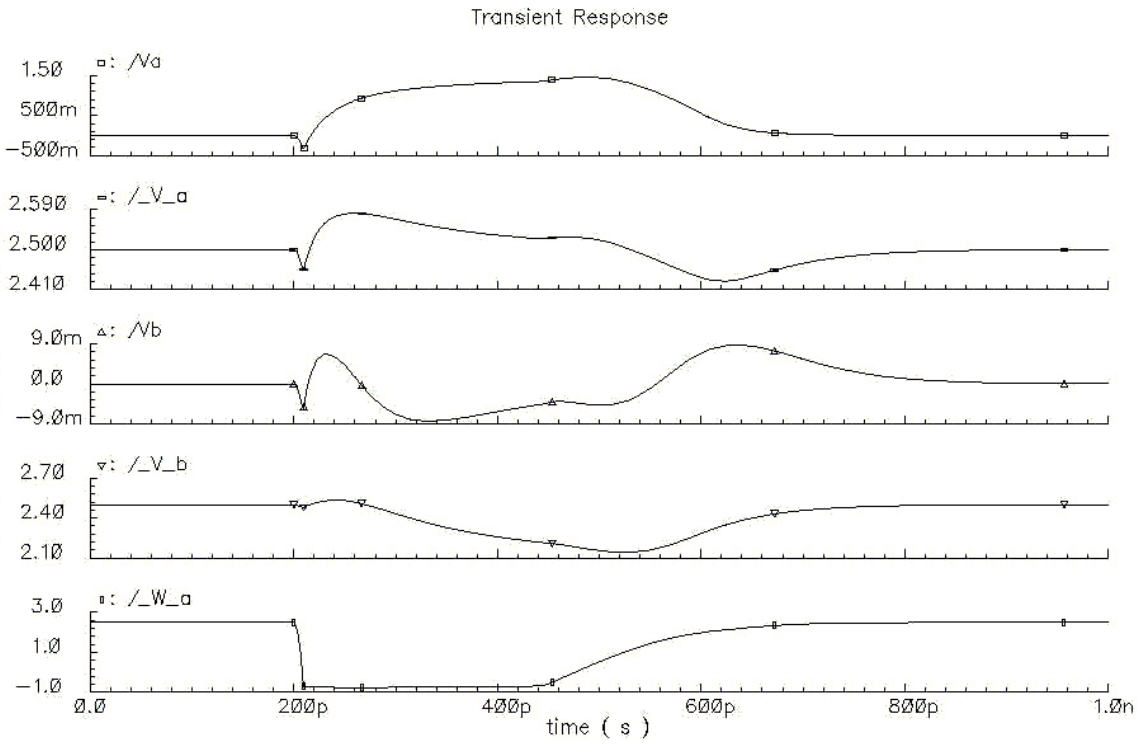

Figure 5.9: Simulation of a SET affecting node Wa in the Buffered DICE Cell

5.8 SEU-Resistant SRAM Cell

Although the buffered DICE cell presented previously has a high resistance to SEUs and SETs, it may not be the best choice for all situations. For example, memory blocks that utilize shared data buses benefit from memory cells that have read enable capability and differential inputs/outputs. Read enable capability allows multiple memory cells to share a data bus, while differential inputs/outputs facilitate faster data transfer. Figure 5.10 illustrates an SRAM cell that possesses these attributes. This cell uses four bidirectional data lines and enable transistors for read and write operations. M1-M8 must be weaker than M9-M20 to insure proper functionality.

