Multiple Event Upsets Aware FPGAs Using Protected Schemes

Costas Argyrides, Dhiraj K. Pradhan

University of Bristol, Department of Computer Science
Merchant Venturers Building, Woodland Road, Bristol, BS8 1UB, UK
costas,pradhan@cs.bris.ac.uk

Abstract. Multiple upsets would be available in SRAM-based FPGAs which utilizes SRAM in different parts to implement circuit configuration and to implement circuit data. Moreover, configuration bits of SRAM-based FPGAs are more sensitive to upsets compared to circuit data due to significant number of SRAM bits. In this paper, a new protected Configurable Logic Block (CLB) and FPGA architecture are proposed which utilize multiple error correction (DEC) and multiple error detection. This is achieved by the incorporation of recently proposed coding technique Matrix codes [1] inside the FPGA. The power and area analysis of the proposed techniques show that these methods are more efficient than the traditional schemes such as duplication with comparison and TMR circuit design in the FPGAs.

Keywords. FPGA, SEUs, ECC, Reliability, MTTF

1 Introduction

SRAM-based field programmable gate arrays are being increasingly used to start new designs because of their growing density and speed, reconfigurability, shot-design cycle and cost-effectiveness [2]. While the use of reprogrammable FPGAs offers a number of important advantages, these SRAM-based FPGAs are very sensible to heavy ion, proton and neutron induced single event upsets (SEUs) [3], [4], [5].

There are many available resources within an FPGA to perform various logic functions. The way in which these resources are utilized and interconnected is specified by the circuit design, also known as a configuration bitstream. The configuration bitstream determines which resources within the FPGA are used to implement a specific logic design.The effect of the SEU on the configuration memory of an FPGA, would lead to a permanent error which remains in the FPGA until the next reconfiguration of a new design [6]. This permanent error may result in a logic error or routing error depending on which part of the configuration memory is affected. A logic error may lead to complement one of the entries of the Look-Up Tables (LUTs) modifying the functionality of the mapped logical function [7]. A routing error may lead to a signal getting misrouted or disconnected [8], [5].

Dagstuhl Seminar Proceedings 08371
Fault-Tolerant Distributed Algorithms on VLSI Chips
http://drops.dagstuhl.de/opus/volltexte/2009/1926

Error detection and correction code (EDAC) is a well-known technique for protecting storage devices against transient faults [7]. An example of EDAC is the Hamming code, which is useful for protecting memories against SEU because of its efficient ability to correct single upsets per code word with reduced area and performance overhead.

In order to overcome SEUs affecting the FPGA configuration memory, several fault-tolerance methods have been proposed in the past years. One of techniques, called scrubbing, is periodically reloading the whole content of the configuration memory [2]. By the use of readback and partial reconfiguration capabilities of FPGAs, a recovery system can be used [9]. Through the readback option, the content of the FPGA's configuration memory is read and compared with the expected one, which is stored in a predefined memory located outside of the FPGA. If a mismatch is found, the correct information is downloaded in the FPGA's configuration memory. During reconfiguration only the faulty portion of the configuration memory is overwritten. There are several fault-tolerant techniques that do not consider detection and correction occurred SEUs, but just aim at masking errors not to propagate elsewhere. These methods are proposed mainly by hardware redundancy.

Triple Modular Redundancy (TMR) is a well-known fault-tolerant technique for preventing error propagation [3]. The TMR implementation uses three identical logic blocks performing the same task in parallel regarding to outputs being compared through majority voter. However, this solution enforces high area overhead, three times more input and output pins, high performance penalties [4]. Moreover, it may not be affordable to put redundancy in each and every module (or component) especially in embedded systems where power and area are important constraints. Another error mitigation technique which is based on modular redundancy and time redundancy has been proposed in [10] which uses Duplication with Comparison (DWC) and Concurrent Error Detection (CED) to create a fault-tolerant system. However, this method is depended on the logic of the circuit that is mapped on to the FPGA and suitable encoding and decoding functions for each such block.

In this paper, we introduce different schemes for detecting and correcting errors in configuration bits of the LUTs. These schemes can be applied at different level of FPGA structure. The experimental studies show that using the proposed schemes in FPGAs, all single and double SEUs are detectable and correctable in just one clock cycle without any FPGA reconfiguration and is independent to the number of CLBs. Moreover, using the proposed schemes, the area and power overhead of the new circuit design is more efficient than the previous schemes such as duplication with comparison (DWC) [10] and TMR [3].

