Selective Triple Modular Redundancy (STMR) Based Single-Event Upset (SEU) Tolerant Synthesis for FPGAs

Praveen Kumar Samudrala, *Member, IEEE*, Jeremy Ramos, *Member, IEEE*, and Srinivas Katkoori, *Senior Member, IEEE*

***Abstract*—We present a design technique for hardening combinational circuits mapped onto Xilinx Virtex FPGAs against single-event upsets (SEUs). The signal probabilities of the lines can be used to detect SEU sensitive subcircuits of a given combinational circuit. The circuit can be hardened against SEUs by selectively applying triple modular redundancy (STMR) to these sensitive subcircuits. However, there is an increase in the number of the voter circuits required for the STMR circuits. Virtex has abundant number of tri-state buffers that can be employed to construct SEU immune majority voter circuits. We also present a SEU fault insertion simulator designed to introduce errors representing SEUs in the circuits. STMR method is thoroughly tested on MCNC'91 benchmarks. With a small loss of SEU immunity, the proposed STMR method can greatly reduce the area overhead of the hardened circuit when compared to the state-of-the-art triple modular redundancy (TMR). STMR method along with the readback and reconfiguration feature of Virtex can result in very high SEU immunity.**

***Index Terms*—Field programmable gate array (FPGA), single-event upset (SEU), triple modular redundancy (TMR).**

I. Introduction

FIELD-PROGRAMMABLE gate arrays (FPGAs) are being increasingly used for space applications because of low cost, reconfigurability, and low design turn-around time. FPGAs such as Xilinx Virtex are a class of programmable devices which use static random access memory (SRAM) cells for implementing logic and interconnections of a mapped design. These cells are highly susceptible to a category of radiation effect known as single-event upset (SEU) [1], [2].

SEUs are a major cause of concern for SRAM based FPGAs such as Virtex, because of the following reasons:

1) Although SEUs show up as soft errors in combinational circuits, they transform into more serious permanent faults when they are mapped to FPGAs. This is because the same combinational circuits are mapped on the FPGA using look up tables (LUTs), which consist of SRAM cells. Hence, an SEU in these cells could be latched, thus transforming the transient fault into a permanent fault.
2) The interconnection of the FPGA is also controlled using the data stored in SRAM cells.
3) Since the information defining the functionality of a FPGA is also stored in memory cells, an upset in them could lead to malfunctioning of the device and prove fatal to the mission. Hence, such SEUs have to be meticulously addressed for a mission employing SRAM based FPGAs.

It is also reported in [6] that SEUs are a major cause of concern for Virtex. Thus, we focus on hardening the design mapped on the Virtex FPGA.

Design hardening is one of the techniques employed to mitigate SEUs. Hardening by design include introducing hardware and/or software redundancy. Electronic devices intended for space applications can be designed from a library of SEU tolerant basic gates and memory cells. Such structures of gates and SRAM cells have been proposed in the previous decade. A SEU hardened version of a boolean gate is obtained by modifying its basic structure by adding a few additional transistors.

Whitaker design [8]–[10], Dice design [11], HIT cells [15], and Barry-Dooley design [16], [17] are some of the SEU tolerant SRAM cells that exist in the literature. These methods, although very effective, unfortunately cannot be applied easily to FPGAs. This is because FPGAs are COTS (commercially off the shelf) devices that are prefabricated. The whole design cycle has to be modified and is cost-prohibitive.

An alternative to using SEU hardened library of cells is to apply modular redundancy. Triple Module Redundancy (TMR), first proposed by Von Neumann [24], is one such technique where a module is replicated three times and the output extracted from a majority voter as shown in Fig. 1(a). The main drawbacks of applying modular redundancy technique are: excessive area overhead (i.e., increase in board space and system payload). The hardened design has 200% more area than the original circuit. In the context of space based applications, this implies an increase of the payload by two times.

TMR system can withstand only single upsets at any instant of time, thus, if two redundant modules are simultaneously upset, then the output cannot be guaranteed to be correct. Also, if two modules are permanently damaged, the whole TMR system has to be discarded. The redundant system is considered SEU tolerant under the assumption that the voter circuit is completely immune to SEUs.

Manuscript received June 19, 2003; revised September 26, 2003 and June 20, 2004. This work was funded by Honeywell Inc., Clearwater, FL (2001 SASSO/CSO Academic Initiatives IR&D Program). The work of P. K. Samudrala was performed when the author was with the University of South Florida, Tampa, FL 33620 USA. Patent pending at USPTO as of June 2004.

P. K. Samudrala is with the Space Micro, Inc., San Diego, CA, USA.

J. Ramos is with Honeywell Space Systems Inc., Clearwater, FL 33620 USA.

S. Katkoori, CSE Department, University of South Florida, Tampa, FL 33620 USA (e-mail: katkoori@csee.usf.edu).

Digital Object Identifier 10.1109/TNS.2004.834955

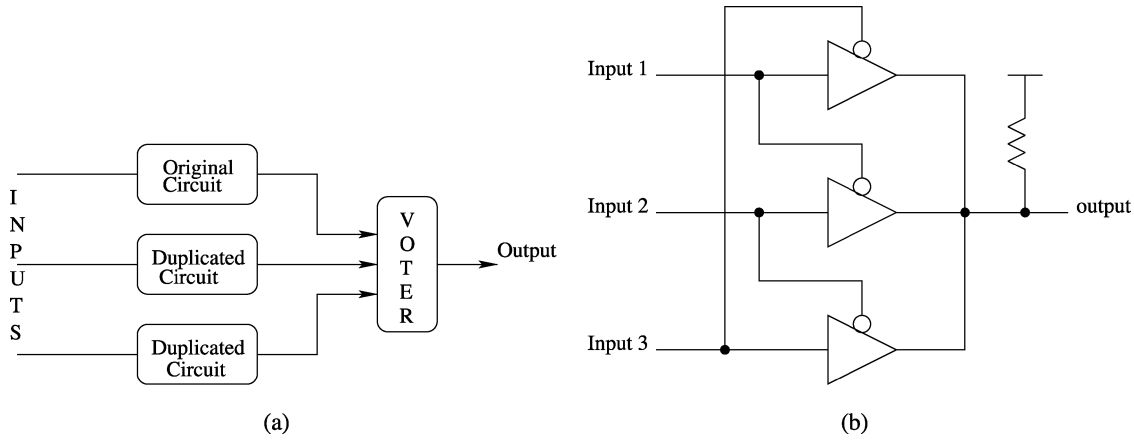

Fig. 1. (a) Triple Modular Redundancy. (b) Tristate voter in virtex FPGA.

TABLE I
SIGNAL PROBABILITY COMPUTATION AT THE OUTPUT OF A BOOLEAN GATE

Gate Type	P_{out}
AND	$\prod_i P_i$
NAND	1 - $\prod_i P_i$
OR	$\sum_i P_i$ - $\prod_i P_i$
NOR	1 - ($\sum_i P_i$ - $\prod_i P_i$)
XOR	$\sum_{i,j} P(i)(1-P(j))$
XNOR	1 - ($\sum_{i,j} P(i)(1-P(j))$)

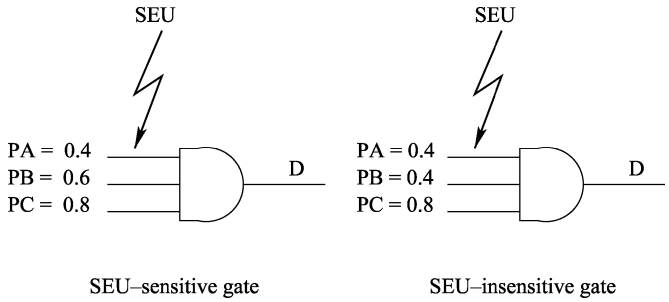

Fig. 2. Sensitive and insensitive gates ($P_{\text{threshold}} = 0.5$).

The correct implementation of TMR depends on the type of the module to be hardened. For example, the method of implementing TMR for sequential circuits is different from that of combinational circuits.

A. SEU Mitigation Techniques for Virtex FPGAs

Virtex series have special features that can be used for SEU hardening:

Readback and Reconfiguration: The configuration data of the Virtex can be downloaded from the device. This is called the "readback" feature. The "readback" data can be compared with the uncorrupted data to identify any occurrence of an SEU. In case of an upset, the device can be "reconfigured" with the uncorrupted data. The readback and reconfiguration procedure could be employed periodically to detect and correct the occurrence of an SEU. The total time taken for readback and reconfiguration is 20 ms (which is highly undesirable).