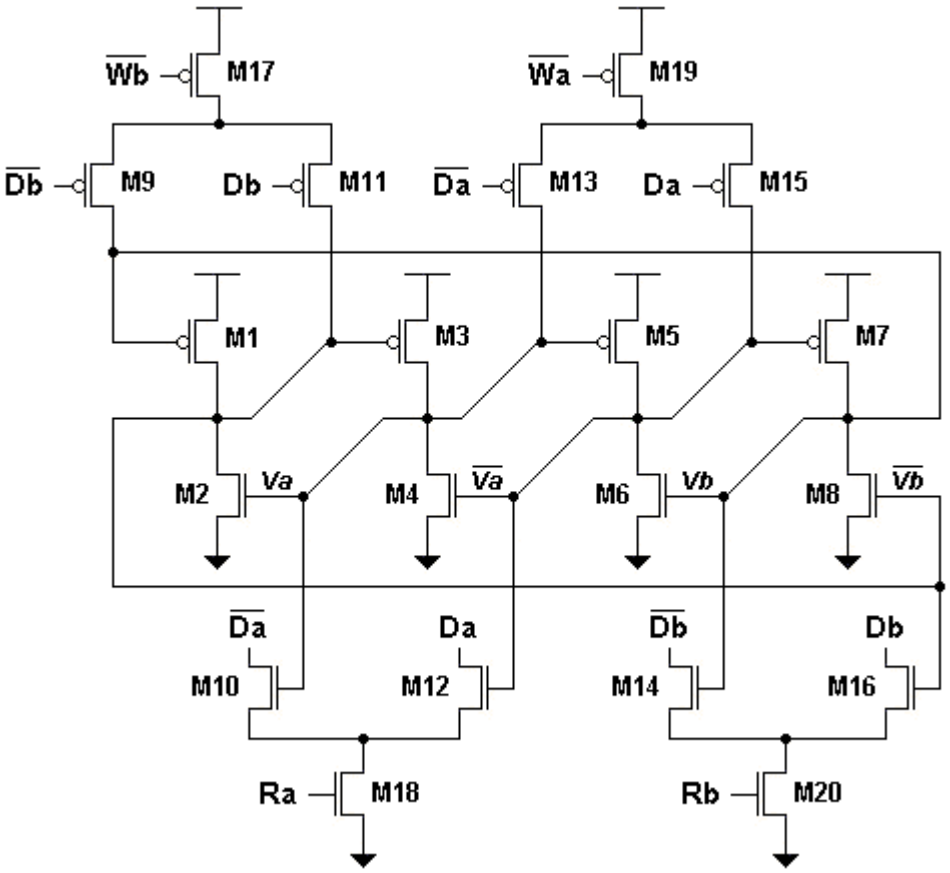

Figure 5.10: SET-Resistant SRAM Cell

The SRAM cell has a high resistance to SEUs and SETs that strike any single node in the circuit. The internal nodes are protected by the DICE configuration, and redundancy prevents errors that strike the data and write enable lines from having an effect. In addition, the internal nodes are completely isolated from the data lines, so parasitic charge sharing at the beginning of a read operation is not an issue.

Figure 5.11 depicts a basic read operation performed by the SRAM cell. *Da*, $\overline{Da}$, *Db*, and $\overline{Db}$ are precharged to logic 1, allowing the cell to discarge the two data lines that should be set to logic 0. M18 and M20 are turned on, which passes GND through these transistors. The latch state is at logic 0101, so $\overline{Va}$ turns on M12, and $\overline{Vb}$ turns on M16. In turn, this discharges

Da and *Db*. The read operation begins at 0.25ns, and *Da,* $\overline{Da}$, *Db*, and $\overline{Db}$ settle at logic 0101 by 0.6ns.

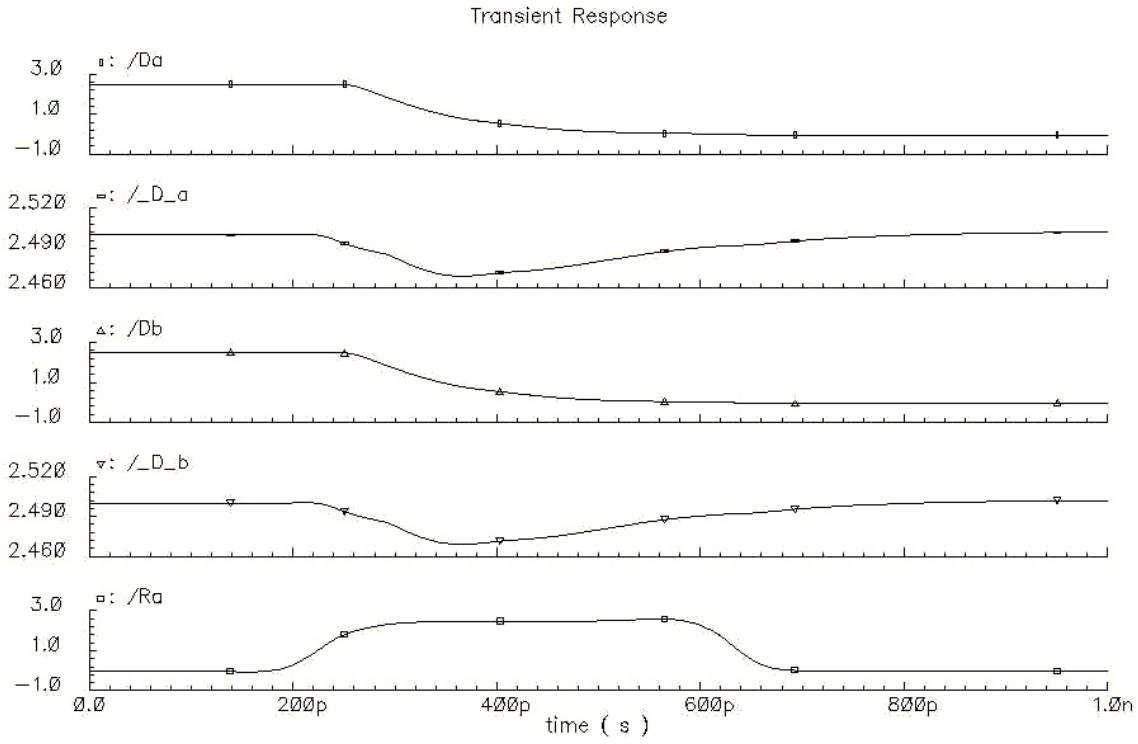

Figure 5.11: Basic Read Operation Performed by the SET-Resistant SRAM Cell

A basic write operation is shown in figure 5.12. The initial state of the latch is logic 0101. At about 0.15ns, the write operation is initiated, and M17 and M19 are turned on. VDD is passed through these transistors and to the sources of M9, M11, M13 and M15. Since *Da,* $\overline{Da}$, *Db,* and $\overline{Db}$ are at logic 1010, M9 and M13 are turned on. This passes VDD to *Va* and *Vb,* which causes the latch state to settle at 1010 by 1.0ns. This operation does not occur especially quickly, as only two nodes are directly controlled during writes to the memory.