The rest of this paper is organized as follows. Section 2 introduces the protection codes and the proposed schemes for the FPGAs. The CLB architecture for fast detection and correction is presented in section 3. Section 4 calculates the probability of having multiple uncorrectable errors in protected Xilinx Virtex II FPGA family. Section 5 compares area, power and correction capability of the proposed technique with related work. Finally section 6 concludes the paper.

2 LUT Protection Methods

2.1 The Matrix Codes (MC)

The proposed detection/correction scheme is called Matrix codes (MC) [1] since the protections bits are used in a matrix format. In this case, the $n - bits$ code word is divided into k_1 number words of width k_2. A (k_1, k_2) matrix is formed where k_1 and k_2 represents numbers of rows and columns. For each of the k_1 rows, the check bits are added for single error correction/double error detection. Another k_2 bits are added as vertical Parity bits. We explain the basic technique by considering a 32 bit word length memory. In this situation, a 16 bit word is divided into a matrix as shown in Fig. 1, where $k_1 = 4$ and $k_2 = 8$. Hamming codes are applied for each row. For a $8 - bit$ data, 4 Hamming check bits are required, so 5 check bits are added at the end of the 8 data bits.

X_0	X_1	X_2	X_3	X_4	X_5	X_6	X_7	C_0	C_1	C_2	C_3	C_4
X_8	X_9	X_{10}	X_{11}	X_{12}	X_{13}	X_{14}	X_{15}	C_5	C_6	C_7	C_8	C_9
X_{16}	X_{17}	X_{18}	X_{19}	X_{20}	X_{21}	X_{22}	X_{23}	C_10	C_11	C_{12}	C_{13}	C_14
X_{24}	X_{25}	X_{26}	X_{27}	X_{28}	X_{29}	X_{30}	X_{31}	C_{15}	C_{16}	C_{17}	C_{18}	C_19
P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7					

Fig. 1. 32 bits Logical Organization of Matrix Codes

The check bits are calculated using the following.

$$C_0 = X_0 \oplus X_1 \oplus X_3 \oplus X_4 \oplus X_6 \tag{1}$$
$$C_1 = X_0 \oplus X_2 \oplus X_3 \oplus X_5 \oplus X_6 \tag{2}$$
$$C_2 = X_1 \oplus X_2 \oplus X_3 \oplus X_7 \tag{3}$$
$$C_3 = X_4 \oplus X_5 \oplus X_6 \oplus X_7 \tag{4}$$
$$C_4 = X_0 \oplus ... \oplus X_7 \ (Overall\ parity) \tag{5}$$

Accordingly we calculate all check bits for all rows.
For the Parity row we will use the following formulas.

$$P_0 = X_0 \oplus X_8 \oplus X_{16} \oplus X_{24} \tag{6}$$
$$P_1 = X_1 \oplus X_9 \oplus X_{17} \oplus X_{25} \tag{7}$$
$$P_2 = X_2 \oplus X_{10} \oplus X_{18} \oplus X_{26} \tag{8}$$
$$P_3 = X_3 \oplus X_{11} \oplus X_{19} \oplus X_{27} \tag{9}$$
$$P_4 = X_4 \oplus X_{12} \oplus X_{20} \oplus X_{28} \tag{10}$$
$$P_5 = X_5 \oplus X_{13} \oplus X_{21} \oplus X_{29} \tag{11}$$
$$P_6 = X_6 \oplus X_{14} \oplus X_{22} \oplus X_{30} \tag{12}$$
$$P_7 = X_7 \oplus X_{15} \oplus X_{23} \oplus X_{31} \tag{13}$$

In other words, X_0 through X_{31} are the data bits, C_0 through C_{19} are the check bits, $P_1 - P_7$ are Parity bits. A Hamming decoder is used to decode each row. Decoding is done in two steps. First, the syndrome bits are calculated and the check bits are generated using the data bits and compared with the syndrome bits saved check bit. This procedure is called syndrome bit generation and S_1 is called syndrome bit of check bit '1'. Second, using syndrome bits, the Single Error Detection (SED)/Double Error Detection (DED)/No Error (NE) signals are generated for each row. If DED is generated using each bit's Parity syndrome bits SPi and the saved value of the bit we can correct any single or double erroneous bits in each row using Equation 14.

$$X_{i_{correct}} = (X_{i_{err}} \oplus O_i) \oplus (DED_j * SP_k) \tag{14}$$

Where $X_{i_{err}}$ is the erroneous bit, O_i the decoder output corresponding to the data bit i.

It is important to mention that if we have more than two errors in each code word, MC can correct them if and only if we have only two errors in each row of the matrix and one in each column (1). If we have only two errors in the entire code word, then these can be corrected without any restriction.