Partial Reconfiguration of Virtex FPGAs: The SEU correction procedure could be speeded by partially reconfiguring the SEU affected part of the configuration memory. As a result only the corrupted data can be reloaded, effectively reducing the correction time to 3 μs [6].

```
1 Algorithm is_sensitive (gate G, Threshold P_threshold)
2 Inputs: A Boolean Gate G, and 0 ≤ P_threshold ≤ 1.
3 Output: Returns TRUE if sensitive else FALSE.
4 begin
5   if(gate type of G is OR or NOR) then
6       dominant_value ← '1'
7   else
8       if(gate type of G is AND or NAND) then
9         dominant_value ← '0'
10      end if
11  end if
12  return_value ← FALSE
13  foreach input i of gate G do
14    foreach input j of gate G j ≠ i do
15      if(probability of input j taking
            not(dominant_value) ≥ P_threshold) then
16                  return_value ← TRUE
17                  break
18      else
19                  return_value ← FALSE
20                  break
21      end if
22    endfor
23    if(return_value = TRUE) then
24                  break
25    end if
26  endfor
27  return return_value
28 end Algorithm
```

Fig. 3. Algorithm to detect a gate's sensitivity to SEU.

Triple Modular Redundancy: TMR is the most robust mitigation technique, but the main drawback of using TMR for SRAM based FPGAs is that the voter circuit has to be implemented using SRAM cells which themselves are highly susceptible to upsets. Unlike other SRAM based FPGAs, Virtex has tristate buffers (BUFT), which can be used to build a SEU tolerant voter circuit [6] shown in Fig. 1(b).

The elements that are susceptible to SEUs in this voter are the routing pips (Programmable Interconnection Points), which are controlled by SRAM cells. However, any upset in these cells would temporarily disconnect the inputs or outputs of one of

```
1 Algorithm Selective_TMR(Circuit C)
2 begin
3   /* Propagate Input Probabilities */
4   MaxLevel ← Levelize (C)
5   foreach level l from 0 to MaxLevel do
6     for each gate g at level l do
7       Compute the signal probability of g's output
8     endfor
9   endfor
10  /* Identify SEU sensitive gates that */
11  /* can affect primary outputs        */
12  L ← φ
13  foreach primary output PI_out of the circuit do
14      L → L ∪ {g_PIout} /* Add primary output gate */
15      Identify_Sensitive_Subcircuits (C, L, PI_out)
16  endfor
17  /* Introduce tmr for the subcircuit */
18  C_stmr ← Introduce_TMR_at_subckt_level (C, L)
19  return C_stmr
20 end Algorithm
```

Fig. 4. Main algorithm to perform selective TMR.

```
1 Algorithm Identify_Sensitive_Subcircuits (C, L, l_out)
2 Inputs: Circuit C, list of sensitive gates, an output line
3 begin
4   Let g_out be the gate whose output is l_out
5   foreach input l_in of gate g_out do
6         Let g_in be the gate whose output is l_in
7         if(is_sensitive (g_in) = TRUE)then
8             L ← L ∪ {g_in}
9             Identify_Sensitive_Subcircuits (C, L, l_in)
10        endif
11  endfor
12 end Algorithm
```

Fig. 5. Algorithm that recursively identifies SEU-sensitive gates.

the BUFTs but not affect the output of the voter. Hence, this method is resilient to single upsets but is prone to fail in case of multiple upsets. The correct implementation of TMR with the Virtex FPGA depends on various factors such as the size and the type of the module to be mitigated.

TMR can be implemented based on the module size in four ways [5]: 1) Module Redundancy; 2) Logic Partitioning Redundancy; 3) Logic Duplication Redundancy; and 4) Device Redundancy. It can also be implemented based on the type of the logic [6]: 1) Throughput logic; 2) State machine logic; 3) I/O logic; and 4) Specialized subsystems (such as clock signal).

A method to assess the probability that an SEU occurring at a node in the circuit causing a error at the output of a flip-flop has been proposed in [23]. A program named SUPER II, that evaluates a circuit to determine the nodes with the highest probability of having an SEU cause an error in the output of a flip-flop has been proposed by the authors.

We propose selective triple modular redundancy (STMR) which extends the basic TMR technique by identifying SEU "sensitive" gates in a given circuit and then introducing TMR selectively at these gates. The sensitivity of a gate to an SEU is determined by the signal probabilities of its inputs. We assume that the input environment is specified by the user in terms of signal probabilities at the primary inputs of the circuit. Given

```
1 Algorithm Introduce_TMR_at_subckt_level (C, L)
2 Inputs: Circuit C and a list of sensitive gates
3 begin
4   C_stmr ← C
5   foreach level l from 0 to MaxLevel do
6     for each gate g at level l do
7       Make three copies of g say g1, g2, and g3
8       and update C_stmr
9       if(the fanin gate of g (say gate v is triplicated)) then
10          connect the output v1 to input of g1,
11                  the output v2 to input of g2,
12                  the output v3 to input of g3,
13      else
14          connect the output of v to input of g1, g2, and
    g3
15      endif
16      insert_voter_flag ← FALSE
17      foreach gate f in fanout of g do
18        if(is_sensitive (f) = FALSE) then
19          insert_voter_flag ← TRUE
20          break
21        endif
22      endfor
23      if(insert_voter = TRUE) then
24          Insert a majority voter whose inputs
25              are fed by g1, g2, and g3
26          Connect the output of the majority voter
27              to all fanout gates that are insensitive
28      endif
29    endfor
30  endfor
31  return C_stmr
32 end Algorithm
```

Fig. 6. Algorithm to introduce TMR at the subcircuit level.

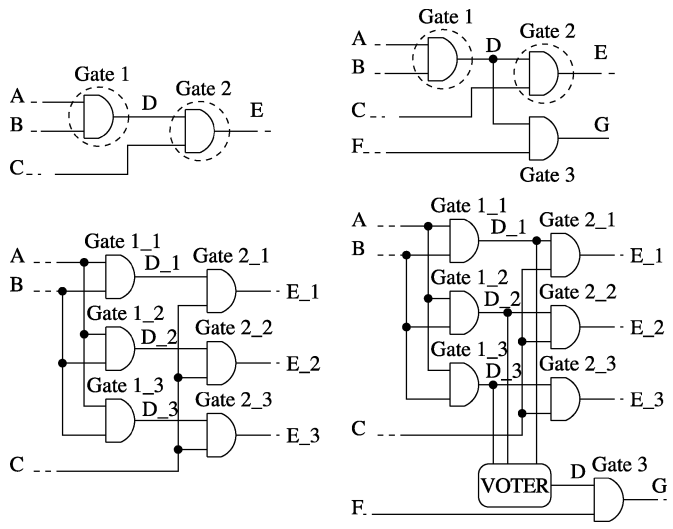

Fig. 7. (a) Connections between two triplicated modules (without fanout); (b) connections between two triplicated modules (with fanout).

a gate-level implementation and the input signal probabilities, we propagate them to compute the signal probability of each internal node. A gate is sensitive if an SEU on any one of the inputs is likely to be propagated to the output of the gate. The advantage of this technique is that the area overhead is typically much smaller than that of the full TMR.

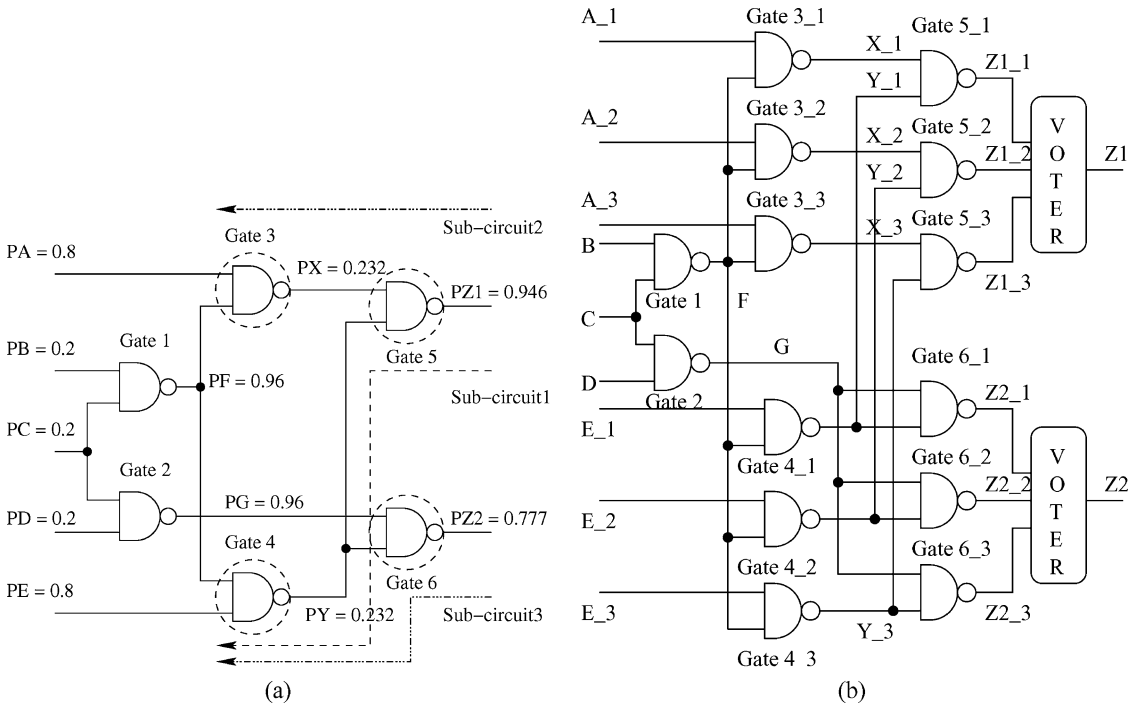

Fig. 8. (a) C17 circuit. (b) STMR version of C17 circuit.