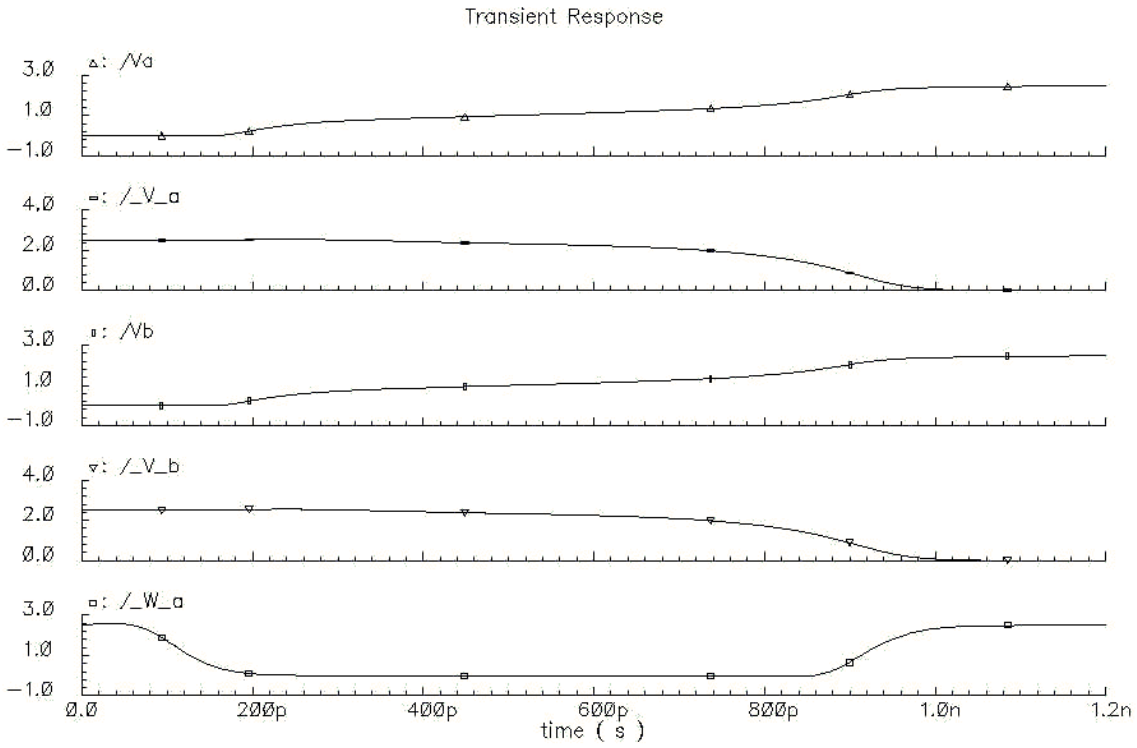

Figure 5.12: Basic Write Operation Performed by the SET-Resistant SRAM Cell

A simulation of a SET affecting *Wa* is shown in figure 5.13. The SRAM cell will not charge state if only one write path is activated. At 0.2ns, *Wa* is forced high and $\overline{Wa}$ is forced low. M19 is tuned on, passing VDD to the sources of M13 and M15. The initial state of the latch is logic 0101, and *Da,* $\overline{Da}$, *Db,* and $\overline{Db}$ are at logic 1010. M13 is turned on, which increases the voltage of *Va.* This turns on M2, which cuases $\overline{Vb}$ to go down slightly. The latch does not come close to flipping, and the state is restored by 0.7ns.