3 Protection levels

3.1 FPGA-level protection

In this scheme, protection is performed for one row of FPGA's CLBs. At time of fault detection and correction, the contents of all LUTs inside of one CLB in the row are read and the syndrome and overall parity is generated. The overall parity and the syndromes are used for the multiple correction and detection if required. We consider each row (element) of a CLB as a row in Fig. 1, and thus a Hamming circuitry is required for each of them. In this scheme, some modification can be applied for decreasing the area overhead of FPGA-level protecting scheme. As each CLBs are checked by corresponded circuitry, therefore we need M different SEC-DED (Hamming) circuitry for each CLBs where M is the number of CLBs in a row. However, we can use just one Matrix circuitry and share it for all rows of the FPGA. This make the area overhead to be decreased but the testing time of each CLB will be increased. We considered these two schemes with name of FPGA-level *with* and *without* shared circuitry in the experimental results.

Fig. 2 shows a simple example of the implementation of FPGA-level protection in which the protection codes are considered for a row of FPGA with four columns FPGA and the protected FPGA's columns are increased to seven columns (Note that for DED we need one more CLB). The gray box show modifications needed to implement the protection code. In this scheme, protection is performed for one row of FPGA's CLBs. At time of fault detection and correction, the contents of LUTs inside of one CLB row are read and the syndrome and overall parity are generated. In this scheme, some modification can be applied for decreasing the area overhead of FPGA-level protecting. As each LUT inside

of CLBs are checked by corresponded SEC-DED circuitry, therefore we need M different SEC-DED circuitry for each LUTs where M is the number of LUTs inside of CLBs. However, we can use just one SEC-DED circuitry and share it for all rows of LUTs. This make the area overhead to be decrease but the testing time of each CLB will be increased. We considered these two schemes with name of FPGA-level with and without shared circuitry in the experimental results.

It should be noted that in this scheme, the length of information bits which is used for protecting is based on the number of columns that FPGA has. This means that the protection capability of this scheme is significantly depended to FPGA size. For example, if the dimensions of FPGA increase, the protection capability of this scheme would decrease.

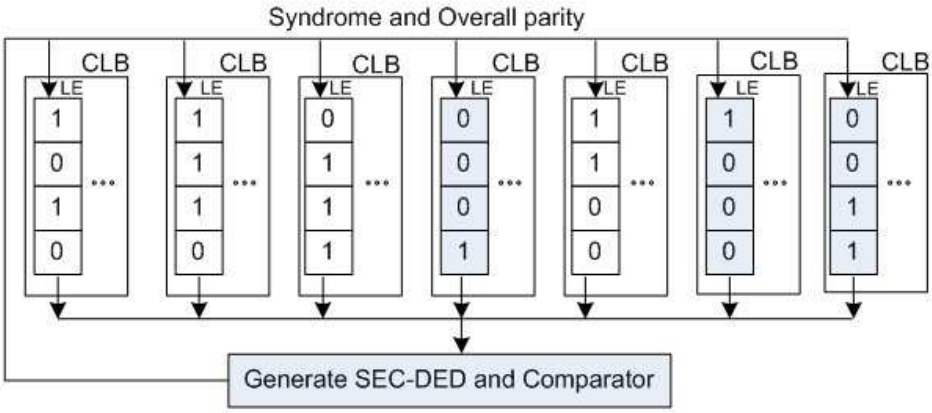

Fig. 2. FPGA-Level protection: an FPGA with SEC-DED protection CLBs

3.2 CLB-level protection

Utilizing Matrix codes can be applied for all of LUTs inside of a CLB, consider each LUT as a row in Fig. 1. Fig. 3 shows a CLB which is protected by Hamming codes. B_0, B_1, B_3 are LUTs required for storing the protection codes of B_2, B_4, B_5 and B_6 LUTs. In this case, all bits in the same significant bit positions in different LUTs are protected in the same significant bit positions in the protection blocks. In this architecture, since the information and protection bits are stored apart and in separated blocks (LUTs), therefore the probability of having more than double errors in each LUT of information and protection bits will be decreased significantly. In this case, all of multiple errors occurred in only one LUT of a CLB can be detected and corrected but if multiple errors occurred in different LUTs of a CLB in same bit positions, they may be detected providing that the number of errors is equal or less than two.

In order to implement this level of protection, a $k-bit$ counter is required to address different bit position of each LUT. The detection and correction of errors in LUTs of a CLB can be achieved by 2^k times of detection and correction for each bit inside of a LUT. The main difference between this level of protection and FPGA-level one is that the information bits in this scheme are much less than the previous one. Moreover, all connections between information and protection bits are router inside of CLB internally and therefore this method is more modular than the previous one. However, the area overhead of this scheme is more than FPGA-level. The implementation of CLB-level protection codes can be done in two different cases with and without sharing the endoding/decoding circuitry. In the case of sharing the circuitry, area overhead of protection is decreased but the time of detection and correction of errors will be increased 2^k times.