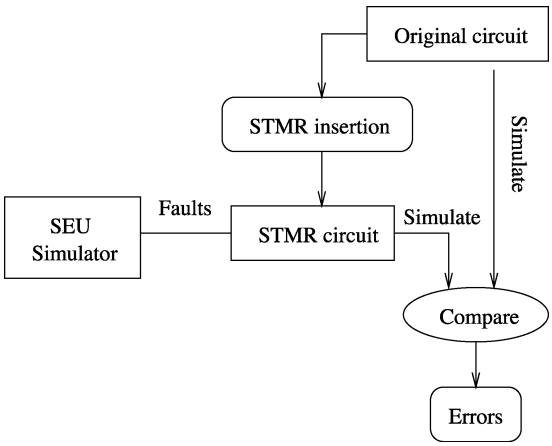

Fig. 9. Comparision of original and faulted circuits.

```
1 function resolve_std_logic (values: in std_logic_vector)
2   return std_logic is
3   variable result: std_logic;
4   begin
5     result := not values(0);
6               for index in values'range loop
7                 if values(0) = 'Z' then
8                     result := values(1);
9                 elsif values(1) = 'Z' then
10                    result := values(0);
11                end if;
12              end loop;
13  return result;
14  end resolve_std_logic;
```

Fig. 10. VHDL resolution function to resolve between values of two drivers on a signal.

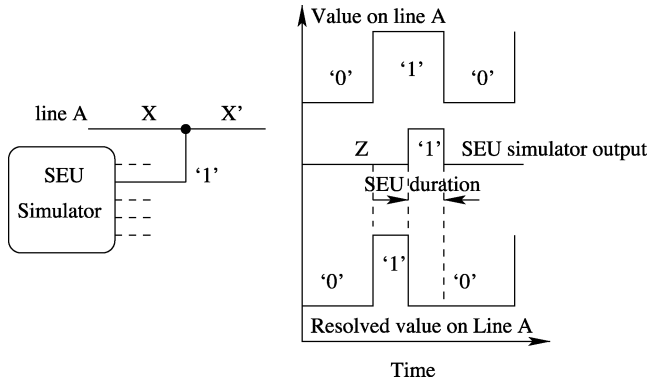

Fig. 11. Fault insertion on line "A" by SEU simulator.

We have implemented the proposed technique and validated it by simulation on various benchmarks chosen from MCNC 91 Benchmark Suite [3]. To start with, each benchmark is synthesized using SIS [4] Logic Synthesis tool. Then, we propagate the input probabilities and generate a STMR circuit, map it onto Xilinx Virtex FPGA, extract structural VHDL netlist, and insert faults randomly on the circuit. Compared with the standalone circuit (i.e., without any hardening) the STMR circuit has high level of immunity against SEUs.

The rest of the paper is organized as follows: Section II presents in detail the proposed selective TMR technique as an algorithm and illustrates with a small example. Section III explains the experimental setup, reports the experimental results, and finally analyzes them. It also briefly discusses an extension of the proposed technique to sequential circuits. Finally, Section IV draws conclusions.

II. STMR Approach

We present in detail the proposed STMR method. The overall idea is as follows: Given the primary input probabilities, we propagate them to primary outputs in one pass. In the next pass,

TABLE II
RESULTS BEFORE MAPPING, THRESHOLD PROBABILITY=0.3

Benchmark	3ns					5ns					20ns					Full TMR	
	Original		STMR		S	Original		STMR		S	Original		STMR		S		
	E	A	E	A	(%)	E	A	E	A	(%)	E	A	E	A	(%)	E	A
C1355	5	566	0	1508	11	14	566	0	1500	11	30	566	0	1506	11	0	1698
C499	18	206	3	398	35	27	206	7	398	35	91	206	20	398	35	0	618
C880	8	318	0	774	18	20	318	0	832	12	52	318	0	788	17	0	954
alu2	1	336	0	688	31	2	336	0	724	28	14	336	1	746	25	0	1008
alu4	1	676	0	1524	24	2	676	0	1380	31	2	676	0	1572	22	0	2028
c8	43	162	4	340	30	86	162	4	356	26	259	162	0	396	18	0	486
cc	12	56	0	148	11	7	56	4	90	46	43	56	0	140	16	0	168
cm138a	0	21	0	41	34	0	21	0	39	38	1	21	0	39	38	0	63
cm150a	0	58	0	60	65	1	58	0	108	37	9	58	0	168	3	0	174
cm151a	5	30	0	88	2	9	30	0	86	4	14	30	15	34	62	0	90
cm152a	39	23	0	69	0	37	23	6	43	37	166	23	0	69	0	0	69
cm162a	10	44	0	120	9	45	44	0	84	36	141	44	0	120	9	0	132
cm163a	3	43	0	119	7	39	43	0	117	9	66	43	0	87	32	0	129
cm42a	31	16	0	48	0	6	16	0	44	8	134	16	0	44	8	0	48
cm82a	69	18	1	44	18	136	18	0	52	3	359	18	5	50	7	0	54
comp	9	107	0	305	4	5	107	0	283	11	21	107	0	303	5	0	321
cordic	19	64	1	94	51	30	64	5	94	51	64	64	1	96	50	0	192
count	2	109	4	143	56	4	109	4	143	56	5	109	3	143	56	0	327
cu	2	54	3	90	44	8	54	0	96	40	26	54	3	100	38	0	162
frg1	5	100	1	112	62	5	100	2	112	62	17	100	0	156	48	0	300
lal	18	113	0	247	27	39	113	8	193	43	129	113	12	267	21	0	339
sct	29	98	0	208	29	49	98	3	246	16	159	98	2	266	9	0	294
tcon	78	33	0	97	2	112	33	3	91	8	210	33	0	95	4	0	99
term1	3	365	0	413	62	2	365	0	405	63	17	365	0	459	58	0	1095
ttt2	19	204	0	466	23	37	204	3	456	25	25	204	2	338	44	0	612
unreg	92	97	4	225	22	128	97	0	225	22	418	97	8	225	22	0	291
x1	15	307	1	553	39	20	307	3	509	44	81	307	1	461	49	0	921

E = Number of times the circuit is functionally upset with the induced SEU.
A = Area of the circuit (No. of gates).
S = Percentage savings in area overhead compared to full TMR.

we start from primary outputs and backtrack to determine SEU-sensitive gates. The notion of a SEU-sensitive gate is described in detail later. A subcircuit that consists of SEU-sensitive gates is identified to be SEU-sensitive. We introduce TMR for each SEU-sensitive subcircuit.

This section is organized as follows: Since the approach is exploiting the input environment information, first, we will discuss how to obtain such information. We will then describe the notion of a SEU-sensitive gate. Based on this notion, we will proceed to explain the STMR algorithm in detail. Finally, we will illustrate the idea by a small example.

A. Characterizing Input Environment

Typically, the user of an application will have some idea of the environment in which it will be employed. In case of space-based applications, for, e.g., weather forecasting satellite, we can characterize the input environment based on the image data that is captured.

Profiling has been a popular method for input characterization. Software profiling techniques are widely used in the software development to identify the often executed portions of the code. Representative benchmarks are used to gather profile data. In the past decade, profiling for hardware design has been extensively used to design low power systems [12]. The profiled data can be summarized either in the form of input signal probabilities or in terms of "representative" input sequence. In the latter case, vector-compaction [13], [14] based scheme has been proposed to reduce the length of such sequences. We can always reduce the representative sequence to input probabilities by simulating the circuit with the sequence. Thus, it is justified to assume that the input environment information is available in the form of input signal probabilities.

In the context of SEU-hardening synthesis technique, [25] proposed a method of calculating the probability of an upset due to a SET on a given combinational circuit. The probabilities are determined based on the radiation environment it will be subjected to and the nature of the circuit.