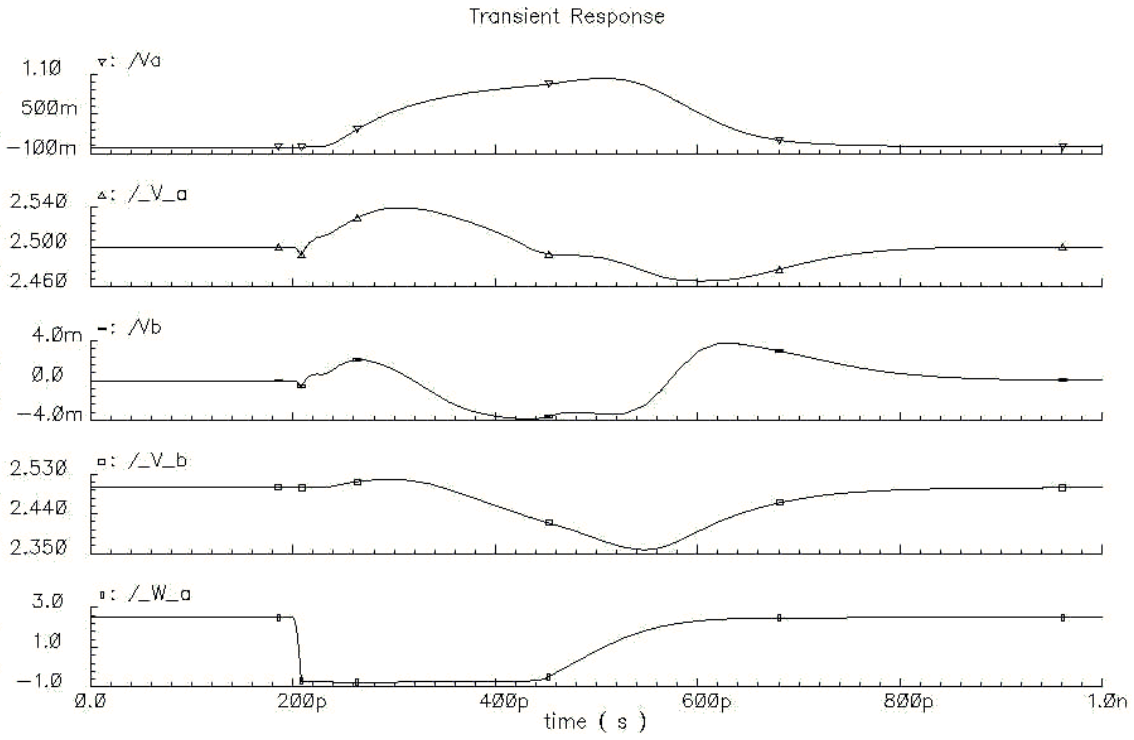

Figure 5.13: Effect of an SET on Node Wa of the SET-Resistant SRAM Cell

Figure 5.14 depicts a transient fault affecting *Da*. The initial latch state is logic 0101, and *Da,* $\overline{Da}$, *Db,* and $\overline{Db}$ start at logic 1010. *Da* is reduced below GND at 0.2ns, but $\overline{Da}$ remains at logic 0. This creates and undesired short between *Va* and $\overline{Va}$, as M13 and M15 are both turned on. *Va* is pulled up and $\overline{Va}$ is pulled down, although not by large enough margins to charge the state of the latch. The feedback in the DICE configuration supports *Va* and $\overline{Va}$, and M13 and M15 add ressistence in the path between *Va* and $\overline{Va}$. The latch state is restored to logic 0101 by 0.7ns.

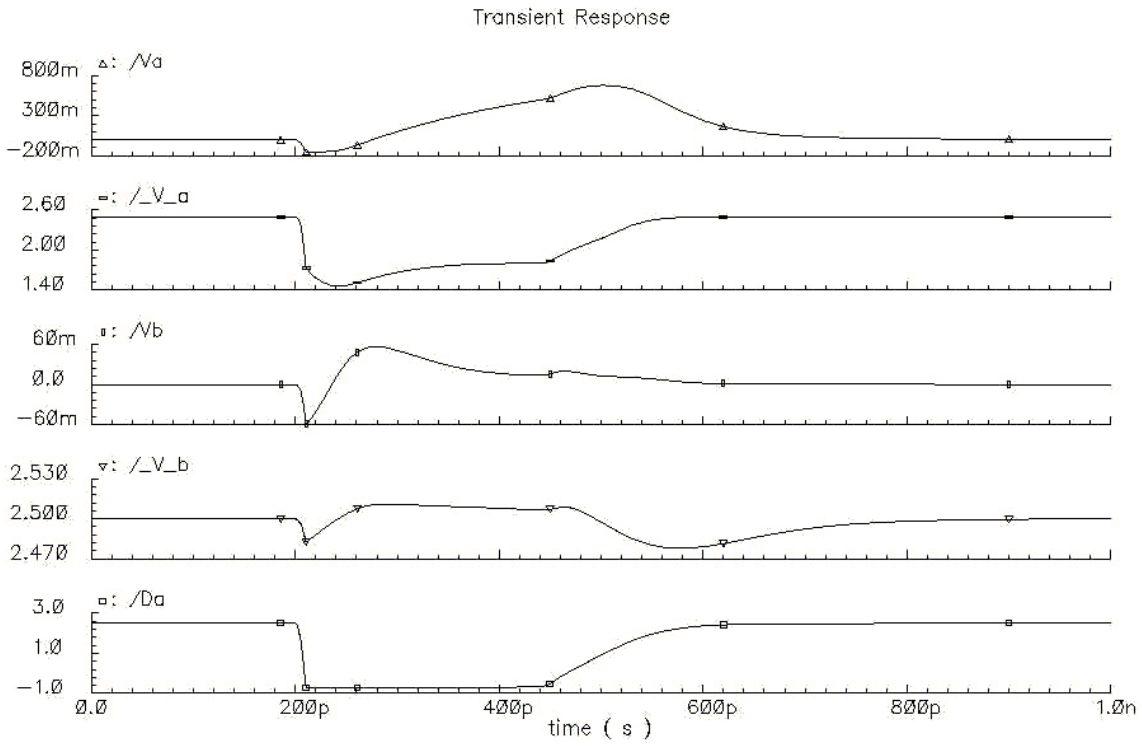

Figure 5.14: Effect of an SET on Node Da of the SET-Resistant SRAM Cell

The SRAM cell has the capability to tolerate one transient fault at any node per clock cycle. It has differential inputs and outputs, and it interfaces with a bidirectional data bus. These qualities make it a good choice for use as a cell inside of a memory block.

5.9 Comparison of Modified DICE with Cross-Parity

Figure 5.15 is a table that summarizes the attributes of the DICE designs described in this section. The original DICE cell required only one control line and an area of 82.1 μm^2, although it could only tolerate SEUs that affected its internal nodes. Protection was added to the control and data lines in the enhanced DICE cell, at the cost of eight control lines and 122.4μm^2 of area. However, charge sharing between the enhanced cell and the data lines could cause this design to fail during read operations. This problem was fixed in the buffered DICE cell, which required two control lines and 122.4μm^2 of area. Finally, the SRAM DICE cell required four control lines

and an area of 122.4μm^2. It can tolerate a single SEU at any of its nodes, and it has differential and bidirectional data lines.