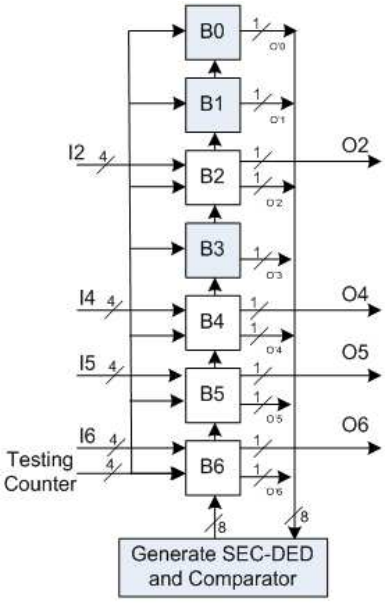

Fig. 3. CLB-Level protection: a CLB with SEC-DED protection LUTs

In Fig. 3, each 16X1 LUT is replaced by a dual-read LUT shown in Fig. 4. Therefore, every CLB has 4 additional input lines that consist of the four output lines of testing counter. The testing counter is a 4-bit counter, 0-15 binary up-counter, provided either on FPGA chip or kept as a stand-lone counter, incremented once every clock cycle. In addition to the LUTs used by the circuit mapped to the FPGA, a few protection LUTs are also added to every CLB of FPGA. These protection LUTs store the pre-computed 16-bit SEC-DED check bits of the other LUTs of CLB. The architecture shown in Fig. 3 performs at-

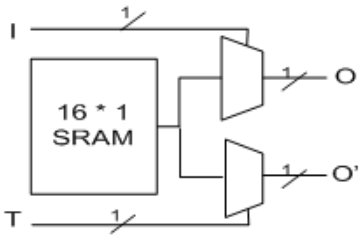

Fig. 4. A dual-port 4-input LUT

speed detection and correction of any single or double error of configuration bits of LUTs without disturbing the normal functioning of the FPGA.

In Fig.5 we show how to employ employing Hamming codes used in a LUT. The gray shapes in this figure show the modifications needed for implementing it in a LUT. In this scheme, each LUT (row in Fig 1) in a FPGA has its own protection code and therefore all single and double errors inside of one LUT can be detected and corrected. The area overhead of this scheme is more than the previous two schemes since each LUT has separated protection circuitry.

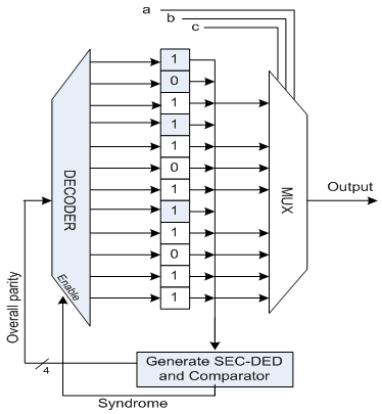

Fig. 5. LUT-level protection: a LUT with SEC-DED protection bits

Based on these three mentioned protection levels, CLB-level protection is suggested for employing in FPGAs. Fig. 4 shows the proposed technique applied on CLB architecture. The main reasons are that CLB-level protection is less complex than FPGA-level to layout the FPGA by manufacturers since the protection structure is localized in each CLB architecture and the protection code

Table 1. Comparison of area and delay overheads and FPGA testing time for 16×16 FPGA (CLB size= 8 3-input LUTs)

Level of protection codes	No. of information + check bits	area m^2	delay(ns)	FPGA testing delay
FPGA testing delay FPGA-level (with shared HW)	16384+3584	2030394	12	$16T_{32}$+ AND + EXOR
FPGA-level (without shared HW)	16384+3584	2166342	12	T_{32}+ AND + EXOR
CLB-level (with shared HW)	16384+8192	2030393	14.6	$256T_8$+ AND + EXOR
CLB-level (without shared HW)	16384+8192	4341458	14.6	T_8+ AND + EXOR

routings are inside of each CLB. Typically, manufacturers manually layout a single tile consisting of logic block and switch block and replicate them across the entire chip. Therefore, CLB-level and LUT-level protection schemes are better ones for implementing compared to FPGA-level. However, CLB-level protection is more reliable than LUT-level, since the information bits that is protected by each check bits are distributed through several LUTs.

Without loss of generality, we assume that FPGA design used for the fault tolerance is composed of $16X16$ CLBs arranged in a square matrix and each CLB consists of 8 3-input LUTs.

