A SEU Tolerant Distributed CLB RAM for In-Circuit Reconfiguration

Karthik.K.S[2], Shyam.S[2], Ramasubramanian.N[2], Noor.M[1], Shoaib.M[1] and Kamakoti.V[1]
Department of Computer Science and Engineering,
[1]Indian Institute of Technology Madras, Chennai, India
[2]National Institute of Technology, Tiruchirappalli, India
kama@cs.iitm.ernet.in, nrs@nitt.edu

Abstract— **Single Event Upsets (SEUs) are transient soft errors prominent in FPGA memories with radiation intensive environments. Hamming Error Correction Control (ECC) is a popular mechanism for SEU correction. However, Look Up Table (LUT) based Hamming designs make the ECC logic itself to be radiation prone. In this paper we propose an SEU tolerant Distributed RAM (ftDRAM) using Configurable Logic Blocks (CLBs) on the Xilinx Virtex FPGA families. Our ftDRAM incorporates unused CLB BlockRAMs for high speed On-Chip memories and SEU resistant Tri-State Buffers (BUFTs) for Hamming ECC. BUFTs are hardwired AND-OR logic elements within the Virtex fabric which are impervious to radiation upsets and BlockRAMs are LUT memories within a CLB which are often unused in typical designs. As a byproduct of this, we propose a novel, integrated In-Circuit Reconfiguration methodology for quick online SEU detection and correction using the ftDRAM. Device configuration data read back from the FPGA is compared at regular intervals with a copy of the original configuration stored in the ftDRAM. This helps in online SEU detection using simple BUFT based XOR comparators. In case of errors in the readback bitstream from the device, the correct configuration bits are written back from the ftDRAM. The ftDRAM which contains the original device configuration data is impervious to radiation errors owing to the Hamming ECC built into the design using the FPGA BUFTs. For typical LSI circuits, using our In-Circuit Reconfiguration methodology, we demonstrate a Mean Time To Repair (MTTR) of about $14.164ns$ which is very less compared to off-time specification in many high performance applications.**

I. Introduction

Reconfigurable computing and adaptive hardware is an emerging technology for many performance critical applications such as in defense and aerospace. Increasing use of Field Programmable Gate Arrays (FPGAs) in these has necessitated an enhanced degree of fault tolerance due to harsh operation environments. Many high reliability systems today entail error detection and correction as an integral component. A Single Event Upset (SEU) [1] is a common fault in SRAM-based FPGAs that operate in the presence of external radiation. These errors are particularly troublesome for memory elements as they store and propagate bits throughout the circuit [2]. Besides error resilience, high performance memories also demand faster operation speeds. Use of on-chip Configurable Logic Block (CLB) buffers for high speed single-port and dual-port RAMs is one popular solution for reduced latencies in memory design [3] [4]. CLBs are functional elements in FPGAs for constructing logic circuits [5]. A Xilinx Virtex CLB is made up of *Slices* which contain logic cells, the interconnection and configuration of which provides a specific functionality. Blocks of RAM within CLBs can be used to implement onchip synchronous memories [3] for high operation speeds and reduced routing overheads. Fault tolerance of these Distributed RAMs (DRAMs) is a vital requirement in mission critical systems [6]. Hardware redundancy for fault tolerance is a resource intensive solution. This involves duplication of logic known as Doubling with Comparison (DWC) which has a redundant area investment, power consumption and pin count. Time redundancy, which involves repetition of a computation on the same circuit, is effective only in the detection of highly transient errors (with a performance penalty) and hence cannot be used to detect errors in storage elements [4]. Error detection codes such as Hamming codes are used in Xilinx Virtex Devices [7] to detect SEUs in SRAM configuration memory. However in [7] there is no consideration of the fact that SEUs affecting the Hamming logic itself may make the entire system unreliable. In this paper we evaluate a technology dependent optimization that exploits the presence of unused blocks of RAM (BlockRAM) and Tri-State Buffers (BUFTs) in the Xilinx Virtex architecture. The aim is to provide a *high speed, SEU-tolerant, on-chip memory using redundant elements on the Virtex FPGA*. Single Event Transients (SETs) are short duration metastable glitches which tend to corrupt logic states. SETs have an effect on combinational outputs based on load capacitances. Transient pulses higher than half V_{dc} (the rail voltage) tend to propagate in buffers. However the probability of their occurance and propagation in limited by the critical charge induction of 0.02pC or a radiation intensity of $2Mev-cm^2mg$ for the worst case [8]. BUFTs in the Virtex architecture, although impervious to SEUs are susceptible to SETs but with a very low probability. These are a topic of transient analysis and short duration stability and are not considered in our discussion of buffer (BUFT) radiation stability in this paper. SEUs or soft errors have a ground level upset rate of 2×10^{-12} upsets/bit-hr [9] and probabilities of Multiple Event Upsets (MEUs) tend to downshoot from this low value to sub-atto rates. BUFT configuration is hardwired with no stored configuration bits. Hamming is a technique for detecting and correcting single bit soft errors in transmitted data [10]. Hamming Error Correction Control (ECC) requires that j parity (p) bits (or check bits) be transmitted with every