DICE Cell	Control Lines	Data Lines	Area (um ^2)	Fault Tolerance
Original	1	2 rd/wr	82.1	Internal Nodes Only
Enhanced	8	2 rd, 2 wr	122.4	Problem w/ Reads
Buffered	2	2 rd, 2 wr	122.4	All Nodes
SRAM	4	4 rd/wr	147.4	All Nodes

Figure 5.15: Specifications of Each Variation of the DICE Cell

A diagram depicting the organization of a 32-bit modified SRAM DICE LUT layout is shown in figure 5.16. The memory is arranged into two 4x4 blocks to facilitate data operations that process two bits in parallel. In a 0.25μm process, each 4x4 block has been estimated to require an area of 2358.5μm^2. After including an 800μm^2 decoder, the total size of the LUT becomes 5517μm^2.

Figure 5.16: High-Level Organisation of Modified DICE LUT Layout

The organization of the cross-parity layout is illustrated in figure 5.17. It consists of the decoder and memory signal generator (Dec-and-MSG), two 4x4 main memory blocks, and the error correction circuitry. The area of each component is displayed in figure 5.18. As it turns out, the decoder and MSG occupied a very large portion of the total layout. This occurred because a large number of control signals were required for this system, which led to a significant

amount of logic and interconnect. All things considered, the 32-bit cross-parity LUT occupied 100,230μm^2 of area.

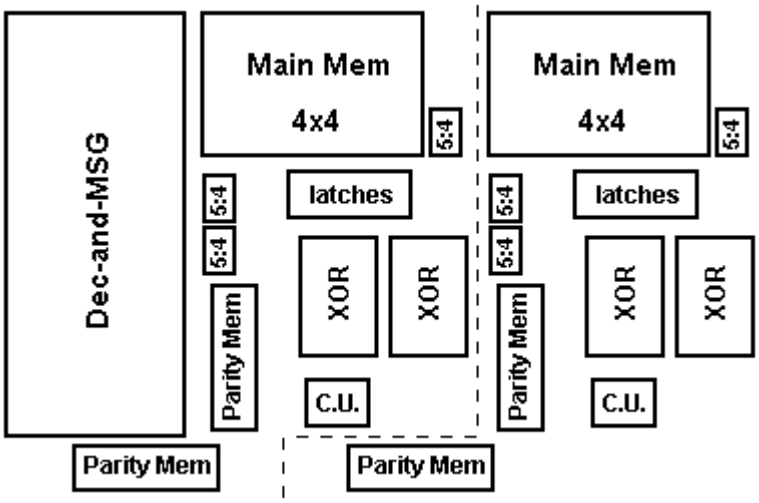

Figure 5.17: High-Level Organization of Cross-Parity LUT Layout

	Width (um)	Height (um)	Area (um^2)
Decoder and MSG (Dec-and-MSG)	134	243	32,562
Main Memory Block (Main Mem)	116	82	9512
Parity Memory Block (Parity Mem)	82	24	1968
XOR Network (XOR)	45	83	3735
Five-to-Four Switch (5:4)	21	33	693
Correction Unit (C.U.)	35	19	665
Entire Layout	390	257	100,230

Figure 5.18: Table of Cross-Parity Layout Dimensions

Figure 5.19 is a table that compares the area and delay specifications of the cross-parity and modified DICE approaches. The first column lists the area and delay of the cross parity scheme in a 0.5μm process, and column two shows the parameters of the SRAM DICE LUT in a

0.25μm process. Column three presents the estimated specifications of the DICE LUT in a 0.5μm process. The area of the DICE LUT is 22,067μm^2 in the 0.5μm process, which is substantially smaller than the 100,230μm^2 cross-parity LUT. Also, the DICE approach had delays of only 0.7ns for reads and 1.7ns for writes, compared to the cross-parity delays of 2.42ns and 4.13ns. In situations where only transient errors occur, the DICE LUT is superior, as it requires only 22.01% of the area, 28.81% of the read delay, and 41.16% of the write delay.

	Cross-Parity (0.5u, 2 metal)	SRAM DICE (0.25u, 3 metal)	SRAM DICE (0.5u, 2 metal)
Total Area	100,230 um^2	5517 um^2	22,067 um^2
Read Delay	2.42ns	0.35ns	0.7ns
Write Delay	4.13ns	0.85ns	1.7ns

Figure 5.19: Comparison of Area and Delay of Cross-Parity and Modified DICE Designs

Chapter 6

Conclusion

The importance of fault-tolerance in IC design is increasing as feature size decreases. Also, the use of reconfigurable systems is growing at a formidable pace due to their flexibility and low cost. In contrast, the price of custom ICs is inflating rapidly. Therefore, it is likely that novel research on fault-tolerant reconfigurable architectures will be used in future commercial IC constructions. It is important to customize each fault-tolerant scheme around the demands of the system and the types of errors that will affect it. Memory is the core element of many reconfigurable systems. It stores vital data and controls the interconnect configuration of the processor. Because of this, the fault-tolerant scheme of such a system should focus on protecting the memory. Fault-tolerance can be expanded into the rest of the system after the dependability of the memory has been assured.