Table 1 shows different implementation of the mentioned schemes and compare them in terms of information and check bits, area, delay of detecting and correcting information bits and delay of testing whole FPGA. In this table, T_8 and T_{32} are time requited for performing encoding and decoding for 8 and 32 data bits, respectively. For each protection scheme, two cases of implementation are considered based on sharing or not sharing the hardware implementation for a group of similar information bits. For example, in the LUT-level with shared hardware, all LUTs inside one CLB are considered to share the hardware needed for encoding, decoding logics needed for Matrix Codes. FPGA-level protection scheme has less area overhead compared to CLB-level and LUT-level. However, FPGA-level testing time is more than other cases because it protects information bits more than others. Moreover, it is more complex to implement compared to others. CLB-level and LUT-level have same area and delay overhead, since the size of LUT and number of LUTs inside one CLB is same in the mentioned FPGA. However, CLB-level is more powerful in terms of correction multiple faults in one LUT.

4 Detection and Correction of multiple faults

In order to estimate the error detection and correction coverage of the proposed technique and previous one, we used fault injection method. Fault injection is

one of the key methods to estimate the error study of the circuits which utilized error detection and correction codes. Using fault injection method, the coverage of proposed method can be estimated.

4.1 Fault Injection Experiments

Without loss of generality, we considered the coverage of the proposed technique for one CLB of the same FPGA as shown in Table 1 since the protection code can be applied on each CLB. The size of a CLB is 8 LUTs of 8 bits each. Both single and multiple faults were injected. For each number of faults in the case of multiple fault injection, about one million experiments were conducted. The obtained values are portrayed in Table 2. For each protection code, there are two separated column for fault detection and correction coverage. The first column shows the number of faulty bits in a word. As we can observe from this table, the fault detection and correction coverage of Matrix are better than Hamming, DWC and TMR. Additionally, in the case of Matrix method, several numbers of faults can be detected or corrected. In this case, as the number of faulty bits increase, the fault detection or correction coverage also decreases.

Table 2. The fault detection and correction coverages for different protection schemes

# of Faults	Matrix Det. (%)	Matrix Corr. (%)	Hamming Det. (%)	Hamming Corr. (%)	TMR Det. (%)	TMR Corr. (%)	DWC Det. (%)	DWC Corr. (%)	No Protection Det. (%)	No Protection Corr. (%)
1	100	100	100	100	100	100	100	0	0	0
2	100	100	100	0	0	0	0	0	0	0
3	96.32	81.25	0	0	0	0	0	0	0	0
4	82.69	58.94	0	0	0	0	0	0	0	0

The reliability of the FPGA is strongly dependent on the reliability of the CLBs. Hence, it is imperative to analyze the reliability of such an architecture technique to validate its applicability in real designs. In order to analyze the reliability of the proposed architecture we make the following assumptions[1]:

1. The probability of a number of faults occurring in a fixed period of time with a known average rate is independent of the time since the last event. (Poisson distribution).
2. Bit failures are statistically independent.
3. We dont take the reliability of the switches into consideration while calculating the reliability of different configurations.

The probability of having exactly i Bit Flips (BF) in a CLB can be given by:

$$Prob\{iF\} = \binom{BF}{i} \cdot (1 - e^{-\lambda t}) \cdot e^{-\lambda(BF-i)t} \tag{15}$$

where λ is the fault rate of a bit flip and t is time parameter.
The reliability, r(t) of a CLB can be expressed as:

$$r_t = P\{NE\} + \sum_{i=0}^{N_{IC}} P\{iF\} \tag{16}$$

where $P\{NE\}$ denotes the probability that there is no error, and $P\{iF\}$ indicates the probability of having i faults. Based on these schemes, the reliability of the FPGA is the product of the reliability of all its CLBs and can be given by:

$$R(t) = r^M(t) \tag{17}$$

where M is number of CLBs in the FPGA. The integration of the reliability function give the mean time to failure MTTF i.e.,