i data (d) bits. The algorithm is called a $(i+j, i)$ hamming code, because it requires $(i+j)$ bits to encode i bits of data. The goal of the Hamming code is to create a set of parity bits that overlap such that a single-bit error (the bit that is logically flipped in value) in a data bit or a parity bit can be detected and corrected. In the following sections we describe a fault tolerant memory design incorporating the Hamming ECC followed by its applications in Reconfigurable hardware.

II. Contribution of this paper

BlockRAM in a Xilinx Virtex FPGA is a 18Kb dual-port RAM which can be configured with various data/address aspect ratios such as $16K \times 1$ bit, $4K \times 4bits$, $2K \times 9bits$ or $512 \times 36bits$. Xilinx Virtex-II Pro FPGAs incorporate 444 blocks of such RAM [5] that can be used as lookup tables to store data and serve as on-chip memory. With two BlockRAMs occupying as much area as 24 CLBs or 192 LUTs [5], the BlockRAM cells may not seem to be a silicon-area efficient solution for onchip memories at first. However, since 444 of these BlockRAMs are anyhow integrated on the chip die but not used in typical designs, they can be considered free [11]. FPGA Tri-State Buffers (BUFTs) in the Virtex architecture are hard wired AND-OR logic elements which are often unused in logic designs, Fig.2. In this paper, we demonstrate a cross connection of these BUFTs to produce SEU proof Hamming functions for Error Correction Control of DRAM memories. The advantage of using BUFTs for designing the ECC logic is that their functionality is not based on the contents of any SRAM cells (no configuration bits), which may get upset. The only aspects of the BUFTs which are controlled by configuration memory cells are the routing pips which connect them together [12]. Upsetting one of these cells would only result in temporarily disconnecting one of the inputs or outputs of the BUFTs. Such an upset would not affect the output of the ECC logic. In fact, this ECC design is completely impervious to single upset (SEU) failure. Use of these BlockRAMS for on-chip buffers and BUFTs for ECC is the key contribution of this paper. These do not reduce the logic or RAM in the circuit as the entire methodology is based on unused BlockRAMs and BUFTs. As a byproduct of this fault tolerant memory design, we also propose a novel incircuit reconfiguration methodology for Fast Dynamic Reconfiguration of FPGA Devices made feasible by the reprogrammability and readback capabilities of the Xilinx Virtex architecture. This run-time reconfiguration makes FPGA designs fault tolerant and bestows them with functional evolution. We also present applications of our fault tolerant memory design in Network on Chips (NoCs) which require a large number of high speed on chip buffers.

III. Fault Tolerant DRAM

The Error Correction Control (ECC) functions described in our DRAM model use hamming codes. This involves transmitting data with multiple check bits and decoding the associated check bits when receiving data to detect errors. The check bits are parity bits generated in parallel by XORing certain bits in the original data word. If bit error(s) are introduced