6.1 Fault-Tolerant Schemes

A number of methods are available for implementing fault-tolerance in memory. Each method has benefits and drawbacks that suit it for particular applications. System-level approaches such as TMR, Hamming code, and cross-parity can recover from permanent faults, but they require significant circuitry outside of memory to achieve their functionality. Circuit-level approaches can only recover from transient faults, but they require substantially less area and delay. In a circuit-level scheme, all of the error correction circuitry is incorporated directly in the memory cells.

Two approaches were used to implement fault-tolerance in the memory inside of the reconfigurable DSP processor. The first approach was a cross-parity system-level scheme. It can correct up to one bit in each side of a 32-bit LUT per read operation. The second approach was a modified DICE scheme, which operates at the circuit level. This method can recover from up to one transient error in every memory cell per clock cycle.

A cross-parity scheme was the first approach selected to implement fault-tolerance in the memory for the reconfigurable DSP processor. The system consists of two mirror-image subsystems that work in parallel to deliver two data bits after receiving four address bits and various control signals. The subsystems each contain a four-by-four bit main memory block that stores data, and an error correction block that can correct the data bit if necessary. Errors are corrected in all cases with up to one error per subsystem. These errors are discovered through the use of a cross parity scheme, which stores parity bits during memory writes, and then compares them to the parity bits generated during memory reads. Errors are discovered and corrected when the parity bits do not match.

The memory block is made up of an address decoder and a signal generator, which generate the control signals, and a main memory unit that stores the data. The error correction block consists of XOR networks that calculate parity bits, parity memory that stores these bits, a Correction Unit that corrects erroneous data, and muxes that route internal signals. The performance of the system has been observed through the use of Cadence simulation tools, and has been analyzed in this report.

The best and worst-case delays have been documented, establishing the performance bounds of the system. Even with the modest .5μm technology, encouraging results have been observed. The largest delay in the circuit was found to be 4.13ns, while the smallest delay was 790ps.

The CMOS layout of this system has been implemented and fabricated. The two halves of the system are mirror images of each other, and so one half of the system was implemented,

and then copied to produce the whole system. This resulted in a redundant address decoder unit, which is acceptable for this test fabrication of the system. After the chip was fabricated by MOSIS, it was tested for functionality.

When considering only transient errors, the circuit-level DICE approach outperforms the cross-parity scheme in all areas. Transient errors are a substantial factor in radioactive environments, which means that the DICE approach is suited for such conditions. The DICE approach utilizes redundant memory and feedback to quickly recover from glitches that affect its internal nodes. All of the error correcting circuitry is included in the memory cells, which allows the DICE approach to be implemented in substantially less area than the cross-parity scheme.

A number of variations of the DICE cell were presented in this report. The original DICE cell described in [8] was shown to have a strong resistance to SEUs that impact its internal nodes. However, it offered no protection against SETs that affected its write enable inputs and data lines. An enhanced DICE cell was proposed to address this problem. It divided the four bidirectional data lines into two read lines and two write lines. Redundant data and enable signals were used to protect against SETs that strike any one of these lines. Unfortunately, it was discovered that data stored in this circuit could be compromised f an SEU altered an internal node during a read operation. A design utilizing buffered read lines was introduced to eliminate this problem. This circuit functions well under all circumstances, as it can recover from an SEU or SET that strikes any of its nodes. It is ideally suited for use as a pipeline latch, as it continuously outputs the stored value onto the read lines. In situations where a SRAM memory cell is required, a differential bidirectional data bus is desirable. For this reason, a SRAM buffered DICE cell was designed. All four of its data lines are used during read and write operations, and these lines are buffered from the internal nodes of the latch. Precharging is used to facilitate proper operation and improve the speed of operation.

The system-level cross-parity scheme and the circuit-level modified DICE approach both have strengths and weaknesses that make them suitable for specific applications. The cross-parity

scheme can correct up to two errors in a LUT per read operation. These errors can be of any type, which adds to the versatility. This scheme is ideal in situations where the error rate is low, and both permanent and temporary faults occur. On the other hand, the modified DICE approach can only recover from transient errors. However, it requires significantly less area and delay than the cross-parity scheme.

The total area of a 32-bit DICE memory block was calculated to be 5517μm^2 in a 0.25μm process, while the delay was 0.35ns for reads and 0.85ns for writes. On the other hand, the cross parity scheme required 100,230μm^2 of area in a 0.5μm process. It had 4.13ns read delays and 2.43ns write delays. To make the comparison fair, the DICE memory block has been projected to require 22,067μm^2 of area and delays of 0.7ns to 1.7ns in a 0.5μm process. When compared to the cross-parity scheme, the DICE cell requires only 22.01% of the area, 28.81% of the read delay, and 41.16% of the write delay. Because of this, it is clear that the modified DICE approach is the better choice for protection against transient errors.

6.2 Contributions

Many approaches have been designed to protect static memory from errors. System-level approaches such as TMR, Hamming code, and parity schemes have been designed to correct all types of errors. On the other hand, transient errors can be overcome with circuit-level schemes like the DICE approach and enhanced-impedance memory cells. However, very little effort has been invested into discovering fault-tolerant solutions for reconfigurable DSP architectures.

Reconfigurable processors are becoming more popular as the price of custom solutions increases, and DSP is a vital component in communications, entertainment, and other mixed-signal systems. In addition, fault-tolerance is becoming more critical as semiconductor feature sizes decrease. Therefore, it is likely that fault-tolerant reconfigurable DSP architectures will become foundations for many systems in the future.