$$MTTF = \int_0^\infty R(t)dt \tag{18}$$

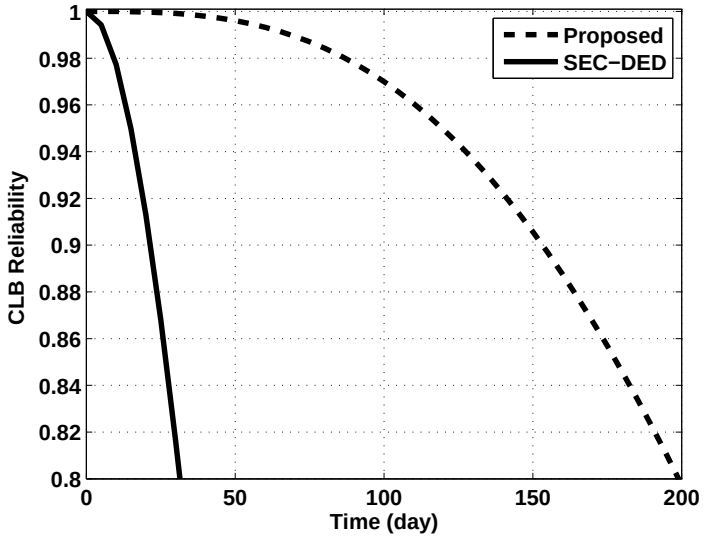

Fig. 6. Reliability of 128bits CLB, $\lambda = 10^{-5}$

Formulas 15 to 18 were described and solved using MATLAB for estimating the reliability and MTTFs of different FPGA architectures with different protection schemes.

The reliability of each 128bit CLB inside the FPGA as shown in Fig. 6 using the proposed technique will be more than 4X, the FPGA reliability will be fall

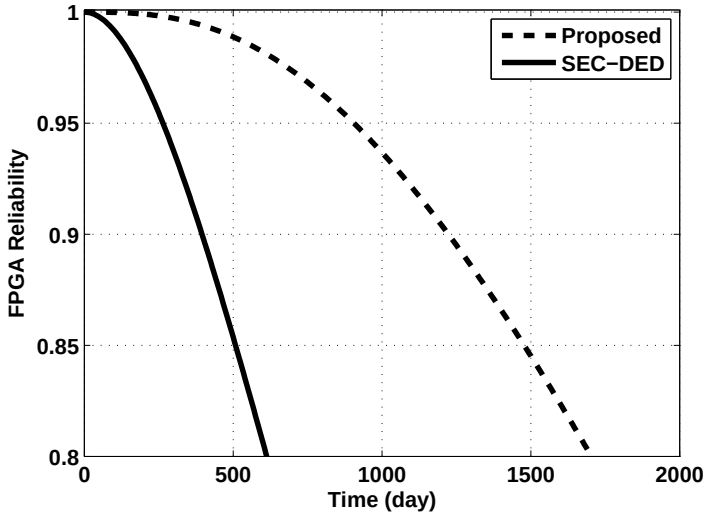

Fig. 7. Reliability of a 16×16 CLBs FPGA, $\lambda = 10^{-5}$

below 0.8 before 50days and the reliability of the FPGA will fall below 0.8 at about 200days.

Fig. 7 shows that the overall reliability of the FPGA using proposed technique is almost 3X compared to the reliability of the FPGA using SEC-DED technique [11].

Table 3 shows the Mean Time to Failure of the FPGAs using different protecting schemes for different fault rates At the fault rate of $\lambda = 10^{-5}$, The MTTF of the proposed configuration is almost 4X larger than that of the SEC-DED. While at $\lambda = 10^{-3}$ and $\lambda = 10^{-4}$ the improvement reduces to 3X.

Table 3. MTTF for a 16×16 CLBs FPGA in different fault rates

Fault rate	Proposed	Hamming
$\lambda = 10^{-3}$	3.5353	1.1196
$\lambda = 10^{-4}$	35.3534	11.1960
$\lambda = 10^{-5}$	308.5610	65.1948

The obtained values are portrayed in Table 2. For each protection type, there are two separated column for fault detection and correction coverage. The first column shows the number of faulty bits in a word. As we can observe from this table, the fault detection and correction coverage of Matrix and is much better than SEC-DED technique, DWC and TMR. Additionally, in the case of Matrix method, several numbers of faults can be detected or corrected. In this case, as

the number of faulty bits increase, the fault detection or correction coverage also decreases. The reliability of the FPGA is strongly dependent on the reliability of the CLBs. Hence, it is imperative to analyze the reliability of such an architecture technique to validate its applicability in real designs.

4.2 Analytical Models

Based on the different levels of protections, the CLB-level is the best level for the protection using Matrix codes method since it can detect multiple adjacent faults in the LUTs with good level of modularity and less complexity. We propose the CLB-level protection method to be used in the FPGA. In this case, any multiple faults in same significant bit position of LUTs of a CLB are detectable while double fault at each significant bit position of LUTs of a CLB are correctable. In this scheme, the probability of un-correcting triple errors inside of a FPGA will be decreased significantly.