We will recall the concept of "sensitive" input of a gate introduced by the Critical Path Tracing (CPT) algorithm [27]. With respect to a test vector, a sensitive input is identified as follows:

Definition 1: A gate input is sensitive (in a test t) if complementing its value changes value of the gate output. The sensitive inputs of a gate with two or more inputs is determined as follows:

1) If only one input j has the dominant value of the gate, then j is sensitive.
2) If all inputs have non dominant values, then all inputs are sensitive.

Since, we have signal probabilities rather than actual test vectors, in order to use the above definition, we define a threshold probability as follows:

Definition 2: The logic value assumed by a line is "0" if its signal probability is less than the threshold probability $P_{\text{threshold}}$ otherwise it assumes a logic "1."

TABLE III
RESULTS BEFORE MAPPING, THRESHOLD PROBABILITY=0.4

Benchmark	3ns					5ns					20ns					Full TMR	
	Original		STMR		S	Original		STMR		S	Original		STMR		S		
	E	A	E	A	(%)	E	A	E	A	(%)	E	A	E	A	(%)	E	A
C1355	8	566	0	1126	33	7	566	1	1140	32	32	566	1	1138	32	0	1698
C499	17	206	4	398	35	29	206	8	398	35	84	206	24	398	35	0	618
C880	10	318	1	728	23	12	318	2	756	20	60	318	3	760	20	0	954
alu2	2	336	1	622	38	4	336	0	638	36	13	336	1	492	51	0	1008
alu4	1	676	0	1438	29	2	676	0	1216	40	2	676	0	1270	37	0	2028
c8	49	162	4	334	31	75	162	2	396	18	271	162	19	330	32	0	486
cc	15	56	0	148	11	16	56	11	92	45	89	56	4	140	16	0	168
cm138a	0	21	1	37	41	0	21	0	41	34	1	21	0	41	34	0	63
cm150a	1	58	2	60	65	1	58	0	122	29	3	58	7	60	65	0	174
cm151a	7	30	0	64	28	3	30	0	50	44	10	30	7	34	62	0	90
cm152a	28	23	8	43	37	71	23	8	45	34	155	23	21	43	37	0	69
cm162a	18	44	0	80	39	14	44	0	126	4	63	44	0	92	30	0	132
cm163a	28	43	0	79	38	50	43	0	87	32	116	43	10	75	41	0	129
cm42a	8	16	0	48	0	1	16	0	48	0	35	16	0	44	8	0	48
cm82a	56	18	0	52	3	157	18	0	52	3	387	18	6	50	7	0	54
comp	12	107	0	179	44	4	107	1	239	25	20	107	0	275	14	0	321
cordic	16	64	0	94	51	36	64	4	96	50	116	64	7	110	42	0	192
count	3	109	0	313	4	4	109	6	143	56	31	109	22	143	56	0	327
cu	14	54	4	98	39	11	54	3	90	44	52	54	4	94	41	0	162
frg1	0	100	1	114	62	5	100	5	114	62	39	100	32	112	62	0	300
lal	13	113	7	165	51	37	113	10	169	50	66	113	26	163	51	0	339
sct	52	98	3	212	27	63	98	1	206	29	179	98	5	220	25	0	294
tcon	58	33	1	91	8	79	33	1	93	6	240	33	5	81	18	0	99
term1	3	365	0	407	62	6	365	1	437	60	18	365	1	415	62	0	1095
ttt2	17	204	2	296	51	42	204	3	442	27	94	204	24	274	55	0	612
unreg	81	97	5	193	33	194	97	3	277	4	490	97	0	289	0	0	291
x1	12	307	2	433	52	19	307	2	473	48	88	307	5	429	53	0	921

E = Number of times the circuit is functionally upset with the induced SEU.
A = Area of the circuit (No. of gates).
S = Percentage savings in area overhead compared to full TMR.

Thus, given a $P_{\text{threshold}}$, we would first assign the logic values, according to the Definition 2, for the inputs of a gate and then we determine the gate's sensitivity according to Definition 1 and Definition 3.

Definition 3: If a gate has one or more sensitive inputs, then we say that the gate is sensitive to SEUs.

The signal probability of the output (P_{out}) of an n-input gate with ith input having P_i signal probability, is determined by the type of gate as shown in Table I.

The **Algorithm is_sensitive**() shown in Fig. 3 is employed to find the sensitivity of a gate. It can be illustrated by an example: Consider a 3-input AND gate with the signal probabilities of the inputs A, B, and C equal to 0.4, 0.6, and 0.8, respectively, as shown in the Fig. 2. Let the threshold probability be 0.5. The lines 5–11 of the algorithm (Fig. 3) assign the dominant value of the input gate depending on its type.

Assume a fault due to SEU on one of the inputs "A" at some instant of time, and assume that all other signals are at logic "1" at that instant. The fault propagates through the gate because all other signals are at nondominant values. In other words, a fault on the input A propagates to the output of the gate only when the the other inputs assume nondominant values. Interpreting this in terms of probabilities: an SEU on one of the inputs of a gate has a higher probability of upsetting its output only if the signal probability of all other inputs being at nondominant value is greater than or equal to the threshold probability. Hence, the gate is assumed to be sensitive to SEUs on its inputs. Consider the 3-input gate with a different set of input probabilities, A (0.4), B (0.4), and C (0.8). The fault on line A has lesser probability of propagating through the gate as the probability of line B assuming nondominant value is less than the threshold probability, consequently making the gate insensitive to SEUs on it inputs. The lines 12–27 of the algorithm shown in Fig. 3 performs the function described above.

B. Proposed STMR Algorithm

The algorithm for synthesizing the hardened circuit from a given circuit is as outlined in Fig. 4. Given the input signal probabilities of a gate-level netlist, first, the algorithm determines the signal probabilities of all the nets in the circuit (lines 3–9). The signal probabilities of the primary inputs are propagated, level by level until the primary outputs of the circuit are reached. SEU sensitive subcircuits are identified as in lines 10–16. A subcircuit is marked as sensitive if an SEU in it has a higher probability of affecting one or more primary outputs of the circuit. Such subcircuits are identified by starting at the primary outputs and backtracking through the circuit and finding the longest cascaded chain of sensitive gates.

The algorithm for identifying a SEU sensitive subcircuit is shown in Fig. 5. The algorithm starts at one of the gates at the last level of the circuit (line 4). All the SEU sensitive gates connected to this gate are found by backtracking recursively as in the lines 6–11. The sensitivity of the gate is determined by using the algorithm shown in Fig. 3.

TABLE IV
RESULTS BEFORE MAPPING, THRESHOLD PROBABILITY=0.5

Benchmark	3ns					5ns					20ns					Full TMR	
	Original		STMR		S	Original		STMR		S	Original		STMR		S		
	E	A	E	A	(%)	E	A	E	A	(%)	E	A	E	A	(%)	E	A
C1355	10	566	0	1092	35	5	566	4	1086	36	29	566	3	1080	36	0	1698
C499	16	206	2	398	35	28	206	5	398	35	88	206	22	398	35	0	618
C880	15	318	0	668	29	18	318	0	716	24	31	318	1	700	26	0	954
alu2	2	336	2	582	42	1	336	3	556	44	18	336	5	492	51	0	1008
alu4	1	676	0	1174	42	2	676	0	1198	40	5	676	0	756	62	0	2028
c8	52	162	6	308	36	99	162	9	328	32	200	162	28	316	34	0	486
cc	3	56	3	86	48	31	56	0	134	20	109	56	5	140	16	0	168
cm138a	0	21	1	37	41	0	21	0	39	38	22	21	9	37	41	0	63
cm150a	1	58	2	60	65	1	58	1	60	65	3	58	6	60	65	0	174
cm151a	3	30	0	66	26	2	30	0	34	62	0	30	0	34	62	0	90
cm152a	32	23	5	41	40	72	23	6	37	46	188	23	26	43	37	0	69
cm162a	20	44	0	80	39	25	44	0	88	33	82	44	0	118	10	0	132
cm163a	15	43	0	109	15	68	43	2	73	43	53	43	7	111	13	0	129
cm42a	20	16	0	44	8	48	16	0	46	4	92	16	0	44	8	0	48
cm82a	77	18	1	50	7	135	18	1	50	7	377	18	38	40	25	0	54
comp	3	107	0	179	44	7	107	0	235	26	24	107	0	171	46	0	321
cordic	26	64	2	94	51	34	64	3	114	40	121	64	0	110	42	0	192
count	17	109	13	143	56	6	109	11	143	56	180	109	0	207	36	0	327
cu	5	54	2	80	50	5	54	7	82	49	63	54	11	82	49	0	162
frg1	2	100	0	114	62	11	100	7	112	62	51	100	34	112	62	0	300
lal	9	113	4	179	47	33	113	9	173	48	77	113	39	173	48	0	339
sct	42	98	5	188	36	60	98	12	158	46	204	98	35	192	34	0	294
tcon	53	33	4	83	16	55	33	10	83	16	294	33	20	81	18	0	99
term1	3	365	0	405	63	5	365	1	411	62	12	365	6	411	62	0	1095
ttt2	15	204	4	312	49	24	204	9	288	52	97	204	22	308	49	0	612
unreg	82	97	8	193	33	132	97	20	193	33	358	97	9	193	33	0	291
x1	11	307	2	407	55	24	307	0	427	53	80	307	3	435	52	0	921

E = Number of times the circuit is functionally upset with the induced SEU.
A = Area of the circuit (No. of gates).
S = Percentage savings in area overhead compared to full TMR.