In this report, two novel approaches were proposed to implement fault-tolerance inside of a reconfigurable DSP processor. The first design was a system-level cross-parity scheme that has the capability to correct any type of error in a 32-bit LUT. The second design was a modified DICE approach that can correct transient errors quickly and efficiently. Each design exhibits characteristics that make it suitable for specific applications. The cross-parity scheme fits well in systems that require a general-purpose solution that can handle relatively low error rates. In contrast, the modified DICE approach is at home in systems that only require protection against transient errors. Because of this, the DICE approach performs well in radioactive environments.

The original cross-parity implementation presented in this report was created to protect the 32-bit LUTs inside of a reconfigurable DSP processor. Reads and writes involving a LUT are two bits wide, so the LUTs were split into parallel 16-bit units to improve speed. This system can correct one error per read operation in each 16-bit unit. It only requires a 50% memory cell overhead, which is much lower than the 200% overhead required for TMR. Utilizing fewer memory cells leads to less area and power consumption. Also, the cross-parity scheme can be expected to experience half as many errors as TMR because it requires only half the memory. In addition, the scheme presented in this report is much less complex than other cross-parity systems (such as the design presented in [9]). All things considered, this cross-parity scheme is a robust and efficient system-level scheme. These attributes make it well suited for use in the small 32-bit LUTs of the reconfigurable DSP processor.

Single Event Upsets are transient errors that have been the subject of a significant amount of recent research. The DICE cell, which was first presented in [8], was specifically designed to tolerate these errors. This report focused on a number of modifications to the original DICE cell, with the intention of improving its resistance to SEUs. Initially, the buffered cell was introduced to demonstrate how the DICE cell could be protected against SETs on its write enable and data lines. This capability will become more important as IC feature sizes decrease, as SETs will have a greater affect on node voltages. After SETs were addressed, it was discovered that charge

sharing between the cell and the data lines could upset the latch at the beginning of a read operation. No other DICE design has considered this problem before. Buffering the internal nodes of the cell from the data lines was the solution presented in this report. This was implemented in the pipeline and SRAM latches, giving them the capability to tolerate a single error at any node in their circuitry.

The work contained in this report is perhaps the first implementation of fault-tolerance inside of a reconfigurable DSP processor. An efficient system-level cross-parity scheme has been designed to offer robust protection to 32-bit LUTs inside of the processor. In addition, a circuit-level modified DICE memory cell has been presented that can recover from a single transient error affecting any of its nodes. Both designs use a novel approach to address issues specific to fault-tolerant reconfigurable DSP.

6.3 Future Work

The modified DICE designs presented in this report are well suited for use as building blocks for a radiation tolerant DSP processor. In the future, the modified DICE memory cells will be integrated into a reconfigurable DSP architecture. The resistance to radiation of the resulting system will be measured. This resistance will be improved by incorporating additional fault tolerance outside of the memory cells. Interconnect and combinational logic can also be affected by SEUs, so it would be beneficial to protect them.

The redundant data and control lines of the modified DICE cell will facilitate the protection of the interconnect and logic. Two independent paths will be used to prevent a single SEU from upsetting data transferred to and from the memory cells. The cells will not change state during write operations if a data or control line is at the wrong voltage level. When combined with temporal redundancy, a fault affecting transmitted data will be ignored by the system. Temporal redundancy is implemented by making the clock cycle longer, which gives the

system time to wait for SEUs to dissipate and write the correct data to memory. If the maximum duration of an SEU is ΔS and the minimum clock cycle length is ΔC, SEUs on the data and control lines will not affect the memory if we increase the clock cycle length to $\Delta S + \Delta C$. This is true as long as only one SEU affects a memory cell per clock cycle.

An in-depth analysis of the system's ability to tolerate SEUs at any location will be performed. This will take into account the probability of a fault occurring at each node and the probability that other circumstances allow the fault to disrupt the state of the system. This study will yield a comprehensive model that will identify the probability that our system fails due to SEUs, dependent on feature size and clock frequency. The results from this proposed research will be applicable to a number of platforms in multiple situations.

This work will expand into the protection of an entire reconfigurable system. This scheme will utilize a redundant path for every data and control signal that is communicated between memory locations. The memory cells will be used to supply and synchronize these signals. All things considered, this future research will detail the decision making process involved in the incorporation of fault-tolerance into reconfigurable architectures.

References

[1] R. Tessier and W. Burleson, "Reconfigurable Computing for Digital Signal Processing: A Survey, Y. Hu (Ed.), *Programmable digital signal processors* (Marcel Dekker Inc., 2001).

[2] J. Smit et al, "Low Cost and Fast Turnaround: Reconfigurable Graph-Based Execution Units, *Proc. 7th BELSIGN Workshop*, Enschede, the Netherlands, 1998.

[3] M.J. Myjak and J.G. Delgado-Frias, "A Two-Level Reconfigurable Architecture for Digital Signal Processing," in *Proc. 2003 International Conference on VLSI*, Las Vegas, NV, pp. 21-27, Jun 2003.

[4] M.J. Myjak, "A Two-Level Reconfigurable Cell Array for Digital Signal Processing," M.S. Thesis, Washington State University, May 2004.

