INTERNATIONAL JOURNAL OF SCIENTIFIC PROGRESS AND RESEARCH (IJSPR) ISSN: 2349-4689
Issue 184, Volume 79, Number 6, June 2021

Priority Based Scrubbing For Multi-Bit Upsets On SRAM FPGAs

Aiswariya A.P., Nithin Joe

ECE Department,Nehru College of Engineering And Research Centre, Pampady, Thrissur, Kerala

Abstract- The SRAM-based field-programmable gate array (FPGA) is extremely susceptible to single event upsets (SEUs) on configuration memory which can lead to soft error and malfunction of the circuit. Facing the ever-growing number of configuration bits in modern FPGAs, traditional scrubbing is getting harder to find errors in time, resulting in mismatching between the SEU sensitivity and scrubbing performance. The proposed method uses a hierarchical scrubbing based on priority technique that makes full use of the single-bit and multi-bit sensitivity based on the adaptive mean time to detect (MTTD) for each frame. The single-bit upset is performed using duplication with comparison and multi-bit upset by using multi dimensional parity bits. It distinguishes the configuration frames with multi-priority and uses different scrubbing methods for different priorities. Also, a model has been built for solving the MTTD allocating problem and enabling an effective scrubbing when SEU and MBU occur. Moreover, the corresponding hardware architecture is supported and the fault injection-based evaluation on a Xilinx FPGA is done. The result shows that it can improve mean upsets to failure, which is proportional to the mean time to failure (MTTF) improvement.

Index Terms- Field-programmable gate array (FPGA), hierarchical scrubbing, mean time to detect (MTTD)allocating, multifrequency scrubbing, rapid scrubbing, single event upset (SEU)and MBUs.

I. INTRODUCTION

SRAM-BASED field-programmable gate arrays (FPGAs) have been widely used especially in aerospace applica-tions in the past few years. The reason is that SRAM-based FPGAs with high logic density can be reconfigurable, which makes the applications more flexible. However, the aeronautical environment contains a large number of high-energy radiation particles. The radiation particle flux increases with altitude. For example, the neutron flux at 2000 m is five times higher than that at sea level, and only 1% of neutrons created by cosmic rays reach the earth’s surface [1]. This may cause the SRAM signal in the configuration memory to change [2].

These errors, which may lead to system failure until being reprogrammed, are called soft errors.Single event upset (SEU) is a kind of typical soft error, specifically representing the data flipping of an SRAM cell.

For reducing the impact of the SEU, there are three mainkindof methods including manufacturing-, redundancy-, andscrubbing-based methods. Manufacturing-based methods areoften expensive and cannot eliminate SEU, whereas they can reduce the SEU sensitivity of the circuits. Triple modular redundancy (TMR) is a typical redundancy-based method for spatial redundancy, working with signal voting by copying hardware to mask errors [3]–[6]. Although TMRisuseful,its large area overhead limits the applications in general. Dual-modularredundancy [7],[8]isanother redundancy-based method to reduce area overhead. However, redundancy-based methods cannotrepair SEU.Thescrubbing-based methods just provide an available method to repair the SEU and avoid the accumulation ofSEU.

In the FPGA configuration layer, the scrubbing-based method [9] is used to protect circuits from SEU. A common circuit is used tocheck and correct the SEU that happened in the configuration memory. The traditional blind scrubbing method periodically overwrites all the configuration memories to correct the SEU. This method needs a gold memory to store the bitstream of the whole configuration. Another scrubbing method is used to periodically read back the bit stream [10] and to detect theSEUbyencoding [11],[12]like ECC.In Stoddard *et al.* [13], both were combined based on theZynq device with ARM core. However, for FPGAs without ARM core, the detection of SEU it stilldifficult.

To deal with this problem, researchers focus more on the connection between the scrubbing methods and test circuits. One kind of optimization for scrubbing is called on-demand scrubbing [14]. This method obtains some signals compared with original circuits and their copies [15]. Through these signals, the shifted scrubbing method [16]–[18] can narrow down the range of SEU position and respond to fix errors within a short time. What is more, the rapidscrubbingmethod [19] could locate the SEU from the application layer to the configuration layer to minimize the repaired time. Except on-demand scrubbing, another optimizationisdone to improve configuration scrubbing rather than using the signals from original circuits. A flexible design framework for hardware scrubber is described in Herrera-Alzu and López- Vallejo [20]. Context-aware resource placement strategy is used in Fouad *et al.* [21] to minimize the configuration frames that need to be protected. Based on the closed resource arrangements, the selective read backmethod [22] reduces the scrubbingframes toenhance theefficiency ofthedetection.

With further research, researchers distinguish the configuration frames to used frames and unused frames. But even if two different frames belong to used frames, their SEU sensitivity is different [23]. In this article, we propose a hierarchical scrubbing technique to distinguish the frames with more priorities and decide the suitable scrubbing method for each configuration frame to make full use of the SEU sensitivity. The main contributions of this work can be summarized as follows.

1) A hierarchical scrubbing technique is proposed.This

technique distinguishes the configuration frames with four priorities according to the SEU sensitivity and then uses different scrubbing methods for different priorities to make full use of the SEU sensitivity. As for the first priority, the hierarchical scrubbing uses the rapid scrubbing method. As for the second and third priorities, the hierarchical scrubbing adjusts the scrubbing rate to suit SEU sensitivity. The fourth priority is unused frames, which would beignored.

A multifrequency scrubbing method is proposed tomax- with mean time to failure (MTTF) and positively related to failures in time (FIT). If the soft error effect without scrubbing is S, the one configuration bit EMR under scrubbing can be expressed as follows:

□
□ $\frac{\text{MTTD}-\text{MTTM}}{}\times S$,MTTD>MTTM

2) imize scrubbing performance for the second and third priorities with nearly no extra resource overhead. This EMR=MTTD □0, MTTD ≤MTTM.