The nondominant value for AND and NAND gate is "1," hence, their sensitivity depends on the same criterion. The sensitivity of OR and NOR gates also depends on a common criterion. EXOR, EXNOR, and NOT gates propagate faults no matter what the signal probabilities of the inputs are, so these gates are always considered SEU sensitive. The output gates are also assumed to be sensitive as a heavy ion bombarding the gate might affect the final output.

The circuit can be immunized against upsets by mitigating SEUs in the sensitive subcircuits. This can be accomplished by applying TMR for all the gates in such subcircuits (lines 17–19 of Fig. 4). The algorithm employed for selective TMR insertion is given in Fig. 6. The lines 4–15 of the TMR-insertion algorithm triplicates all the gates of subcircuits that are identified as SEU sensitive. A voter is introduced between gates depending on the fanout connections of the sensitive gates as in the lines 16–31.

If the fanout of a sensitive gate is connected to only sensitive gates, then the outputs of the triplicates can be directly connected to the inputs of the triplicates of the next level. This implies that the introduction of a voter between such levels is not necessary.

For example, consider two sensitive gates, Gate1 and Gate2 (marked by dotted circles) connected as shown in Fig. 7. The output of the SEU sensitive gate Gate1, D is connected only to Gate2 which is also sensitive. Hence, the triplicated structure for this subcircuit is as shown in Fig. 7(a).

If the fanout of the sensitive gate is connected to a non triplicated gate, then a voter is introduced between them. The mitigated output is then fed to the non triplicated gate. This is illustrated in Fig. 7(b). The output of Gate1, D, is connected to a SEU-sensitive gate (Gate2) and non sensitive gate (Gate3). Hence, the outputs of the triplicated structure D_1, D_2, and D_3 have to be mitigated using a voter before it is fed to the gate Gate3. It is assumed that the Gate3 is not in the last level of the circuit.

C. An Illustrative Example

Consider the (MCNC'91 benchmark) C17 circuit, as shown in Fig. 8. The signal probabilities of the inputs are marked beside them.

The signal probabilities of the nets are calculated as in the lines 3–9 of Fig. 4. Gates 3, 4, 5, and 6 are found to be SEU sensitive [shown by dotted circles in Fig. 8(a)] as in lines 10–16 of the algorithm. Gate 4 is SEU sensitive as a fault on line F or line E has a high probability of affecting its output Y. Similarly, Gate 3 is sensitive as a SEU on input A (or input F) has a high probability of affecting its output X; an SEU on line Y has a high probability of affecting the signal Z2 which is the output of gate 6, hence it is considered sensitive. Although Gate 5 has no sensitive input, it is considered SEU sensitive as it is in the last level of the circuit.

SEU sensitive subcircuits can be obtained by starting at one of the outputs and backtracking through the continuous chain

TABLE V
RESULTS AFTER MAPPING, THRESHOLD PROBABILITY=0.3

Benchmark	3ns					5ns					20ns					Full TMR	
	Original		STMR		S	Original		STMR		S	Original		STMR		S		
	E	L	E	L	(%)	E	L	E	L	(%)	E	L	E	L	(%)	E	L
C1355	106	82	10	317	-28	179	82	13	317	-28	478	82	53	334	-35	0	246
C499	92	82	31	202	9	143	82	57	202	9	496	82	204	250	-12	0	222
C880	95	132	13	344	12	158	132	15	357	9	503	132	52	359	8	0	393
alu2	52	157	2	349	26	77	157	4	370	22	157	157	14	383	19	0	477
alu4	49	285	6	698	16	80	285	7	629	24	220	285	13	731	12	0	834
c8	79	54	27	115	29	167	54	19	124	23	420	54	81	146	9	0	162
cc	94	26	1	81	-3	97	26	12	55	29	404	26	12	71	8	0	78
cm138a	138	9	2	27	0	181	9	5	27	0	330	9	24	35	-29	0	27
cm150a	14	21	2	22	63	62	21	9	44	26	169	21	7	66	-10	0	60
cm151a	49	12	9	33	0	96	12	8	34	-3	185	12	114	14	57	0	33
cm152a	66	6	3	18	0	114	6	33	12	33	293	6	38	18	0	0	18
cm162a	148	15	6	40	11	182	15	36	32	28	510	15	31	43	4	0	45
cm163a	152	14	13	40	4	202	14	13	38	9	561	14	198	30	28	0	42
cm42a	199	13	12	30	23	272	13	15	30	23	888	13	18	39	0	0	39
cm82a	199	4	28	12	0	333	4	45	13	-8	1000	4	202	14	-16	0	12
comp	31	47	0	137	2	52	47	0	131	7	129	47	3	135	4	0	141
cordic	25	21	4	47	25	31	21	3	47	25	103	21	4	53	15	0	63
count	86	46	10	75	45	123	46	8	75	45	349	46	20	97	29	0	138
cu	63	26	5	59	24	117	26	7	49	37	308	26	66	59	24	0	78
frg1	15	43	6	49	62	27	43	14	49	62	76	43	43	70	45	0	129
lal	63	59	16	145	18	91	59	5	123	30	289	59	8	162	8	0	177
sct	64	45	3	107	20	142	45	11	124	8	422	45	7	130	3	0	135
tcon	103	8	20	24	0	170	8	51	24	0	530	8	132	26	-8	0	24
term1	15	160	5	187	61	11	160	18	183	61	76	160	19	193	59	0	480
ttt2	49	95	5	233	18	112	95	4	231	18	242	95	8	180	36	0	285
unreg	99	32	5	96	0	156	32	4	96	0	453	32	47	128	-33	0	96
x1	51	152	10	291	36	77	152	15	268	41	228	152	27	260	42	0	456

E = Number of times the circuit is functionally upset with the induced SEU.
A = Area of the circuit (No. of gates).
S = Percentage savings in area overhead compared to full TMR.

of sensitive gates. For example, the subcircuit1 can be obtained by starting at the primary output Z1. Backtracking from Gate 5, we find that Gate 3 and Gate 4 as sensitive gates connected to Gate 5. The algorithm now backtracks recursively through Gate 3 and Gate 4 in two passes. Assume that the algorithm proceeds through Gate 4. Backtracking from Gate 4 we find no sensitive gates. Hence, we stop at Gate 4 and mark gates 4 and 5 as the sensitive gates in subcircuit1. Similarly, subcircuit2 and subcircuit3 shown in Fig. 8(a) can be obtained.

TMR is now applied selectively on the subcircuits to harden the circuit against SEUs. The resulting STMR circuit, is as shown in Fig. 8(b), with all the gates in the sensitive subcircuits replaced with their triplicates. The hardened circuit has two voters introduced at the primary outputs. The voter can be implemented using either LUTs or tristate buffers. However, they are implemented using tristate buffers as they are resistant to SEUs.

It is evident from Fig. 8(b) that the SEU hardened STMR circuit has a total of 14 gates. Whereas the same circuit when hardened by full module TMR has 18 gates ($6\ \text{gates} \times 3 = 18\ \text{gates}$). Hence, there is a savings of 4 gates for the given set of input signal probabilities for the C17 benchmark circuit. However, the STMR circuit has an overhead of one voter circuit when compared to the TMR method. But the voter circuit is not considered as an overhead, as Virtex has abundant number of tristate buffers which usually go unused [6]. Hence, they can be employed for implementing tristate voters.

III. EXPERIMENTAL SETUP AND RESULTS

We first elaborate the experimental flow used for validating the proposed STMR method. Then, we discuss the SEU simulator we have developed to insert faults representing SEUs. We also discuss the functional testing procedure employed for assessing the SEU immunity of the STMR circuit. And, last, we analyze the results obtained by applying selective TMR technique on benchmarks.