When four upsets occurred, if they happened each 2 of them in same bit positions of LUTs of a CLB cannot be correctable. Let N be the number of LUTs in the device and each CLB composed of 8 4-input LUTs and CLB-level protection is employed in the FPGA. The probability of having four and five configuration upsets undetectable by this scheme is given by:

$$P_{3\ undetectable\ errors} = \frac{\binom{16N}{1} \cdot \binom{12}{3}}{\binom{192N}{3}} \tag{19}$$

$$P_{4\ undetectable\ errors} = \frac{\binom{16N}{1} \cdot \binom{12}{4}}{\binom{192N}{4}} \tag{20}$$

$$\begin{aligned} P_{5\ undetectable\ errors} = & \frac{\binom{16N}{1} \cdot \binom{12}{3} \cdot \binom{15N}{1} \cdot \binom{12}{2}}{\binom{192N}{5}} \\ & + \frac{\binom{16N}{1} \cdot \binom{12}{4} \cdot \binom{15N}{1} \cdot \binom{12}{1} + \binom{16N}{1} \cdot \binom{12}{5}}{\binom{192N}{5}} \end{aligned} \tag{21}$$

Also, the probability three and four configuration upsets uncorrectable by this scheme are given by:

$$P_{3\ uncorrectaable\ errors} = \frac{\binom{16N}{1}\binom{12}{3} + \binom{16N}{2}\binom{12}{2}\binom{12}{1}}{\binom{192N}{3}} \tag{22}$$

$$\begin{aligned} P_{4\ uncorrectable\ errors} = & \frac{\binom{16N}{1}\binom{12}{4} + \binom{16N}{2}\binom{12}{2}\binom{12}{2}}{\binom{192N}{4}} \\ & + \frac{\binom{16N}{2}\binom{12}{1}\binom{12}{3} + \binom{16N}{3}\binom{12}{2}\binom{12}{1}\binom{12}{1}}{\binom{192N}{4}} \end{aligned} \tag{23}$$

We computed the probabilities for a series of Xilinx FPGAs which are mentioned in Tables 4 and 5. As these table shows, the probability of having 3 uncorrectable errors for the mentioned scheme is very low and this probability decreases when the size of FPGA increased. However, the probability of having four and five errors uncorrectable in FPGA is more than the probability of having three uncorrectable in FPGA because the employed protection code can correct double faults. It should be noted that in the real application the probability of occurring four and five errors is considerably less than the probability of occurring three errors. Therefore, if the correction of LUT contents happened in appropriate time slots, the content of LUTs will not be erroneous.

Table 4. The probability of multiple errors not being detectable for protected Xilinx Virtex II FPGAs

Device	No. of CLBs	Prob. of having 3 undetectable errors	Prob. of having 4 undetectable errors	Prob. of having 5 undetectable errors
XC2V40	8 x 8	7.28E-007	5.33E-010	8.79E-009
XC2V80	16 x 8	1.82E007	6.67E-011	1.09E-009
XC2V250	24 x 16	2.02E-008	2.47E-012	4.07E011
XC2V500	32 x 24	5.06E-009	3.08E-013	5.08E-012
XC2V1000	40 x 32	1.82E-009	6.66E-014	1.09E-012
XC2V1500	48 x 40	8.09E-010	1.97E-014	3.26E-013
XC2V2000	56 x 48	4.13E-010	7.20E-015	1.19E-013
XC2V3000	64 x 56	2.32E-010	3.03E015	5.01E-014
XC2V4000	80 x 72	8.99E-011	7.32E-016	1.21E-014
XC2V8000	112 x 104	2.20E-011	8.86E-017	1.46E-015

Table 5. The probability of multiple errors not being correctable for protected Xilinx Virtex II FPGAs

Device	No. of CLBs	Prob. of having 3 uncorrectable errors	Prob. of having 4 uncorrectable errors
XC2V40	8 x 8	1.30E-003	1.80E-003
XC2V80	16 x 8	6.71E-004	8.95E-004
XC2V250	24 x 16	2.28E-004	2.98E-004
XC2V500	32 x 24	1.12E-004	1.49E-004
XC2V1000	40 x 32	7.71E-005	8.95E-005
XC2V1500	48 x 40	4.48E-005	5.97E-005
XC2V2000	56 x 48	3.20E-005	4.26E-005
XC2V3000	64 x 56	2.40E-005	3.20E-005
XC2V4000	80 x 72	1.49E-005	1.98E-005
XC2V8000	112 x 104	7.37E-006	9.83E-006

5 Area and Power overhead

The CLB architecture shown in Fig. 2 and Fig. 3, parity-protected, DWC, TMR and SEC-DED FPGA architectures were synthesized with Synopsys©CAD tool and 0.18 micron CMOS technology to compare the area, power and delay requirements. The advantage and disadvantage of proposed architecture over standard DWC technique in terms of area, power, delay and additional configuration memory requirements are shown in Table 6. The area overhead of parity-protected CLB architecture is about 48 percent regarding to the area of simple CLB architecture. The area overhead of DWC and TMR methods is also about 79 and 204 percent compared to the simple CLB architecture. Based on these results the area overhead of our proposed technique is less than DWC and TMR schemes. In the case of power consumption, the parity-protected CLB architecture consumes less power among the other protection schemes, but it can be only used for detecting errors. However, the power consumption of the proposed technique is less than the DWC and TMR but, ore than SEC-DED schemes. This is expected since the implementation of the proposed hardware causes several extra check bits (DWC and TMR) and routes to perform error detection and correction.