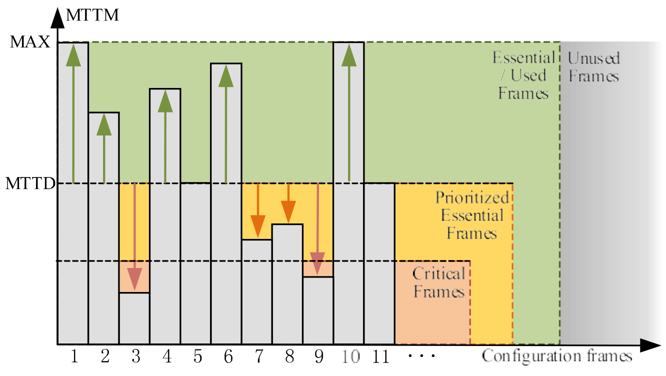


Fig. 1. MTTM and MTTD of configuration frames.

method adjusts the scrubbing rate for each configuration frame to make full use of the SEU sensitivity. Also, a model has been built for solving the optimization problem. It consists of three subproblems, and the multi-frequency scrubbing method provides optimal solutions for every problem.

3) The corresponding hardware architecture is supported, and its evaluation based on a random fault injection test has been done on the Xilinx Kintex-7 FPGA device. Onthebasisofthetest results,thehierarchical scrubbing technique canimprove mean upsets tofailure from

1.56 to 146.93 when comparing with the conven-

tional traversal scrubbing method provided by the Xilinx SEM.

The remainder of this article is organized as follows.

Section II describes the evaluation of SEU sensitivity. Section III introduces the proposed hierarchical scrubbing technique in detail. The multifrequency scrubbing method is introduced in Section IV. Then, Section V explores the hardware design of the hierarchical scrubbing. Section VI shows the setup of the experiment and the results of the random fault injection test. Finally, Section VII concludes this article.

II. SENSITIVITY EVALUATION

A. *Quantitative Model*

A quantitative analysis model is established by Asadi and Tahoori [24] for analyzing the scrubbing technology. The model indicates that the scrubbing effect is primarily deter- mined by mean time to manifest (MTTM) and mean time to detect (MTTD). MTTM refers to the average time from the SEU occurrence to the system failure, while MTTD represents the average time from the SEU occurrence to the discovery of the SEU position on the FPGA configuration layer. In this model, the soft error effect due to scrubbing is evaluated using error manifestation rate (EMR). EMR isnegatively correlated.

In addition, the EMR of the whole circuit can be obtained by accumulating the EMR of all configuration bits.

In general, MTTM is the circuit characteristic, whereas MTTD is determined by scrubbing, thus we obtain the fol- lowing empirical equation:

$$\text{MTTD} \quad 1 \quad t \times N_f. \qquad (2)$$

Here, t represents the time to check one frame, which is a fixed value. N_f represents the number of frames that need to be checked. The configuration frame is the smallest unit of the FPGA scrubbing technique, which consists of many configuration bits. All the operations such as reading and writingtheconfigurationdataarebased onthewhole frame.

B. *Analysis andPriority*

The conventional traversal scrubbing method makes all the configuration frames to have the same MTTD, but not all of the configuration frames have their MTTD adapted to their MTTM. It can be seen from Fig. 1 that some configuration frames have a smaller MTTM and some have a larger MTTM. Based on (1), consider a frame with a large MTTM. Its acceptable MTTD is large, so it is not necessary to detect this frame so frequently. The reason is that the improvement of this configuration bit from MTTD finally provides a little contribution to the system EMR. Configuration frames with smaller MTTM require a smaller MTTD to reduce EMR so thatthesystem EMRwouldbesmaller.

Therefore, this article proposes a hierarchical scrubbing technique that can make full use of the SEU sensitivity of different configuration frames. This method, through the MTTM, allows to decide the priority of the configuration frames andthen uses twoscrubbingmodes tomake theMTTD more suitable for MTTM, just as the arrows shown in Fig. 1. The difference between the hierarchical scrubbing and other scrubbing methods is that hierarchical scrubbing coulddeal with each frame on its merits. For each frame, we try to find suitable protection with low overhead and make full use of the sensitivity.

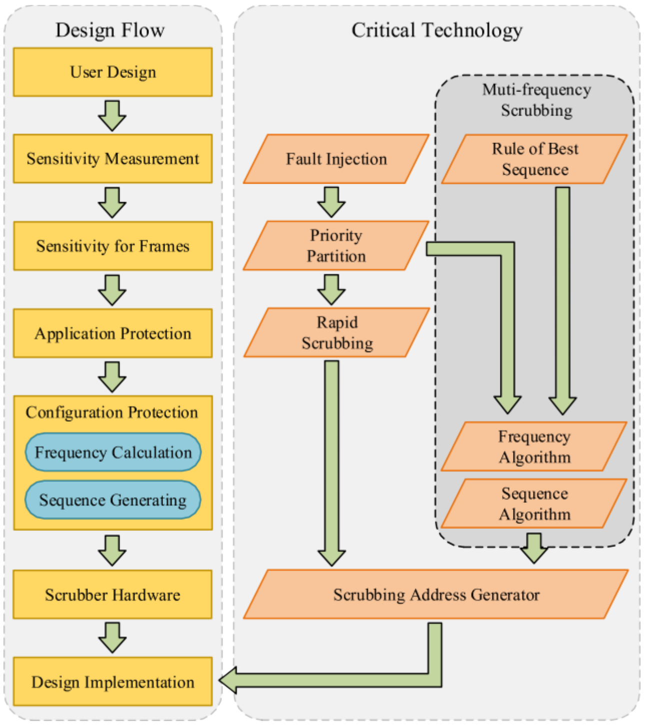


Fig. 2. Procedure of hierarchical scrubbing technique.

As shown in Fig. 1, all the frames are divided into four priorities. The first priority includes the configuration frames which are most sensitive to the SEU. To protect these frames, we use duplication with comparison (DWC) to locate the position of the SEU and make a rapid scrubbing. The second prioritycontainstheframes whicharesensitive enough butnot so sensitive as the first priority. The third priority consists of the frames which indicate that the system failure only happens in a few cases when SEU takes place. For these frames, the MTTD of the second and third priorities is adjusted to let their MTTD matches their MTTM. The rest frames, also called unused frames, belong to the fourthpriority.