A. Experimental Flow

The experimental flow involves the following four tasks:

1) Preparing the Input file: The STMR mitigation is tested on the combinational circuits of the MCNC 91 benchmark suite. The netlists which are in BLIF (Berkeley Logic Interchange Format) are converted into VHDL format. This ensures that the generated VHDL file could be fed into the Xilinx Foundation Tools 4.1i to map the designs onto Virtex FPGAs.
2) STMR insertion: The STMR algorithm discussed in the previous chapter is coded in "C" language. The generated VHDL netlist is fed into the STMR algorithm. A random set of probabilities is generated and assigned to the inputs of the given circuit. The probabilities are then propagated through the circuit. SEU sensitive subcircuits are identified and structural modifications are made to the orig-

TABLE VI
RESULTS AFTER MAPPING, THRESHOLD PROBABILITY=0.4

Benchmark	3ns					5ns					20ns					Full TMR	
	Original		STMR		S	Original		STMR		S	Original		STMR		S		
	E	L	E	L	(%)	E	L	E	L	(%)	E	L	E	L	(%)	E	L
C1355	105	82	12	319	-29	170	82	22	301	-22	464	82	63	298	-21	0	246
C499	98	82	32	202	9	197	82	53	202	9	524	82	158	202	9	0	222
C880	96	132	6	323	17	197	132	28	343	12	529	132	60	348	11	0	393
alu2	40	157	5	332	30	75	157	14	340	28	137	157	48	257	46	0	477
alu4	50	285	4	676	18	76	285	9	564	32	265	285	40	595	28	0	834
c8	95	54	14	121	25	133	54	29	134	17	492	54	64	116	28	0	162
cc	111	26	5	81	-3	138	26	35	55	29	605	26	58	71	8	0	78
cm138a	137	9	8	25	7	156	9	2	27	0	395	9	25	27	0	0	27
cm150a	21	21	10	22	63	47	21	10	48	20	102	21	62	22	63	0	60
cm151a	59	12	10	27	18	92	12	44	20	39	160	12	96	13	60	0	33
cm152a	67	6	16	12	33	88	6	35	12	33	278	6	101	12	33	0	18
cm162a	102	15	30	32	28	192	15	17	44	2	612	15	163	32	28	0	45
cm163a	106	14	30	30	28	170	14	39	30	28	443	14	122	28	33	0	42
cm42a	199	13	11	30	23	324	13	22	30	23	915	13	39	30	23	0	39
cm82a	199	4	27	13	-8	333	4	76	12	0	986	4	190	13	-8	0	12
comp	25	47	2	100	29	38	47	6	121	14	138	47	1	133	5	0	141
cordic	18	21	1	47	25	56	21	2	49	22	220	21	17	56	11	0	63
count	122	46	10	140	-1	133	46	6	75	45	383	46	43	75	45	0	138
cu	80	26	10	52	33	124	26	11	52	33	322	26	34	49	37	0	78
frg1	16	43	9	49	62	25	43	18	49	62	71	43	45	49	62	0	129
lal	49	59	13	103	41	103	59	48	107	39	227	59	48	103	41	0	177
sct	91	45	18	113	16	122	45	8	108	20	393	45	71	110	18	0	135
tcon	100	8	25	24	0	167	8	47	24	0	506	8	145	24	0	0	24
term1	7	160	3	186	61	23	160	10	197	58	52	160	14	187	61	0	480
ttt2	42	95	10	161	43	133	95	6	224	21	294	95	91	147	48	0	285
unreg	93	32	27	64	33	199	32	5	96	0	555	32	39	96	0	0	96
x1	57	152	16	236	48	74	152	7	263	42	236	152	47	236	48	0	456

E = Number of times the circuit is functionally upset with the induced SEU.
A = Area of the circuit (No. of gates).
S = Percentage savings in area overhead compared to full TMR.

inal circuit by applying STMR. The VHDL netlist of the STMR circuit is then fed to SEU simulator. The results so obtained represent the behavior of the circuit before mapping.

The STMR circuit is mapped onto Virtex FPGA using Xilinx Foundation Tool 4.1i. The mapped netlist is again fed to the SEU simulator and tested for SEU immunity. The results so obtained represent the behavior of the circuit after mapping.

3) SEU Simulation: A SEU simulator is designed to create a realistic scenario of the faults injected into the space electronics due to SEUs. The simulator is explained in detail later (Section III-B).
4) Error Calculation: Fig. 9 shows the process of fault insertion and testing. The STMR circuit is faulted by introducing SEUs using the simulator and simulated. The functional operation of the STMR circuit is compared against that of the original unfaulted circuit. This is done by EXOR-ing the outputs of both the circuits. A disparity between these outputs indicate that the SEU induced in the STMR circuit has propagated to its output(s), thus leading to a functional failure. The number of errors are thus calculated. This process is repeated with the STMR netlist obtained both before mapping and after mapping onto Virtex.

B. SEU Simulator

The SEU simulator designed for the purpose of fault (SEU) injection has the following three important features.

1) An SEU can occur on any line of the circuit thus injecting a fault. The simulator subjects the circuit to this condition by randomly injecting a fault on any one signal.
2) When an SEU occurs at any node, it temporarily inverts the value on that line. The simulator allows the variation of SEU duration. The duration of SEU represents the period of fault injection due to an SEU. A bit-flip in an SRAM cell is simulated by introducing a fault for the entire input duration (in our simulations each input vector is applied for 20 ns). On the other hand, the SEUs on the interconnection and combinational logic are introduced by flipping the logic value on the circuit line temporarily for a short duration.
3) An SEU can occur during the input transitions or at any instance during the application of inputs. The simulator introduces faults on a line randomly in time.

The SEU simulator operates as follows: Each output of the fault generator is assigned to force either logic "Z" or logic "1" on one of the lines of the circuit. We assume that all signals except the primary inputs of the circuits and the primary ouputs of the circuits (coming out of voter circuits for STMR circuits) are

TABLE VII
RESULTS AFTER MAPPING, THRESHOLD PROBABILITY=0.5

Benchmark	3ns					5ns					20ns					Full TMR	
	Original		STMR		S	Original		STMR		S	Original		STMR		S		
	E	L	E	L	(%)	E	L	E	L	(%)	E	L	E	L	(%)	E	L
C1355	104	82	18	345	-40	170	82	20	322	-30	331	82	44	367	-49	0	246
C499	95	82	18	202	9	143	82	35	202	9	314	82	73	202	9	0	222
C880	92	132	18	311	20	190	132	24	330	16	420	132	55	321	18	0	393
alu2	53	157	20	296	37	49	157	19	295	38	103	157	39	255	46	0	477
alu4	74	285	13	561	32	72	285	18	585	29	157	285	90	330	60	0	834
c8	99	54	13	120	25	162	54	22	119	26	299	54	89	115	29	0	162
cc	64	26	14	55	29	189	26	20	71	8	404	26	52	76	2	0	78
cm138a	132	9	9	25	7	164	9	7	27	0	314	9	42	25	7	0	27
cm150a	23	21	14	22	63	43	21	23	22	63	77	21	53	22	63	0	60
cm151a	50	12	9	27	18	41	12	35	13	60	79	12	51	13	60	0	33
cm152a	72	6	18	12	33	107	6	74	10	44	205	6	54	12	33	0	18
cm162a	98	15	20	32	28	184	15	50	30	33	450	15	41	44	2	0	45
cm163a	124	14	6	44	-4	155	14	27	28	33	435	14	53	43	-2	0	42
cm42a	199	13	16	30	23	315	13	26	30	23	609	13	36	30	23	0	39
cm82a	199	4	27	13	-8	333	4	51	13	-8	667	4	79	18	-50	0	12
comp	30	47	6	100	29	57	47	17	117	17	89	47	27	94	33	0	141
cordic	31	21	10	47	25	72	21	6	60	4	118	21	11	60	4	0	63
count	99	46	21	75	45	141	46	31	75	45	337	46	58	131	5	0	138
cu	69	26	10	51	34	114	26	19	51	34	201	26	16	49	37	0	78
frg1	13	43	2	49	62	31	43	17	49	62	54	43	39	49	62	0	129
lal	45	59	4	110	37	100	59	20	109	38	162	59	43	109	38	0	177
sct	95	45	17	99	26	126	45	24	83	38	282	45	52	97	28	0	135
tcon	98	8	24	24	0	173	8	46	24	0	328	8	89	24	0	0	24
term1	12	160	8	189	60	26	160	8	197	58	53	160	16	198	58	0	480
ttt2	53	95	8	172	39	69	95	16	158	44	176	95	32	169	40	0	285
unreg	89	32	27	64	33	161	32	37	64	33	251	32	92	64	33	0	96
x1	43	152	13	229	49	68	152	20	233	48	155	152	32	242	46	0	456

E = Number of times the circuit is functionally upset with the induced SEU.
A = Area of the circuit (No. of gates).
S = Percentage savings in area overhead compared to full TMR.

sensitive to SEUs. So the simulator produces as many outputs as there are signals in the circuit. Hence, at any point of time the nets to be upsetted are each driven by two sources, the original value and the simulator output.