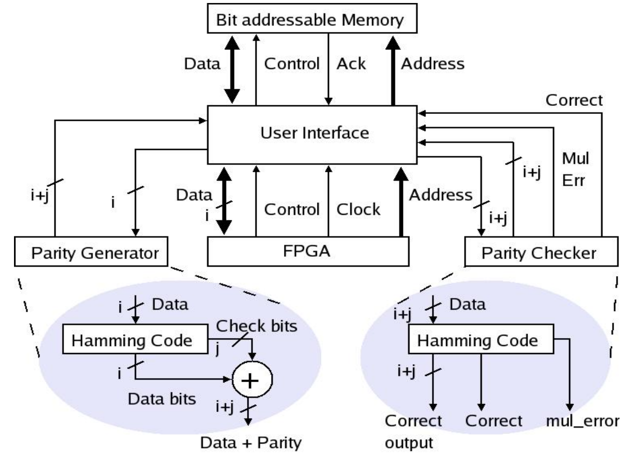


Fig. 1. The Proposed Hamming ECC Fault Tolerant DRAM Memory Model

in the codeword, several check bits show parity errors after decoding the retrieved codeword. The combination of these check bit errors display the nature of the error. In addition, the position of any single bit error is identified from the check bits. Fig.1 shows the system architecture of the proposed memory design. The *bit addressable memory* can read/write a single bit data from the given address. This incorporates the use of CLB RAMs. The *parity generator* takes as input i bits of data and gives back a $(i+j)$ bits of a data+parity word. The parity checker takes a $(i+j)$ bit word and checks it for errors. If no errors are detected, the *correct pin* is set. In case of a single bit error, the pin is reset and the corrected data is sent to the user as well as to the memory for correction. For multiple errros, the *correct pin* is reset, mul_error is set and the data output is set to high impedance.

Modeling with Buffers: In any self correcting mechanism, the error correcting component should not be suscpetible to errors. As described in Sec.II, buffer gates in the FPGA (BUFTs) are not susceptible to soft errors. We have used buffer gates entirely to model the parity generator and the hamming parity checker. Buffer gate $notif1$ was used to construct all the other required gates. The buffer gate $notif1$ and the wor construct in verilog were used to implement the nor gate, which is universal. Using this, all other gates were constructed.

Two input NOR gate construction: Let the inputs be a,b. To obtain $(a+b)'$, we pass $a+b$ to $notif1$, with the control being 1. The verilog code is as follows.

```
module my$_$nor(a,b,c)
      input a,b;
      output c;
      wor temp;
      assign temp = a;
      assign temp = b;
      notif1 n1(c,temp,1);
end module
```

The If-Else contruct: The if-else construct is needed to perform toggling of bits if found to be erroneous. $notif1$ gates

are used and a *wor* verilog construct is required. The verilog code is as follows.

```
module my$_$if(a,b,c) // if (~b)
     input a,b;        // then return
     output c;         // a else
     wor c;            // return (~a)
     wire na,nb;
     notif1 n1(na,a,1);
     notif1 n1(nb,b,1);
     notif1 n1(c,a,nb);
     notif1 n1(c,na,nb);
end module
```

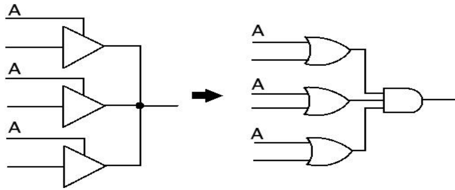
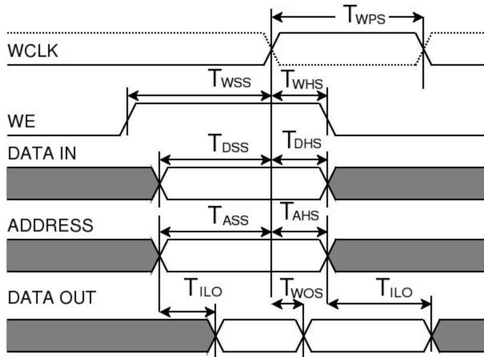


Fig. 2. The Virtex Tri-State Buffers: Equivalent to Wired AND-OR Logic.

Operation: Our model proposes a fault tolerant memory system that can detect and correct single bit errors and detect multiple errors. The user interface provides an abstraction of the bit addressable memory as a byte addressable memory and takes care of error detection and correction. The whole process is transparent to the user. The basic operations are read, write and correct. The order of priority is $write > correction > read$. Whenever the user wants to write to memory, the address and data are supplied and the $write_enable$ pin is set. The parity generator supplies the $(i+j)$ bit word (data + parity) to be written into memory. For a read operation, the user provides the address and sets the $read_enable$ pin. Now if the memory is not busy. i.e no write/correction is being performed. A $(i+j)$ bit-word is read. The parity checker finds out the validity of the word. In case of no error, the correct pin is set, the i bit



Configure dev with encoded bit stream. Copy it into Mem: i [D]

Compute Hamming ECC and Syndrome Matrix for Correction.