III. HIERARCHICAL SCRUBBING TECHNIQUE

A. *Overview*

The scrubbing methods used in hierarchical scrubbing pro- tect different sensitive configuration frames from two per- spectives, which adopted the FPGA application layer and the configuration layer. The configuration frames with high sensitivity and small MTTM would be protected through the application layer by the rapid scrubbing method [19]. For the remaining configuration frames, the proposed multifrequency scrubbing method will be adopted. By the end, the MTTDs of the configuration frames would match their MTTM. Briefly, shown in Fig. 2, the proposed hierarchical scrubbing technique requires a total of five steps as shown in the following:

1) Measure the SEU sensitivity of configuration bits intheimplemented circuit through fault injection [25]–[28].

2) Divide the configuration frames into different priori- ties according to the sensitivity. For priority partition, the SEU sensitivity of the configuration bits should be added to get the SEU sensitivity of the configuration frames because the frame is the smallest unit for oper- ations.

3) Add DWC redundancy detection to the circuit modules corresponding to the configuration frames with high sensitivity. Then, the system can make a rapid scrubbing to these configuration frames by locating the position of the SEU through the DWCdetection.

4) Balance the MTTD of rest configuration frames with their sensitivity by adjusting the scrubbing rates. In this way, we optimize the scrubbing frequency of each frame to minimize the system EMR based on the multifre- quency scrubbing method. There are three steps to ana- lyze the scrubbing. The first step is to analyze the best distributionofconfigurationframes withfixed frequency in one cycle. The next step is to calculate the frequency of each frame to get the smallest EMR. Finally, generate the scrubbing sequence used in real-time according to the frequency of configurationframes.

5) Store the information obtained by two scrubbing meth- ods to the scrubber storage structure inhardware.

The hierarchical scrubbing technique will normally make multifrequency scrubbing on all used frames according to different frequencies. When the DWC redundancy finds an error, the scrubber changes to the rapid scrubbing mode and makes scrubbing detection on the frame with position-aware DWC to repair the SEU as soon as possible. By hierarchical scrubbing,theMTTDcanbetter adapttotherespective MTTM of the frames, thereby reducing the system EMR.

B. *Sensitivity Measurement and PriorityPartition*

To measure the MTTM of the configuration bits, an auto- matic soft error sensitivity measurement platform has been established based onthefaultinjection method.First,copytwo original user circuits. One is a standard circuit and the other is the test circuit. Next, inject the SEU to the specific bit in theconfigurationframe corresponding tothetest circuit.Then, apply the same excitation to the two circuits and compare the circuit output. The test circuit outputs a system error when the test circuit output is different from the standard circuit. The test platform would record the MTTM time between the fault injection of the corresponding bit and the observed system error. After this, the test platform would repair the fault and select the next bit to inject SEU. Finally, repeat the operations of fault injection, output comparison, and MTTM recording just as shown in Fig. 3. For each SEU fault, the platform would test in a fixed period. If no error is observed during the period, it would change tothe next SEU.

Due to the configuration structure limitation of FPGA, the configuration bit MTTM should be aggregated to the con- figuration frame sensitivity. There are several ways to aggre- gate the MTTM. One is to use the maximum bit MTTM to represent theframe sensitivity.Thiswouldresult ininsufficient scrubbing performance when a large number ofconfiguration

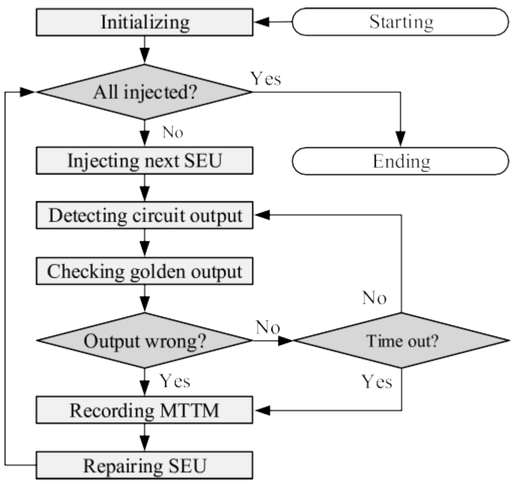


Fig. 3. Procedure of sensitivity measurement.

bits with small MTTM are masked which cannot be repaired in time. The other way is to use the minimum bit MTTM to calculate the frame EMR. However, taking the minimum bit MTTM could result in excessive scrubbing performance and itisawaste oftheconfigurationportbandwidth.

In this article, we use the MTTMs of all the configuration bitstocalculate theEMRs,andthen wesumthem uptogetthe total EMR for a frame. The method for obtaining the frame EMR conforms to the additivity of FIT and it could better reflect the comprehensive situation in the whole configuration frame.

Mainly, the configuration frames could be divided into four priorities. The frames with high sensitivity and small MTTM are the first priority, which could be called critical frames. The second priority is called prioritized essential frames. The MTTM of prioritized essential frames is smaller than its MTTD, and the scrubbing rate would be increased to reduce its MTTD. The third priority is denoted as essential frames or used frames. Except for the prioritized essential frames, the rest frames in essential frames would decrease its scrubbing rate to increase the scrubbing rate of prioritized essential frames.Other frames arecalled unused frames,which would be ignored by ourscrubbing.

C. *RapidScrubbing*

1) Scrubbing Technique: The rapid scrubbing method based onDWCenables aneffective scrubbingasearly aspossible by accurate position information when SEU occurs just as shown in Fig. 4. The circuit shown in Fig. 4(a) is the original circuit without the rapid scrubbing protection. Assume it could be divided into four functional modules, i.e., A, B, C, and D, where modules A and B are more critical to circuit function and thus get protected by DWC. Once the comparator of either module AorBdetects anerror asshowninFig.4(b),itsoutput will be forwarded to the configuration frame address generator in the scrubber, which immediately translates its work mode to a rapid scrubbing mode. At this time, the scrubber would translate the error signal to the configuration frame address using the position information obtained from the analysis of theFPGAconfigurationstructure bytheerror locatingmethod.