Consider an example, where line A has to be induced with an SEU, during simulation. It is assigned to one of the outputs of the SEU simulator. When the simulator forces "Z" on the signal, the resolution function shown in Fig. 10 resolves the effective value on the Line A to be its original value as shown in Fig. 11. But, when there is a logic "1" on the simulator output driving Line A the function resolves the line value to be the inverted value of the original, during simulation as shown in Fig. 11.

C. Results

STMR technique was tested on various benchmark circuits. The synthesized STMR circuits were introduced with SEUs and simulated. The same STMR circuits were then mapped onto Virtex FPGAs using Xilinx Foundation Tool. The mapped netlists were extracted, introduced with SEUs and simulated.

Tables II, III, and IV show the results obtained before mapping for three sets of threshold probabilities (0.3, 0.4, and 0.5, respectively). Similarly, Tables V, VI, and VII show the results after mapping. The circuits are tested with three SEU durations (3, 5, and 20 ns) for each set of threshold probabilities shown at the top of each table. The columns corresponding to "original" represent the statistics of the original circuits. Whereas that marked by "STMR" are that of the synthesized STMR circuits. The column "S" show the area savings of the STMR circuit over the TMR design of the same circuit.

The columns marked as "E" denote the number of times an induced SEU affected the correct operation of the circuit. The column "A" denotes the area of the circuit (in terms of gates) before mapping. The column marked as "L" in the results after mapping represents the resources in terms of LUTs used for mapping the designs onto Virtex.

For each set of threshold probability the original and STMR circuits are simulated with the same set of 1000 test vectors. The input test vectors randomly generated adhere to the appropriate probabilities of the inputs that were employed in generating the corresponding STMR circuits. The duration of each input is 20 ns. Hence, an SEU duration of 20 ns represents the faults due to SEUs in the SRAM memory cells. The designs are induced with 1000 SEUs, one per each test vector. This is equivalent to simulating the circuit in actual radiation environment for a period of 1000 days. This is assumed based on the empirical data, according to which there are $10^{-10}\ \mathrm{bit-upsets/d}$ [7], which can be approximated to one SEU per day in the electronics. And the duration of an SEU upsetting the device is less than 200 ps [28].

The efficiency of the STMR method in decreasing the area of the STMR circuit is a factor of: 1) The nature of the combinational circuit; 2) The input signal probabilities; and 3) The

TABLE VIII
SEU SENSITIVITY OF CIRCUITS *BEFORE* MAPPING

Benchmark	$P_{threshold}$ = 0.3			$P_{threshold}$ = 0.4			$P_{threshold}$ = 0.5		
	3ns	5ns	20ns	3ns	5ns	20ns	3ns	5ns	20ns
C1355	100	100	100	100	85	96	100	20	89
C499	83	74	78	76	72	71	87	82	75
C880	100	100	100	90	83	95	100	100	96
alu2	100	100	92	50	100	92	0	-200	72
alu4	100	100	100	100	100	100	100	100	100
c8	90	95	100	91	97	92	88	90	86
cc	100	42	100	100	31	95	0	100	95
cm138a	0	0	100	0	0	100	0	0	59
cm150a	0	100	100	-100	100	-133	-100	0	-100
cm151a	100	100	-7	100	100	30	100	100	0
cm152a	100	83	100	71	88	86	84	91	86
cm162a	100	100	100	100	100	100	100	100	100
cm163a	100	100	100	100	100	91	100	97	86
cm42a	100	100	100	100	100	100	100	100	100
cm82a	98	100	98	100	100	98	98	99	89
comp	100	100	100	100	75	100	100	100	100
cordic	94	83	98	100	88	93	92	91	100
count	-100	0	40	100	-50	29	23	-83	100
cu	-50	100	88	71	72	92	60	-40	82
frg1	80	60	100	0	0	17	100	36	33
lal	100	79	90	46	72	60	55	72	49
sct	100	93	98	94	98	97	88	80	82
tcon	100	97	100	98	98	97	92	81	93
term1	100	100	100	100	83	94	100	80	50
ttt2	100	91	92	88	92	74	73	62	77
unreg	95	100	98	93	98	100	90	84	97
x1	93	85	98	83	89	94	81	100	96

Each column corresponds to the sensitivity of the STMR circuit for the given pulse width of the SEU.

number of gates in the last level of the combinational circuit. For example the more the number of EXOR, EXNOR, and NOT gates, greater the area of the STMR circuit. This is because, as mentioned before, these gates are always sensitive to SEUs no matter what the input signal probabilities are. Also, since the gates in the last level of the circuit are also considered sensitive independent of the signal probabilities of their inputs; the area of the STMR circuit is highly dependent on the number of gates in the last level of the original circuit.

It should be noted that the concept of $P_{\text{threshold}}$ has been adopted only to assign logic values to the lines. The performance of the STMR circuits is not a function of the $P_{\text{threshold}}$. However, if the circuit consists of only AND and/or NAND gates the number of functional errors decrease and the area of the STMR circuits increase with decrease in the threshold probability. This is because as the threshold probability is decreased, more number of lines are assumed to be at logic "1" and hence more number of gates are marked sensitive. This leads to less number of errors to propogate to the output(s).

If the given circuit is made of only OR and/or NOR gates, any decrease in $P_{\text{threshold}}$ will decrease the number of sensitive gates and hence lead to lesser area of the STMR circuit. This will consequently increase the number of functional errors in the STMR circuit.

For a circuit consisting of AND, NAND, OR, and NOR gates, a change in $P_{\text{threshold}}$ may not necessarily signify a decrease or increase in area. Hence, a given circuit with a given set of input probabilities should be synthesized and simulated to get the best STMR circuit that satifies the required area and error constraints.

As seen from Tables II–VII, the area of the STMR design is significantly less than that required for full module TMR of the same design. It can be noticed from the Table II (results before mapping and $P_{\text{threshold}} = 0.3$) that the maximum savings in area is obtained for the benchmark circuit cm150a, which is 65%. As mentioned before the statistics of the STMR circuit is a function of input signal probabilities. This is evident from the results of the circuit cm42a with a different set of input probabilities (and threshold probability of 0.4) as shown in Table III. The circuit now has a savings of 8% (for SEU duration 20 ns).

Comparing results before and after FPGA mapping, we observe that in some cases the area savings *after* mapping is very less and for some other circuits it is more than that of the full TMR circuits. This is because of the voter circuits and the level of area optimization performed by the Xilinx Foundation Tool during the mapping of STMR circuits onto the Virtex FPGAs. For example the area savings of the cm42a circuit after mapping is 23% (Table V) and whereas that before mapping is 0%. However, the area savings of the C1355 STMR circuit before mapping is 11% and that after mapping is -28% (minus sign in the area savings indicates that the area of the STMR circuit is more than that of full TMR circuit).

Table VIII and Table IX show the SEU sensitivity of the benchmark circuits before mapping and after mapping repec-

TABLE IX
SEU SENSITIVITY OF CIRCUITS *AFTER* MAPPING

Benchmark	$P_{threshold}$ = 0.3			$P_{threshold}$ = 0.4			$P_{threshold}$ = 0.5		
	3ns	5ns	20ns	3ns	5ns	20ns	3ns	5ns	20ns
C1355	90	92	88	88	87	86	82	88	86
C499	66	60	58	67	73	69	81	75	76
C880	86	90	89	93	85	88	80	87	86
alu2	96	94	91	87	81	64	62	61	62
alu4	87	91	94	92	88	84	82	75	42
c8	65	88	80	85	78	86	86	86	70
cc	98	87	97	95	74	90	78	89	87
cm138a	98	97	92	94	98	93	93	95	86
cm150a	85	85	95	52	78	39	39	46	31
cm151a	81	91	38	83	52	40	82	14	35
cm152a	95	71	87	76	60	63	75	30	73
cm162a	95	80	93	70	91	73	79	72	90
cm163a	91	93	64	71	77	72	95	82	87
cm42a	93	94	97	94	93	95	91	91	94
cm82a	85	86	79	86	77	80	86	84	88
comp	100	100	97	92	84	99	80	70	69
cordic	84	90	96	94	96	92	67	91	90
count	88	93	94	91	95	88	78	78	82
cu	92	94	78	87	91	89	85	83	92
frg1	60	48	43	43	28	36	84	45	27
lal	74	94	97	73	53	78	91	80	73
sct	95	92	98	80	93	81	82	80	81
tcon	80	70	75	75	71	71	75	73	72
term1	66	-63	75	57	56	73	33	69	69
ttt2	89	96	96	76	95	69	84	76	81
unreg	94	97	89	70	97	92	69	77	63
x1	80	80	88	71	90	80	69	70	79

Each column corresponds to the sensitivity of the STMR circuit for the given pulse width of the SEU.

tively. The SEU sensitivity of a given circuit is calculated by dividing the difference between the number of errors in the original circuit and those in the STMR circuit by the number of errors due to SEUs in the original circuit. In other words, it is the percentage of the SEUs the STMR circuit has withstood. It can be inferred from the tables that the SEU sensitivity of the STMR circuits is excellent in many cases. However, for some of the circuits such as *alu2*, the sensitivity is negative. It indicates that the STMR version of *alu2* is more prone to SEUs than its original circuit.