Fig. 3. The DRAM ECC Cycle Clock Timing Diagram.

data is extracted and given to the user. If a single error has been detected, the correct pin is reset, the correction process is triggered and then the user is supplied with the corrected value. For multiple errors, the correct pin is reset, mul_error pin is set. As the user should not be provided with the wrong value, the output is set to high impedance. The read-write timing diagram for the fault tolerant memory model is shown in Fig.3.

IV. Application: InCircuit Reconfiguration

A key application of the proposed fault tolerant memory design is an incircuit reconfiguration methodology for LUT based FPGA designs. Applications which abundantly employ reconfigurable systems, for example space applications, demand a severe necessity for fault tolerant architectures. Once in space, FPGAs are susceptible to several kinds of errors like: Single Even Upset (SEU), Single Hard Error (SHE), Single Event Burnout (SEB), Single Event Gate Rupture (SEGR), Single Event Effect (SEE) and Multiple Bit Upset (MBU). Except SEUs and MBUs, all the others cause permanent damage to the hardware. As the probability of multiple errors occuring in the same system is infinitesimal [9], the objective of our incircuit reconfiguration model is to correct a single bit error (SEUs) and to detect multiple errors using the memory model described in Sec.III. For mitigating SEUs in FPGAs, numerous models have been proposed in the literature, one of the most popular among them is the Triple Modular Redundancy (TMR) [13] which creates two extra copies of every data element. To verify the correctness, a voter circuit is used which outputs the value that is stored by at least two of the three copies. In case two of the copies are wrong, the result of the voter circuit is erroneous. Thus, this model is unable to detect multiple errors in two different modules. Moreover, having two extra copies is costly. Another model (Cluster-based Detection of SEU-caused Errors in LUTs of SRAM based FPGAs) uses simple parity to detect errors. Though this is the cheapest method to detect errors, it can detect odd-errors only [4]. Yet another model [7] performs single error correction and double error detection using Hamming Codes. In this model, the errors detected are not corrected back into the memory. If another error occurs in the same word, the model will detect it as a case of multiple error though this condition can be prevented. A major problem with all these designs, as mentioned in Sec.II, is that the ECC logic itself may not be tolerant to radiation upsets. This necessitates a robust ECC design impervious to SEUs. Our Incircuit Reconfiguration design (Fig.4) which uses the BUFT and BlockRAM memory model described in Sec.III achieves this error resilience. The Hamming Error Correcting Code (ECC) is capable of correcting single bit errors and detecting multiple errors. The above process of handling error(s) is done by using additional check-bits that are clubbed together with data-bits. An 8 bit data word, for example, requires 4 parity bits to handle single bit errors and an extra bit to detect multiple errors. An example of the basic algorithm is as follows - All bit positions that are powers of two are used as parity bits (positions $1, 2, 4, 8, 16, 32, 64$

etc). All other bit positions are for the data to be encoded (positions $3, 5, 7, 9, 10, 11, 12, 13, 14, 15, 17$ etc). The parity bit at position 2^k check bits in positions having bit k set in their binary representation [14]. For example, 1011 is encoded into 01100110. Even parity is followed in the system description to follow. The ftDRAM with the BUFT ECC which is resilient to SEU errors is used to store FPGA device configuration data. The stored ftDRAM configuration is error tolerant as the memory is protected with a hamming ECC built using hardwire BUFTs in the FPGA. This stored configuration bitstream is compared with the readback data from the FPGA device at regular intervals. In case of an SEU in the FPGA device logic, the Hamming ECC routes corrected configuration data to be written back to the device. The time utilized for the reading of the device configuration, its correction and writing back is called the Mean Time To Repair (MTTR). This In-Circuit Reconfiguration methodology is simple as well as error tolerant.