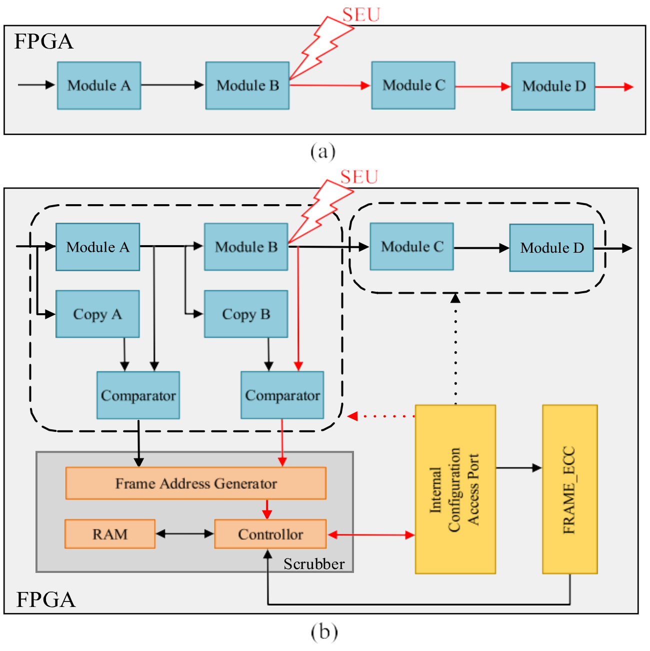


Fig. 4. SEU occurrence in different situations. (a) SEU occurrence of original circuit. (b) SEU occurrence of circuit under rapid scrubbing.

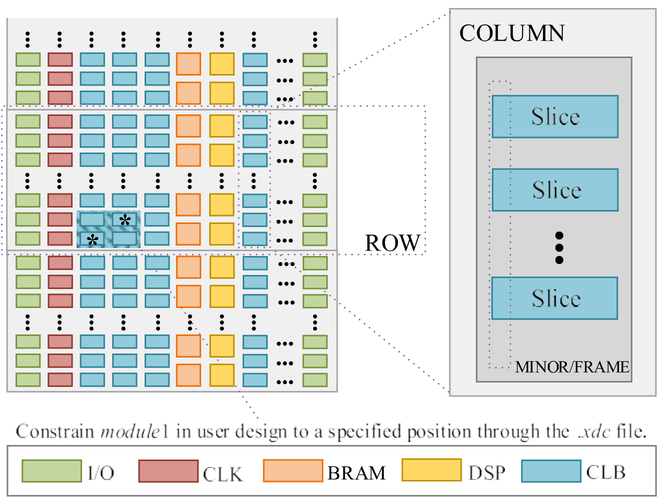

COLUMN
Slice
Slice
Slice
MINOR/FRAME
ROW
Constrain module1 in user design to a specified position through the .xdc file.
I/O
CLK
BRAM
DSP
CLB

Fig. 5. Resource map of user circuit in FPGA.

Then, the scrubber will detect and correct the configuration frames corresponding to the error module.

The position-aware DWC enables rapid scrubbing and makes it different from other scrubbing techniques, which search all the frames or the essential frames. They have to read many frames back for error checking which may result in much longer MTTD. Instead, rapid scrubbing reacts promptly on error detection with DWC and can locate the error within a limited number of frames that improve theMTTD.

Error Locating Method: Generally, it is difficult to locate a fault frame in FPGA configuration memory because of the agnostic of how the application circuit relates to configuration memory. As shown in Fig. 5, the configuration frame address consists of block type, top/bottom address, row address, columnaddress,andminoraddress.

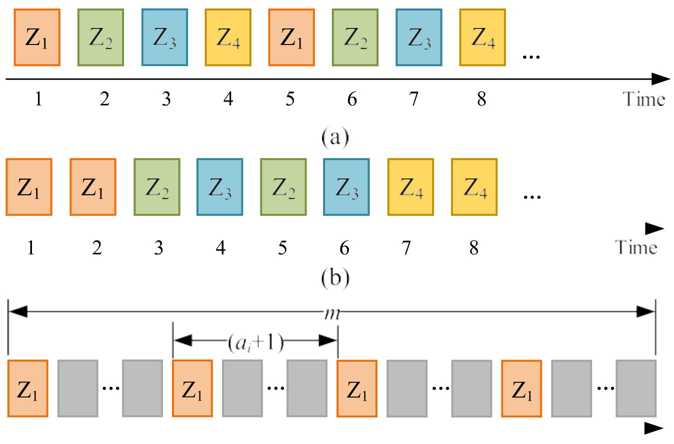

Z1 Z2 Z3 Z4 Z1 Z2 Z3 Z4 ...
1 2 3 4 5 6 7 8 Time
(a)
Z1 Z1 Z2 Z3 Z2 Z3 Z4 Z4 ...
1 2 3 4 5 6 7 8 Time
(b)
m
(ai+1)
Z1 ... Z1 ... Z1 ... Z1 ...

Fig. 6. Influence of different scrubbing frame arrangement under the same scrubbingfrequency. (a)Frame arrangement withevenly distributed.(b)Frame arrangement without evenly distributed. (c) Typical frame arrangement for $Z1$ frame.

Acolumnrepresents a and the distance between adjacent $Z_1$is four frames. If SEU occurs in Z_1at the time of detecting the 1/2/3/4/5/6/7/8 frame, the time to find the SEU is 1/4/3/2/1/4/3/2 frames correspondingly. The average MTTD is 2.5 frames for Z_1. However, as shown in Fig. 6(b), with the same scrubbing frequency, the frame arrangement in Fig. 6(b) is notevenly distributed over time. The average MTTD is 3.625frames for Z_1. Therefore, under a fixed frequency, the different arrangement of the scrubbing sequence has a greater impact on the MTTD. To minimize the MTTD, it is necessary to find the best arrangement of thescrubbing sequence.