[5] E.L. Peterson et al, "Calculations of Cosmic Ray Induced Soft Upsets and Scaling in VLSI Devices," IEEE Trans. on Nuclear Science, vol. NS-29, no. 6, Dec 1982, pp. 2055-2063.

[6] A.M. Chugg, "Ionising Radiation Effects: A Vital Issue for Semiconductor Electronics," *Engineering Science and Education Journal*, vol. 3, pp. 123-130, Jun 1994.

[7] J.J. Wang et al, "SRAM Based Re-Programmable FPGA for Space Applications," IEEE Trans. on Nuclear Science, vol. 46, no. 6, Dec 1999, pp. 1728-1735.

[8] T. Calin, M. Nicolaidis, and R. Velazco, "Upset Hardened Memory Design for Submicron CMOS Technology," *IEEE Trans. on Nuclear Science*, Indian Wells, CA, vol. 43, pp. 2874-2878, Dec 1996.

[9] M. Pflanz et al., "On-Line Error Detection and Correction in Storage Elements with Cross-Parity Check," *Proc. Eighth IEEE International On-Line Testing Workshop*, pp. 69-73, 2002.

[10] L.R. Rockett, Jr., "An SEU-Hardened CMOS Data Latch Design," *IEEE Trans. on Nuclear Science*, Portland, OR, vol. 35, pp. 1682-1687, Dec. 1988.

[11] J. Rabaey, A. Chandrakasan, B. Nikolic, *Digital Integrated Circuits: A Design Perspective, 3rd Edition*, Pretence Hall, Inc., New Jersey, 2003.

[12] E.L. Petersen, J.C. Pickel, E.C. Smith, P.J. Rudek, and J.R. Letaw, "Geometrical Factors in SEE Rate Calculations", *IEEE Transactions on Nuclear Science 40*, pp. 1888-1909, 1993.

[13] Pickel, J. C., and J. T. Blandford, Jr., "Cosmic-Ray-Induced Errors in MOS Devices", *IEEE Trans. Nucl. Sci., NS-27*, 1006-1015, 1980.

[14] K.J. Hass and J.W. Ambles, "Single Event Transients in Deep Submicron CMOS," *Proc. 42nd Midwest Symp. On Circuits and Systems*, Las Cruces, NM, vol. 1, pp. 122-125, Aug 1999.

[15] P. Shivakumar et al, "Modeling the Effect of Technology Trends on Soft Error Rate of Combinational Logic," *2002 International Conference on Dependable Systems and Networks (DSN 2002)*, Bethesda, MD, Jun 2002.

[16] S. D'Angelo et al., "Fault-Tolerant Voting Mechanism and Recovery Scheme for TMR FPGA-Based Systems," *Proc. 1998 International Symp. on Defect and Fault Tolerance in VLSI Systems*, pp. 233-240, Nov 1998.

[17] M.N. Liu and S. Whitaker, "Low Power SEU Immune CMOS Memory Circuits," *IEEE Trans. on Nuclear Science*, New Orleans, LA, vol. 39, pp. 1679-1684, Dec. 1992.

[18] R. Velazco et al., "Two CMOS Memory Cells Suitable for the Design of SEU-Tolerant VLSI Circuits," *IEEE Trans. on Nuclear Science*, Tucson, AZ, vol. 41, pp. 2229-2234, Dec 1994.

[19] D.R. Blum and J.G. Delgado-Frias, "A Fault-Tolerant Memory-Based Cell for a Reconfigurable DSP Processor," *Proc. 2003 International Conference on VLSI*, Las Vegas, NV, pp. 21-27, Jun 2003.

[20] C. Dick et al, "Configurable Logic for Digital Signal Processing", 1999. Available at: http://www.xilinx.com/products/logicore/dsp/config_logic4_99.pdf

[21] R. Hartenstein, "Coarse Grain Reconfigurable Architectures, *6th Asia and South Pacific Design Automation Conference*, 2001.

[22] J. Delgado-Frias, M. Myjak, F. Anderson, and D. Blum, "A Medium-Grain Reconfigurable Cell Array for DSP," *Proc. 3rd IASTED Int. Conf. On Circuits and Systems,* Cancun, Mexico, pp. 231-236, May 2003.

[23] N. Dutt and K. Choi, "Configurable Processors for Embedded Computing", *IEEE Computer*, 36(1), 2003, 120-123.

[24] P. Heysters et al, "A Reconfigurable Function Array Architecture for 3G and 4G Wireless Terminals", *Proc. World Wireless Congress*, San Francisco, USA, 2002, 399-405.

[25] E. Pauer et al, "Environment for Implementing DSP Algorithms in Reconfigurable Hardware, *Proc. High Performance Embedded Computing Workshop (HPEC),* 2000.

[26] R. Canham and A. Tyrrell, "An Embryonic Array with Improved Efficiency and Fault Tolerance," *2003 NASA/DoD Conference on Evolvable Hardware (EH-2003),* Chicago, IL, 2003.

[27] F. Anderson and J. Delgado-Frias, "A Reconfigurable Crossbar Switch for a DSP Array," *Proc. of The 2003 International Conference on VLSI,* Las Vegas, NV, Jun 2003.

[28] M.J. Myjak, D.R. Blum, and J.G. Delgado-Frias, "Enhanced Fault-Tolerant CMOS Memory Elements," *2004 IEEE International Midwest Symposium on Circuits and Systems (MWSCAS 2004)*, Hiroshima, Japan, submitted for publication.