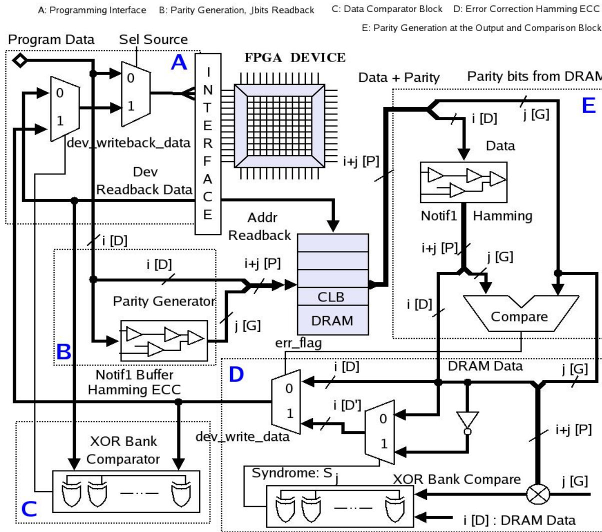


Fig. 4. The respectively.

A. Design Specifics

Fig.4 shows the system design of the incircuit reconfiguration of FPGAs for fault tolerance. FPGA configuration bits are readback for SEU correction using InCircuit Reconfiguration (IR) as described in [15] [16] and [17]. *Block A* in the design distinguishes fresh FPGA configuration from IR for SEU mitigation. The hamming ECC uses a codeword that is an (un)ordered union of the parity and the data bit vectors. It is described by the ordered set $(i + j, i)$. As an example, let us consider a hamming codeword with i data bits and j parity bits concatenated as $(d_i, d_{i-1} \dots d_1, p_j, p_{j-1} \dots p_1)$. **Parity bits** Given the data set $D = (d_i, d_{i-1} \dots d_1)$ of i elements, we can define a j element parity set $P = (p_i, p_{i-1} \dots p_1)$ with a minimum number of check bits for single bit error-correction constrained by Eq.(1). [7]

$$i + j + 1 \leq 2^j \tag{1}$$

The parity set may be defined as a modulo-2 sum of a peremptive subset of randomly chosen k elements $(d_x, d_y, d_z \dots k\ elements)$ from D where $\binom{i}{k} \geq j$. This parity bit generation and regeneration is done in *Blocks B and E* respectively in Fig.4. **Encoding** The parity matrix, $[P]$, which encodes a given data of i bits into a codeword, is expressed in Eq.(2)

$$[P] = [D] \bullet [G] \tag{2}$$

where, $[D]$ is the i bit data matrix and $[G]$ is the generator matrix. The $[G]$ matrix consists of an identity matrix $[I]$ and a creation matrix $[C]$ of order $i \times (i + j)$ and is given by $[G] = [I : C]$. The creation matrix is a concatenation of column vectors containing a 1 in the row corresponding to each of the k data bits included in its computation and a 0 is all other rows. The encoded data is stored in the CLB RAM designed in Sec.III without the ECC logic. **Decoding** Upon receiving a dynamic reconfiguration request by the FPGA after a given number of operation cycles, the Hamming codeword stored in the DRAM is compared with the readback configuration data in *Block C*. Decoding of the Hamming ECC in a situation without errors is straight forward. Disregarding the parity bits leaves us with the data. For example, if an an encoding produces 1011010 as the hamming vector $[P]$ in a $(7, 4)$ encoding scheme we have after pruning out the first 3 parity bits, 1010 as the 4 bit data. In case of an error, which is detected by recomputing the parity bits, we construct a $j \times (i + j)$ parity check matrix $[H]$ such that the jth row containes a 1 in the jth column corresponding to the jth parity bit and all of the k data bits included in its parity calculation. The parity check matrix is ordered as $[H] = [C : I]$. Multiplying the matrix $[H]$ of order $j \times (i+j)$ with the data code matrix of order $(i+j) \times 1$ produces a matrix $[S]$ of order $j \times 1$ called the *syndrome*. In other words, the data code vector multiplies with the transpose of the generator matrix to produce the syndrome vector. This is shown in Eq.(3)

$$[S] = [D, P] \bullet [G^{'}] \tag{3}$$

If all the elements of the syndrome vector are zeros, no error is reported. Any other non-zero result reflects the column in which the error has occured in $[H]$. If the syndrome vector elements correspond to the elements of the qth column of $[H]$, it indicates an error in the $(q - j)^{th}$ data bit which is corrected by inverting the vector corresponding to the syndrome as shown in *Block D* in Fig.4. This corrected vector is passed on to the XOR comparator to compare with the readback device bitstream. In case of an SEU error, the DRAM configuration data is written back to the FPGA in the following device configuration cycle else the existing FPGA configuration data is retained.