Take Z_1, for example, to findthe arrangement condition ofthe minimum MTTD under a fixed frequency. The variables are described asfollows as shown in Fig. 6(c).

1) The distance between two adjacent frames in thescrubbing sequence is 1. The distance increased by oneforseries of identical resources, such as CLB Slice, I/O, CLK, BRAM control logic, and DSP. If a logic circuit is placed in a CLB slice, the corresponding configuration data is placed in the frame with the same column address. Then, we can get the frame address of each circuit module by automatically analyzing the position of themodules.

The error locating method can be divided into four steps. First, read the synthesized circuit, analyze how the circuit modules are connected, and learn how many resources are needed for each module. Second, use the Xilinx design con- straint file (.xdc) to constrain the layout of each circuit module to a specified position as Fig. 5, and then redo the synthesis and implementation. Third, update the constraint file and repeat step 2untilsuccessful implementation.Finally,interpret the position of the resources occupied by modules, and obtain the frame address based on the resource arrangement.

IV. MULTIFREQUENCY SCRUBBING

The multifrequency scrubbing method is to balance theMTTD among the remaining configuration frames to degrade the system EMR. Here is the scrubbing problem description.It is assumed that the scrubber needs to protect n different frames, which are respectively called Z_1, Z_2,…, Z_n. If $Z_i$isdetected for $F_i$times during a scrubbing cycle, the scrubbingfrequency between n frames is called F_1F_2… F_n. Then,$n$frames are arranged to generate a frame sequence in whicheach additional frame in between.

2) If there are $a_i$frames between the i thZ_1andthe

(i 1)thZ_1, the distance is a_i1.

3) a_{F1} means that there are a_{F1} frames between the (F_1)th $Z_1$and the 1stZ_1because the sequence repeats after a whole scrubbingcycle.

4) The sum of the frames ina whole scrubbing cycle is$m$.AsshowninFig.6(c),there isatotalofa_i 1cases when SEU occurs between the i thZ_1andthe(i 1)thZ_1. Todetect theSEU,thenumber of frames areseparately

$1,2,\ldots,a_i$ 1.The sum $S_i$is

S_i=[1+2+· · ·+$(a_i$+1)]. (3)

The average frames detected for SEU in $Z_1$are the quotient between Sum of $S_i$for all $Z_1$and the number of all cases m. m is equal to the number of the frames in a scrubbing cycle. Therefore, the MTTD$_1$ for Z_1could be obtained by multiplying the average frames and one frame detecting time t

$$\text{Sum} = \sum_{i=1}^{F_1} S_i \quad (4)$$

$$m = [(a_1 + 1) + (a_2 + 1) + \cdots + (a_{F_1} + 1)] \quad (5)$$

$$\text{MTTD}_1 = \frac{\text{Sum}}{m} \times t. \quad (6)$$

Here, t represents the fixed time to check a configuration frame just as shown in (2). Also, m is a constant value under a fixed frequency. Therefore, obtaining the minimum of MTTD is the same as minimizing the *Sum* of $S_i$for all cases. According to Cauchy inequality, the minimum of *Sum*is$Z_i$appears $F_i$times. This sequence represents the order ofthe frames detected by the scrubber over time. When the hardware scrubber is working, it reads the frame address from the sequence in turn and writes the correct configuration frame back when SEU is detected.

$$\begin{aligned}\text{Sum} &= \sum_{i=1}^{F_1} S_i = \sum_{i=1}^{F_1}[1 + 2 + \cdots + (a_i + 1)] \\ &= \frac{1}{2} \times \sum_{i=1}^{F_1}[(1 + a_i + 1) \times (a_i + 1)] \\ &= \frac{1}{2} \times \left[\sum_{i=1}^{F_1}(a_i + 1) + \sum_{i=1}^{F_1}(a_i + 1)^2\right] \\ &\geq \frac{1}{2} \times \left(m + \frac{m^2}{F_1}\right). \quad (7)\end{aligned}$$

A. *Optimal Scrubbing Frame Arrangement*

The first problem is the optimal scrubbing frame arrange- ment under a fixed frequency. As shown in Fig. 6(a), the fre-

quency of Z_1, Z_2, Z_3, Z_4 in a scrubbing cycle is $2 : 2 : 2 : 2$

According to the inequality, the minimum of MTTD requires the distance between the same frames in the scrubbing sequence to be equal, which means the frames are evenly distributed in the sequence. This conclusion could be applied to Z_2, Z_3, and all frames. Therefore, the optimal scrubbing sequence should satisfy the condition: all frames are evenly distributed in the scrubbing sequence. This will enable us to minimize the MTTD of all configuration frames under a fixed frequency.

Also, the equation means that if the scrubbing sequence is evenly distributed, the optimal frame MTTD is decided only by the scrubbing frequency. Therefore, the scrubbing problem could be divided into two parts. The first part is to calculate the scrubbing frequency to minimize the overall system EMR and the second part is to generate the evenly distributed scrubbing sequence, which is corresponding to Sections IV-C and IV-D.

A.*Optimal Scrubbing Frequency Calculation*

Through the abovementioned, we have aggregated the con- figuration bits MTTM to obtain the frame EMR. Just as described before, the scrubber needs to protect n differ- ent frames, which are respectively called Z_1, Z_2, ... , Z_n.

Each frame corresponds to an equivalent MTTM denoted as M_1, M_2, ... , M_n. The problem is to calculate the scrubbing frequency F_1F_2... $F_n$according to the EMR and mini- mized MTTD formulas for minimizing the overall EMR. It is assumed that all frames are evenly distributed in the scrubbing sequence.

To minimize system EMR, we propose a gradual replace- ment algorithm for solving the optimal frequency calculation problem. In the scrubbing

sequence, if a frame is replaced with another, the frequency of the replaced frame is reduced and the frequency of the replacing frame is increased. The replacing frame would reduce the frame EMR due to the increased scrubbing frequency while the replaced frame would increase the frame EMR. To obtain a reduction in system EMR, the reduced EMR should be more than the increased EMR. This is the basic principle of the proposed gradual replacement algorithm.