Table 6. Comparison of area, power and configuration memory requirement for a CLB

CLB architecture	No. of LUTs	Area m^2	%	Power w	%	No. of SRAM bits	Single Error Det.	Double Error Det.	Single Error Cor.	Double Error Cor.
Standard FPGA (Virtex II)	8	10240	100	230	100	128	0%	0%	0%	0%
Protected FPGA with parity [8]	9	15258	149	331	144	144	100%	0%	0%	0%
Duplication with comparison [10]	16	16282	179	525	228	256	100%	100%	0%	0%
TMR-based FPGA [3]	24	31130	304	802	348	348	100%	100%	0%	0%
FPGA with SEC [11]	13	16506	161	532	231	192	100%	100%	0%	0%
Our proposed FPGA	14	16506	161	532	231	160	100%	100%	100%	100%

6 Conclusions

In this paper, we have presented two mechanisms to tolerate multiple upsets in LUTs of SRAM-based FPGAs and compared with a SEC-DED technique. The MTTF and the reliability analysis have shown that by employing Matrix codes the MTTF and the Reliability of the FPGA will be improved by at least 3X.

Matrix codes were employed in the FPGA using two techniques. The first is employing Matrix codes in each CLB and in the second case, employing Matrix codes in each row of FPGA to correct double errors in CLBs. On the other hand the analytical results have also shown that using the proposed CLB architecture improves the reliability of CLB so that the probability of having four uncorrectable errors in a CLB is decreased significantly. The results of implementation comparison have shown that this method imposes less area and power overhead compared to the previous fault-tolerant schemes such as duplication with comparison and triple modular redundancy schemes.

References

1. Argyrides, C., Zarandi, H., Pradhan, D.K.: Matrix codes: Multiple bit upsets tolerant method for sram memories. 22nd IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'07) (2007)
2. Application, X.: Single event upsets through virtex partial reconfiguration (June 2000) Note XAPP216.
3. Kastensmidt, F.L., Sterpone, L., Carro, L., Reorda, M.S.: On the optimal design of triple modular redundancy logic for SRAM-based FPGAs. IEEE Design, Automation and Test in Europe (2005) 1290–1295
4. Asadi, G., Tahoori, M.B.: Soft error mitigation for sram-based FPGAs. 23th IEEE VLSI Test Symposium (2005) 207–212
5. Srinivasan, S., A.Gaysen, N.Vijaykrishnan, M.Kandemir, Y.Xie, Irwin, M.: Improving soft-error tolerance of FPGA configuration bits. IEEE/ACM International Conference on Computer Aided Design (2004) 107–110
6. Reorda, M.S., Sterpone, L., Violante, M.: Multiple errors produced by single upsets in FPGA configuration memory: a possible solution. IEEE European Test Symposium (2005) 136–141
7. Reddy, E.S.S., Chandrasekhar, V., Sashikanth, M., Kamakoti, V.: Novel CLB architecture to detect and correct SEU in LUTs of SRAM-based FPGAs. IEEE International Conference on Field-Programmable Technology **39** (2004) 121–128
8. Reddy, E.S.S., Chandrasekhar, V., Sashikanth, M., Kamakoti, V.: Online detection and diagnosis of multiple configuration upsets in luts of SRAM-based FPGAs. 19th IEEE International Parallel and Distributed Processing Symposium (2005) 172–175
9. Gokhale, M., Graham, P., Johnson, E., Rollins, N., Wirthlin, M.: Dynamic reconfiguration for management of radiation-induced faults in FPGAs. 18th IEEE Parallel and Distribution Processing Symposium (2004) 145–150
10. Lima, F., Carro, L., Reis, R.: Designing fault tolerant systems into SRAM-based FPGAs. IEEE/ACM Design Automation Conference (2003) 650–656
11. Zarandi, H., Miremadi, S., Argyrides, C., , Pradhan, D.K.: Online detection and correction of soft-errors in luts of sram-based FPGAs. 12th European Test Symposium - ETS 2007 (2007)