The Reconfiguration algorithm using the fault tolerant DRAM is shown illustrated in 4 and described in Algorithm.1. The algorithm requires a FPGA readback capability and a configuration map scan using an SelectMAP/ JTAG interface. A Mean-Time-to-Repair (MTTR) requirement of atleast $m + n + p$ should also be ensured where, n is the number of clock

cycles to readback data from the FPGA device, m to read BlockRAM data and p to write back device configuration data. A $capture_enable$ flag is raised to initiate correction and the $config_data$ pointer holds the address of the last memory location containing the device configuration data. mem_data_y holds the parity matrix. z is also iterated upto the $config_data$ pointer and raises an error flag if there is a mismatch in the generator matrix calculated by the BUFT ECC (G_z) and that stored in the memory (G_z^{mem}). In case of an error, if the syndrome matrix, S_z, generated from the readback data matches the stored device configuration data, there is no need for correcting that memory location. In case there is a mismatch, dev_write_data is updated to D_z the original configuration data. This forms the correction part of the algorithm (ECC cycle). After error correction, if the err_flag is set to 1 at the end of the ECC Cycle ($i.e$ at the end of $m + n$ clock cycles), new data dev_write_back to be written back to the device is updated to the corrected write data dev_write_data.

Algorithm 1 DYNAMICPARTIALRECONFIGURATION(n)

Require: Incircuit FPGA Configuration Readback.
Ensure: SelectMAP/ JTAG Boundary Scan/ μController.
Ensure: $MTTR \geq m + n + p$

1: **for all** x such that $0 \leq x < capture_enable$ **do**
2: **for all** y such that $y \leq config_data$ **do**
3: $[D]_y = [d_i, d_{i-1} \dots d_1]$
4: $[G]_y = [p_j, p_{j-1} \dots p_1]$
5: $[P]_y = [D_y : G_y]$
6: $mem_data_y = [P]_y$ //write memory
7: **end for**
8: **for all** z such that $z \leq mem_data$ **do**
9: **if** $[G]_z = G_z^{mem}$ **then**
10: **return** $err_flag = 0$
11: **return** Break;
12: **end if**
13: $err_flag = 1$
14: $S_z = P_z \bullet G_z^T$
15: **for all** y such that $y \leq i$ **do**
16: **if** $[S]_z == D_z$ **then**
17: **return** Break;
18: **end if**
19: $dev_write_data = D_z$
20: **end for**
21: **end for** // ECC Cycle
22: **if** $err_flag = 1$ **then**
23: **return** $dev_write_back = dev_write_data$
24: **return** Break;
25: **end if**
26: **end for**

V. EXPERIMENTAL RESULTS

The fault-tolerant memory model using BlockRAMs and BUFTs was simulated on a Xilinx Virtex II Pro platform FPGA Device XC2VP2-7FG256. The Hamming ECC was implemented on the device using BUFTs and Table.I shows the BUFT usage and clock speeds for different dataword sizes used by the ECC to generate parity information. For wordlengths beyond 64 bits, the BUFTs on the Virtex device were insufficient. However, a 64 bit data word which constitutes about 16 BlockRAM addresses for a $4K \times 4bits$, can be used to generate parity information for clustered memory addresses. This provides error tolerance for data clusters in about 16 memory locations. Assuming this kind of operation speed we have a typical clock period, T_{WCLK}, of $7.082ns$. Fig.5 shows the operation cycles (device read, memory write, memory read and device write cycles) of a typical Incircuit Reconfiguration (IR) model. Assuming an asynchronous memory read, based on this operation cycle delays, we can estimate a Mean Time To Repair (MTTR) of about $p + 2$ clock cycles, where p is the time taken to write back the data into the device. Not including the programming time p in the MTTR computation, we arrive at an approximate MTTR order of about $14.164ns$. This repair time is quite less compared to a configuration power cycle of about $20ms$ [12] for typical FPGA configurations. Also, typical off-times in FPGA space circuits can have a tolerance of a few nano-seconds [9]. This quick MTTR cycle would mean SEU tolerant FPGA logic which can be repaired at regular intervals to ensure correct operation. This is built with the help of unused device elements (BlockRAMs and BUFTs) along with lower off-times. Table.II shows FPGA area occupancy and device operation speeds for varying memory sizes on the FPGA. It shows a memory operation with a clock period of about $7.376ns$ for a $4kb \times 8bit$ DRAM.