The gradual replacement algorithm has three basic steps to obtain a reduction in system EMR. The first step is to calculate the initial EMR of each frame. Based on the same frequency and obtained MTTM in the previous procedure, the EMR of each frame is calculated. The second step is to calculate the EMR effect of changing the scrubbing frequency. Each time the scrubbing frequency is changed, the replaced frame will be detected one less time in a whole scrubbing cycle. Therefore, the increase in EMR for the replaced frame would be calculated. Similarly, the replacing frame will be detected one more time in the whole scrubbing cycle, and the EMR reduced for replacing the frame would be calculated. The third step is to replace the frame with the least EMR reduced by the frame with the most EMR increased, which improves the overall EMR most.

Based on the frequency after replacement, repeat the operation with updating effect and frame replacement until the maximum EMR reduced is smaller than the minimum EMR increased. By the time, it is no longer possible to reduce the system EMR and the scrubbing frequency is optimal.

The MTTM of the frames is the minimal MTTM of the configuration bits in the frame. The original MTTD depends on the scale of the circuits. in which the improved MTTD is well matched to the frames MTTM. This would help to improve the system EMR at all.

B. Scrubbing Sequence Generation

According to the optimal scrubbing frame arrangement, the more evenly frames are distributed in the scrubbing sequence, the better the system reliability is. However, the evenly distributed sequence may not exist because some scrubbing frequency makes it impossible. In this article, we proposed an algorithm to generate the scrubbing sequence where allframes aredistributed asevenly aspossible.

Similarly, the scrubber needs to protect *n* different frames, which are, respectively, called Z_1, Z_2, … , Z_n. Each frame corresponds to the scrubbing frequency of $F_1 F_2 \ldots$ F_n. AccordingtotheEMRformula,generate ascrubbingsequence for minimizing the overall system EMR. It is best that all frames are evenly distributed in the scrubbing sequence. There is another way to express the sequence generation problem: Every time, the address generator just selects one frame. It is required to distribute as evenly as possible in the sequence for all the frames based on the scrubbingfrequency.

This problem is similar to the load balancing problem. The purpose is to issue multiple requests from the users evenly to each computing resource and improve the utilization [29]. The strict priority (SP) algorithm divides the server queues into SP levels and arranges them from high to low like Fig. 8(a). Only after the highest priority, the next highest priority is arranged [30]. As shown in Fig. 8(b), round-robin (RR) algorithm sets the same priority for all queues and selects the servers in turn [31]. The conventional traversal scrubbing method is just like the RR algorithm in scrubbing sequence generation. In addition, the detecting frames include allused frames andunused frames intheconventional traversal scrubbing method while the selective readback scrubbing and multifrequency scrubbing just include used frames. Weighted round robin (WRR) is improved on the RR algorithm. It can better allocate the servers according to the weight of each server. For those servers with high weight, WRR provides more chances to be selected. At the same time, theservers with low weight are possible to be selected [32], as shown in Fig. 8(c). Based on WRR and the optimal scrubbing frequency, we propose the minimized distance algorithm to generate a multifrequency scrubbingsequence:

1) Initialize *Weight* and distance for each frame. Thetotallength of the scrubbing sequence is *m*, and *m* is the sum of the scrubbing frequency F_i. *i* represents the frame number from 1 to *n* while *n* represents the total number of frames included in the user circuit. Then, the parameter *Weight* of $Z_i$is initialized to m/F_i, which represents the average distance between the two $Z_i$in the scrubbing sequence. The distance variable of $Z_i$is initialized to 0. The signal bit variable is initialized to 0. Signal$_i$ indicates whether the firstoccurrence of $Z_i$in the scrubbing sequence is found. If not found, set Signal$_i$ to 0, else set Signal$_i$ to 1.

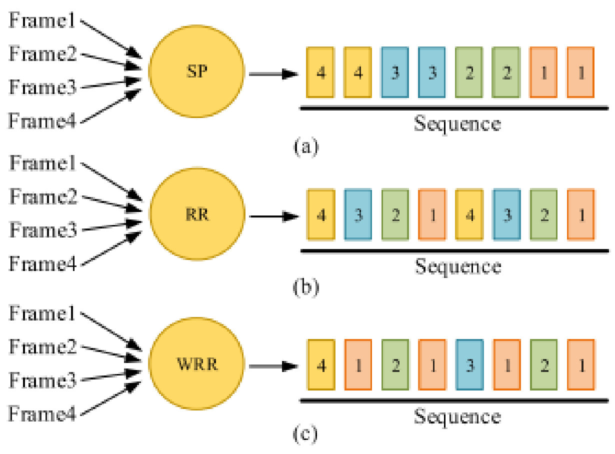


Fig. 8. Algorithms for load balancing problem. (a) Output of SP algorithm.

(b) Output of RR algorithm. (c) Output of WRR algorithm.

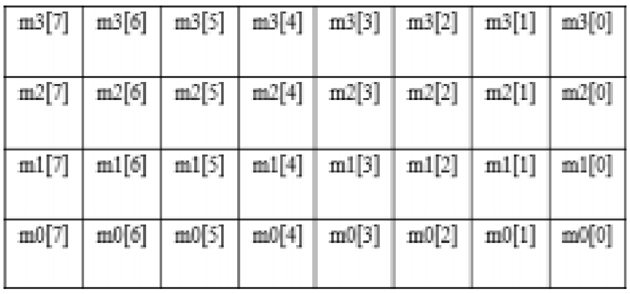
2) Compare the distance of all frames to find the $Z_j$with the smallest distance. If there is more than one frame having the minimum distance, select the frame with the smaller W_i. If there is still more than one frame having the minimum of W_i, select the frame with the smaller frame address. Then, $Z_j$is the next detected frame selected by thescrubber.