It has been inferred from the experimental analysis that the number of SEUs affecting the smooth operation of the STMR circuit is typically less than 20. Hence, it can be assumed that over a period of 1000 d (roughly 3 yr), the STMR circuit mapped on FPGA malfunctions 20 times when it is hit with an SEU inducing particle.

The main advantage of TMR over STMR is that it guarantees 100% immunity of the circuits against SEUs. Whereas STMR circuit is prone to propagate some errors. However, readback and reconfiguration on STMR circuit can guarantee almost 100% immunity of the circuit against SEUs. The main advantage of the STMR method over TMR is that the area of the STMR circuit is roughly two-thirds of the area of the TMR circuit. The number of voter circuits required for the STMR circuit is very high when compared to the TMR circuit. The STMR method can be used on a device which has abundant tristate buffers such as Xilinx Virtex.

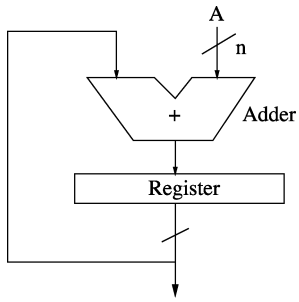

Fig. 12. Accumulator.

D. Extensions to Sequential Circuits

The STMR method discussed in this paper addresses the hardening technique for only combinational circuits. The method can be extended to harden the sequential circuits as follows. Assuming the sequential circuit is modeled as a synchronous state machine model, i.e., a combinational circuit with a feedback path consisting of state registers, the combinational block can be hardened against SEUs by applying STMR method. The state registers can be replaced with any SEU hardened latches reported in the literature [8]–[10]. Instead of hardened latches, we may TMR the state registers. For example, consider the accumulator unit as shown in the Fig. 12. Knowing the bit level probabilities of the input signal A, we can STMR the adder.

IV. Conclusion

We conclude from this paper that the proposed STMR is an effective technique for SEU hardening in FPGAs. The effectiveness of the proposed method is highly dependent on the input signal probabilities and the nature of the circuit. The area of the STMR circuit in the worst case can be equal to that of the full TMR circuit. STMR along with other mitigation features of the Virtex series can provide immunity against SEUs comparable to that with full module TMR, with less area overhead.

STMR technique is beneficial to those circuits with input environments wherein the size of the SEU sensitive subcircuit(s) is much smaller than the original circuit. For such circuits, the area overhead of STMR technique will be lesser than (upto 60–70% as observed in some of the examples) that of the TMR.

Acknowledgment

The authors would like to thank the anonymous reviewers for their invaluable feedback.

References

[1] SEU mitigation techniques for Virtex FPGA's in space applications, P. Blain, C. Carmichael, E. Fuller, and M. Caffrey. [Online]. Available: http://www.xilinx.com/appnotes/VtxSEU.pdf

[2] Q. Shi and G. K. Maki, "New design techniques for SEU immune circuits," in *9th NASA Symp. VLSI Design*, 2000, pp. 4.2.1–4.2.16.

[3] MCNC 91 Benchmarks [Online]. Available: http://www.cbl.ncsu.edu

[4] E. M. Sentovich *et al.*, SIS: A system for sequential circuit synthesis, May 1992.

[5] K. A. Label and M. M. Gates, "Single-event-effect mitigation from a system perspective," *IEEE Trans. Nucl. Sci.*, vol. 43, no. 2, 1996.

[6] C. Carmichael, Triple modular redundancy design techniques for Virtex FPGA's (DRAFT), 2001.

[7] P. Brinkley, P. Avnet, and C. Carmichael, SEU mitigation design techniques for the XQR4000XL, 2000.

[8] S. Whitaker, "Single event upset hardening CMOS memory circuit," U.S. Patent no. 5 111 429.

[9] J. Canaris, S. Whitaker, and M. N. Liu, "SEU hardened memory cells for a CCSDS Reed Solomon encoder," *IEEE Trans. Nucl. Sci.*, vol. 38, pp. 1471–1477, Dec. 1991.

[10] M. N. Liu and S. Whitaker, "Low power SEU immune CMOS memory circuits," *IEEE Trans. Nucl. Sci.*, vol. 39, pp. 1679–1684, Dec. 1992.

[11] M. Nicolaidis, T. Calin, and R. Velazco, "Upset hardened memory design for submicron CMOS technology," *IEEE Trans. Nucl. Sci.*, vol. 43, pp. 2874–2878, Dec. 1996.

[12] N. Kumar, S. Katkoori, L. Rader, and R. Vemuri, "Profile-driven behavioral synthesis for low power VLSI systems," *IEEE Design Test Comput.*, pp. 70–84, 1995.

[13] C. Tsui, R. Marculescu, D. Marculescu, and M. Pedram, "Improving the efficiency of power simulators by input vector compaction," in *Design Automation Conf. Proc.*, June 1996, pp. 165–168.

[14] R. Marculescu, D. Marculescu, and M. Pedram, "Vector compaction using dynamic Markov models," *IEICE Trans. Fundamentals (Special Issue on VLSI design and CAD Algorithms)*, Oct. 1997.

[15] R. Velazco and D. Bessot *et al.*, "Two CMOS memory cells suitable for the design of SEU-tolerant VLSI circuits," *IEEE Trans. Nucl. Sci.*, vol. 41, pp. 2229–2234, 1994.

[16] M. J. Barry, "Radiation resistant SRAM memory cell," U.S. Patent no. 5 157 625.

[17] J. G. Dooley, "SEU-immune for gate array, standard cell, and other ASIC applications," U.S. Patent no. 5 311 070.

[18] F. Vargas and M. Nicolaidis, "SEU-tolerant SRAM design based on current monitoring," in *24th Int. Symp. Fault Tolerant Computing*, June 1994, pp. 106–115.

[19] M. Nicolaidis, T. Calin, F. Vargas, and R. Velazco, "A low cost, highly reliable SEU-tolerant SRAM: prototype and test results," *IEEE Trans. Nucl. Sci.*, vol. 42, pp. 1592–1598, 1995.

[20] L. R. Rockett Jr., "Simulated SEU hardened scaled CMOS SRAM cell design using gated resistors," *IEEE Trans. Nucl. Sci.*, vol. 39, no. 5, pp. 1532–1541, Oct. 1992.

[21] J. Canaris and S. Whitaker, "Circuit techniques for the radiation environment of the space," in *IEEE 1995 Custom Integrated Circuits Conf.*, 1995, pp. 5.4.1–5.4.4.

[22] J. Venbrux, K. Cameron, K. Arave, L. Arave, M. N. Liu, D. Wiseman, J. Canaris, and K. Liu, "Design and testing of SEU/SEL immune memory and logic circuits in a commercial CMOS process," in *Rec. 1993 IEEE Radiation Effects Data Workshop*, July 1993, pp. 51–55.

[23] K. C. Holland and J. G. Tront, "Probability of latching single event upset errors in VLSI circuits," in *IEEE Proc. Southeastcon '91*, Apr. 1991, pp. 109–113.

[24] J. Von Neumann, "Probabilistic logics and synthesis of reliable organizms from unreliable components," in *Automata Studies*, C. E. Shannon and J. McCarthy, Eds. Princeton, NJ: Princeton Univ. Press, 1956, pp. 43–98.

[25] K. J. Hass, "Probabilistic estimates of upset caused by single event transients," in *8th NASA Symp. VLSI Design*, 1999, pp. 4.3.1–4.3.9.

[26] G. S. Ditlow, J. Savir, and P. H. Bardell, "Radom pattern testability," *IEEE Trans. Computers*, vol. C-33, pp. 79–90, Jan. 1984.

[27] M. A. Breuer, M. Abramvoici, and A. D. Friedman, *Digital Systems Testing and Testable Design*: IEEE, 1990.

[28] D. G. Mavis and P. H. Eaton, "SEU and SET mitigation techniques for FPGA circuit and configuration bit storage design," in *Proc. 2000 Military and Aerospace Applications of Programmable Devices and Technol. Conf.*, Sept. 2000.