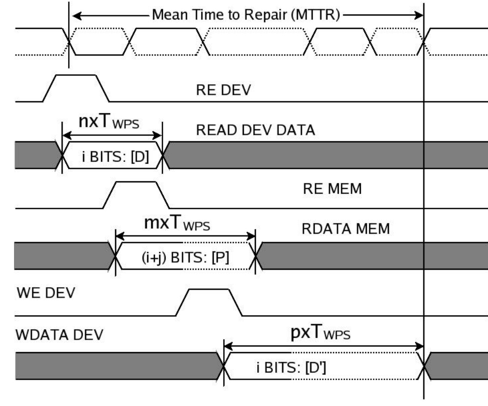


Fig. 5. The Incircuit Reconfiguration Timing Diagram.

VI. FUTURE WORK

The technique proposed in this paper is limited to the hugely popular Xilinx Virtex families of FPGAs which has flexible readback capabilities, large amount of unused BlockRAM and radiation immune buffers (BUFTs) [11] [12] [18]. The technique seeks to address SEU mitigation in radiation intensive environments. Multiple upset (or MEU) probabilities in practical environments are infinitesimal [9]. However, they have a finite probability of occurence. The incircuit reconfiguration

TABLE I

TEST CASES WITH VARIABLE DATAWORD LENGTHS FOR 1KB FTDRAM ON XILINX VIRTEX II XC2VP2-7FG256 DEVICE

Data Width (i)	# Slices	# Slice FF	4 i/p LUTs	BUFTs	$T_{WCLK-min}$	$T_{CLKH-min}$
8 Bits	588	351	1022	30	6.237 nS	3.522 nS
16 Bits	645	365	1193	46	6.485 nS	3.370 nS
32 Bits	708	389	1307	78	6.876 nS	3.407 nS
48 Bits	756	414	1349	106	7.203 nS	3.422 nS
64 Bits	780	431	1382	118	7.082 nS	3.432 nS

TABLE II

TEST CASES FOR VARIABLE MEMORY SIZES WITH AND WITHOUT ERROR CHECKS

Mem Size	ERR CHK	Slices	Slice FF	4 i/p LUTs	Min Period	Max T_{hold}
8×1KB	no	458	286	831	5.206 nS	3.290 nS
8×1KB	yes	588	351	1022	6.237 nS	3.522 nS
8×2KB	no	766	484	1384	5.727 nS	3.290 nS
8×2KB	yes	911	547	1663	7.370 nS	3.508 nS
8×4KB	no	1462	768	2132	5.854 nS	3.290 nS
8×4KB	yes	1543	940	2793	7.376 nS	3.522 nS

technique proposed in this paper helps in the detection of these using the ECC to raise an error flag which may be further used to do offline correction. The choice of Hamming codes for an ECC procedure is owing to the fact that this provides MEU detection and a simpler XOR implementation using the BUFT gates. Comparison of different ECC implementations, online MEU correction and porting of the technique to various FPGA platforms is the future direction of this work.

VII. CONCLUSION

In this paper we describe the top-down design of a robust, SEU tolerant Distributed RAM (ftDRAM). The key benefit of the proposed memory design is that the it is built using BUFTs and BlockRAMs which are resources left unused in typical FPGA designs. We make use of the BUFTs to construct the Hamming corrector used an ECC the ftDRAM design. This makes our ECC logic SEU proof unlike other designs in the literature which make use of radiation prone CLBs to construct the ECC logic. Moreover, the use of BlockRAMs within Configurable Logic Blocks (CLBs) for storing the FPGA configuration data provides a very fast on-chip memory. As an application of our ftDRAM model, we demonstrate an In-Circuit Reconfiguration methodlogy for SEU mitigation in systems exposed to high intensity radiations. The stored configuration data in the ftDRAM is compared with device bitstreams readback from the FPGA at regular intervals of time. An error in the device configuration is detected based on this comparison. This reconfiguration model also enables us to perform online error correction by writing back the correct configuration data from the ftDRAM. We demonstrate our In-Circuit methodology to have an estimated Mean Time to Repair (MTTR) of about $14.164ns$ for a $4kb \times 8bit$ DRAM.