3) If the $Z_j$is selected for the first time, set Signal *j* to 1, indicating that the frame has been selected for the first time in the scrubbing sequence. In addition, add the distance of $Z_j$with its W_i.

V. Diagonal Hamming Based Multi-Bit Error Detection

For example, message of 32 bits is accounted in the proposed method. The message is represented in the form m x n matrix. The grouping of the message bits is depicted in Fig.9. The encoder generates the hamming bits and the hamming bits are obtained by grouping the message bits and the hamming bits are calculated with the help of hamming code. The message bits are patterned as shown in Fig. 11. Eight diagonals are considered in this Diagonal Hamming method and each diagonal consists of 4 message bits.

Message bits are grouped as shown in the Fig. 12 in the specified directions.



m3[7]	m3[6]	m3[5]	m3[4]	m3[3]	m3[2]	m3[1]	m3[0]
m2[7]	m2[6]	m2[5]	m2[4]	m2[3]	m2[2]	m2[1]	m2[0]
m1[7]	m1[6]	m1[5]	m1[4]	m1[3]	m1[2]	m1[1]	m1[0]
m0[7]	m0[6]	m0[5]	m0[4]	m0[3]	m0[2]	m0[1]	m0[0]

Fig9. 32-bit message organization (C=8 and R=4)

The first diagonal consists of m3[7], m3[5], m2[6], m1[7], the second diagonal has m3[6], m2[7], m0[1], m1[0], the third diagonal has m0[0], m0[2], m1[1], m2[0], the fourth diagonal has m3[4], m2[5], m1[6], m0[7], the fifth diagonal has m3[3], m2[4], m1[5], m0[6], the sixth diagonal has m3[2], m2[3], m1[4], m0[5], the seventh diagonal has m3[1], m2[2], m1[3], m0[4] and the eight diagonal hasm3[0], m2[1], m1[2], m0[3]. Each one of the *diagonals* has four message bits. For the respective groups, the hamming bits are calculated as shown in Fig. 10. The hamming bits are shown as R1, R2, R3, R4, R5, R6, R7, R8 arrays and these arrays consist of 3 bits.

The hamming bits are calculated as given in equations (1)-:

For the first row:

$R1[1] = m1[7] \oplus m2[6] \oplus m3[7]$; (1)

$R1[2] = m1[7] \oplus m3[5] \oplus m3[7]$; (2)

$R1[3] = m2[6] \oplus m3[5] \oplus m3[7]$; (3)

For the second row:

$R2[1] = m1[0] \oplus m0[1] \oplus m3[6]$; (4)

$R2[2] = m1[0] \oplus m2[7] \oplus m3[6]$; (5)

$R2[3] = m0[1] \oplus m2[7] \oplus m3[6]$; (6)

For the third row:

$R3[1] = m2[0] \oplus m1[1] \oplus m0[0]$; (7)

$R3[2] = m2[0] \oplus m0[2] \oplus m0[0]$; (8)

$R3[3] = m1[1] \oplus m0[2] \oplus m0[0]$; (9)

For the fourth row:

$R4[1] = m0[7] \oplus m1[6] \oplus m3[4]$; (10)

$R4[2] = m0[7] \oplus m2[5] \oplus m3[4]$; (11)

$R4[3] = m1[6] \oplus m2[5] \oplus m3[4]$; (12)

For the fifth row:

$R5[1] = m0[6] \oplus m1[5] \oplus m3[3]$; (13)

$R5[2] = m0[6] \oplus m2[4] \oplus m3[3]$; (14)

$R5[3] = m1[5] \oplus m2[4] \oplus m3[3]$; (15)

For the sixth row:

$R6[1] = m0[5] \oplus m1[4] \oplus m3[2]$; (16)

$R6[2] = m0[5] \oplus m2[3] \oplus m3[2]$; (17)

$R6[3] = m1[4] \oplus m2[3] \oplus m3[2]$; (18)

For the seventh row:

$R7[1] = m0[4] \oplus m1[3] \oplus m3[1]$; (19)

$R7[2] = m0[4] \oplus m2[2] \oplus m3[1]$; (20)

$R7[3] = m1[3] \oplus m2[2] \oplus m3[1]$; (21)

For the eight row:

$R8[1] = m0[3] \oplus m1[2] \oplus m3[0]$; (22)

$R8[2] = m0[3] \oplus m2[1] \oplus m3[0]$; (23)

$R8[3] = m1[2] \oplus m2[1] \oplus m3[0]$; (24)

In the encoder, the hamming bits are calculated for message. We get 24 hamming bits in total for 32-bit message.

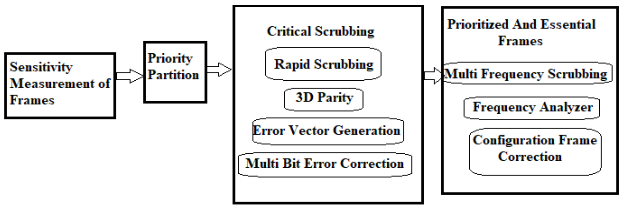

Sensitivity Measurement of Frames
Priority Partition
Critical Scrubbing
Rapid Scrubbing
3D Parity
Error Vector Generation
Multi Bit Error Correction
Prioritized And Essential Frames
Multi Frequency Scrubbing
Frequency Analyzer
Configuration Frame Correction

Fig10 Block diagram for error detection in MBUs

The message bits which are encoded and kept in memory as a matrix as shown in Fig. 10, and are given as input to the decoder. The decoder now segregates message and hamming bits and it recalculates the hamming bits and evaluates syndromebits.