REFERENCES

[1] *Understanding Soft and Firm Errors in Semiconductor Devices – Questions and Answers* Technical Report by Actel, http://www.actel.com/documents/SER_FAQ.pdf, December 2002.

[2] Octavian-Dumitru Mocanu and Joan Oliver, *Fault-Tolerant Memory Architecture Against Radiation-Dependent Errors: A Mixed Error Control Approach* Journal of Electronic Testing, Volume 14, Numbers 1-2, pp.169-180, February 1999

[3] *Using Look-Up Tables as Distributed RAM in Spartan 3 Generation FPGAs* Xilinx Application Note, XAPP464, March 2005.

[4] E. S. S.Reddy, V. Chandrasekhar, M. Sashikanth, V. Kamakoti, *Cluster Based Detection of SEU-Caused Errors in LUTs of SRAM based FPGAs* Proc. Asia South Pacific Design Automation Conf., pp.1200-1203, 2005.

[5] *Virtex Series Configuration Architecture User Guide* Xilinx Application Note, XAPP151, October 2004.

[6] Miron Abramovici, John M. Emmert, Charles E. Stroud, Roving Stars, *An Integrated Approach To On-Line Testing, Diagnosis, And Fault Tolerance For Fpgas In Adaptive Computing Systems* Proceedings of the The 3rd NASA/DoD Workshop on Evolvable Hardware, p.73, July 12-14, 2001.

[7] *Error Detection and Correction in Virtex-II Pro Devices* Xilinx Application Note, XAPP645, February 2004.

[8] *Single Event Transient (SET), MAPLD Short Course* klabs.org/.../Tutorial/MiniCourses/radiation_mapld_2002/ Radiation_Course_2002_MAPLD.hardcopyF_SET.ppt

[9] *Presentation by E. Normand* December 1998, to the C-17 Avionics Group, http://www.boeing.com/assocproducts/radiationlab/publications/,

[10] Elaine Ou, Woodward Yang, *Fast Error-Correcting Circuits for Fault-Tolerant Memory* Proceedings of the International Workshop on Memory Technology, Design and Testing, pp.8-12, March 07-11, 2004

[11] *FIFOs Using Virtex-II Block RAM* Xilinx Application Note, XAPP258 (v1.4) January 7, 2005.

[12] C. Carmichael, E. Fuller, P. Blain and M. Caffrey, *SEU mitigation techniques for Virtex FPGAs in space application* MAPLD'99 Poster, page 24, 1999.

[13] F. Lima Kastensmidt, L. Sterpone, L. Carro, M. Sonza Reorda, *On the Optimal Design of Triple Modular Redundancy Logic for SRAM-based FPGAs* Proceedings of the conference on Design, Automation and Test in Europe, pp.1290-1295, March 07-11, 2005

[14] *Hamming Code General Algorithm* http:w//en.wikipedia.org/wiki/Hamming_code#General_algorithm

[15] Wolfgang Hoflich,*Using the CX4000 Readback Capability* Xilinx Application Note, XAPP015.000.

[16] Cristiana Bolchini, Davide Quarta, Marco D. Santambrogio, *SEU mitigation for sram-based fpgas through dynamic partial reconfiguration* Proceedings of the 17th great lakes symposium on Great lakes symposium on VLSI, p.55-60, 2007.

[17] Fernanda Lima, Luigi Carro, Ricardo Reis, *Designing fault tolerant systems into SRAM-based FPGAs,* Proceedings of the 40th conference on Design automation, June 02-06, 2003, Anaheim, CA, USA

[18] *Xilinx Virtex-V Capabilities - Block RAM* http://www.xilinx.com/ products/silicon_solutions/fpgas/virtex/virtex5/capabilities/block_ram.htm