The syndrome bits are evaluated using the equations given (25)-(48) :

For first row

$S1[1] = R1[1] \oplus m1[7] \oplus m2[6] \oplus m3[7]$; (25)

$S1[2] = R1[2] \oplus m1[7] \oplus m3[5] \oplus m3[7]$; (26)

$S1[3] = R1[3] \oplus m2[6] \oplus m3[5] \oplus m3[7]$; (27)

For the second row:

$S2[1] = R2[1] \oplus m1[0] \oplus m0[1] \oplus m3[6]$; (28)

$S2[2] = R2[2] \oplus m1[0] \oplus m2[7] \oplus m3[6]$; (29)

$S2[3] = R2[3] \oplus m0[1] \oplus m2[7] \oplus m3[6]$; (30)

For the third row:

$S3[1] = R3[1] \oplus m2[0] \oplus m1[1] \oplus m0[0]$; (31)

$S3[2] = R3[2] \oplus m2[0] \oplus m0[2] \oplus m0[0]$; (32)

$S3[3] = R3[3] \oplus m1[1] \oplus m0[2] \oplus m0[0]$; (33)

For the fourth row

:

$S4[1] = R4[1] \oplus m0[7] \oplus m1[6] \oplus m3[4]$; (34)

$S4[2] = R4[2] \oplus m0[7] \oplus m2[5] \oplus m3[4]$; (35)

$S4[3] = R4[3] \oplus m1[6] \oplus m2[5] \oplus m3[4]$; (36)

For the fifth row:

$S5[1] = R5[1] \oplus m0[6] \oplus m1[5] \oplus m3[3]$; (37

$S5[2] = R5[2] \oplus m0[6] \oplus m2[4] \oplus m3[3]$; (38)

$S5[3] = R5[3] \oplus m1[5] \oplus m2[4] \oplus m3[3]$; (39)

For the sixth row:

$S6[1] = R6[1] \oplus m0[5] \oplus m1[4] \oplus m3[2]$; (40)

$S6[2] = R6[2] \oplus m0[5] \oplus m2[3] \oplus m3[2]$; (41)

$S6[3] = R6[3] \oplus m1[4] \oplus m2[3] \oplus m3[2]$; (42)

For the seventh row:

$S7[1] = R7[1] \oplus m0[4] \oplus m1[3] \oplus m3[1]$; (43)

$S7[2] = R7[2] \oplus m0[4] \oplus m2[2] \oplus m3[1]$; (44)

$S7[3] = R7[3] \oplus m1[3] \oplus m2[2] \oplus m3[1]$; (45)

For the eight row:

$S8[1] = R8[1] \oplus m0[3] \oplus m1[2] \oplus m3[0]$; (46)

$S8[2] = R8[2] \oplus m0[3] \oplus m2[1] \oplus m3[0]$; (47)

$S8[3] = R8[3] \oplus m1[2] \oplus m2[1] \oplus m3[0]$; (48)

If all the syndrome bits are equal to zero, then it represents that the message bits are not corrupted and if anyone of the syndrome bits in non-zero, then it represents the message bit(s) are corrupted. These corrupted bits need correction so, the message bits are sent to the error correction part. In the correcting of error part, the location of error is identified by doing the following calculations: Suppose if the error is in the third row of the message organization, then the error position is calculated as:

$(S3[3] * (22)) + (S3[2] * (21)) + (S3[0] * (20))$; (49)

After the position is calculated, the error corrector negates the bit corresponding to that position to correct the data bit. This process is done till all the corrupted bits are corrected. Now the output of the decoder will be the form of Fig. 7.

V. SIMULATION RESULTS

The simulation result for the configuration frame and the parity generator is obtained by using VHDL in XILINX ISE 8.1i.

VHDL program for parity generator

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity parity_gen is
port( f  : in std_logic_vector(15 downto 0);
     h  : out std_logic_vector(3 downto 0);
     v  : out std_logic_vector(3 downto 0);
     d  : out std_logic_vector(2 downto 0)
     );
end parity_gen;

architecture df of parity_gen is
begin

process(f)
begin

h(0)<=f(15) xor f(14) xor f(13) xor f(12);
h(1)<=f(11) xor f(10) xor f(9) xor f(8);
h(2)<=f(7) xor f(6) xor f(5) xor f(4);
h(3)<=f(3) xor f(2) xor f(1) xor f(0);


v(0)<=f(15) xor f(11) xor f(7) xor f(3);
v(1)<=f(14) xor f(10) xor f(6) xor f(2);
v(2)<=f(13) xor f(9) xor f(5) xor f(1);
v(3)<=f(12) xor f(8) xor f(4) xor f(0);


d(0)<=f(15) xor f(3) xor f(6) xor f(9) xor f(12) xor f(0);
d(1)<=f(11) xor f(14) xor f(2) xor f(5) xor f(8);
d(2)<=f(7) xor f(10) xor f(13) xor f(1) xor f(4);
end process;
end df;
```

Different circuits have different MUTF improvements.

The proposed multifrequency scrubbing could improve the MUTF with 1.46× in MEM_CTRL while the proposed hierarchical scrubbing improves 1.56×. A similar improvement could be found for the USB_FUNC circuit. However, the proposed scrubbing methods have significantly increased in MUTF for FPU and OPEN_FX especially 146.93× improvement for hierarchical scrubbing in FPU. This diversity is based on the MTTM diversity of the test circuits. The MEM_CTRL and the USB_FUNC circuits have less MTTM than the other two circuits for most observed failures. If there are too many frames with very small MTTM, the proposed gradual replacement algorithm is difficult to balance the MTTD for all frames because most frames need EMR down while only a few frames could provide the acceptable EMR up. On the contrary, the other two circuits have most frames with big MTTM so the proposed scrubbing methods provide excellent MUTF improvements. In summary, the hierarchical scrubbing can improve the MUTF from 1.56× to 146.93×, which is proportional to the MTTF improvement.

As for different scrubbing methods, the MUTF of the proposed scrubbing methods is further improved. The multifrequency scrubbing method could balance the MTTD among the different frames to improve the MUTF.

Besides the multifrequency scrubbing, the hierarchical scrubbing uses the rapid scrubbing at the same time to protect the sensitive configuration frames from two perspectives, which adopted the FPGA application layer and the configuration layer. This method could help make full use of the SEU sensitivity of SEU to improve the MUTF. The proposed hierarchical scrubbing can improve the MUTF from 1.56× to 146.93×.

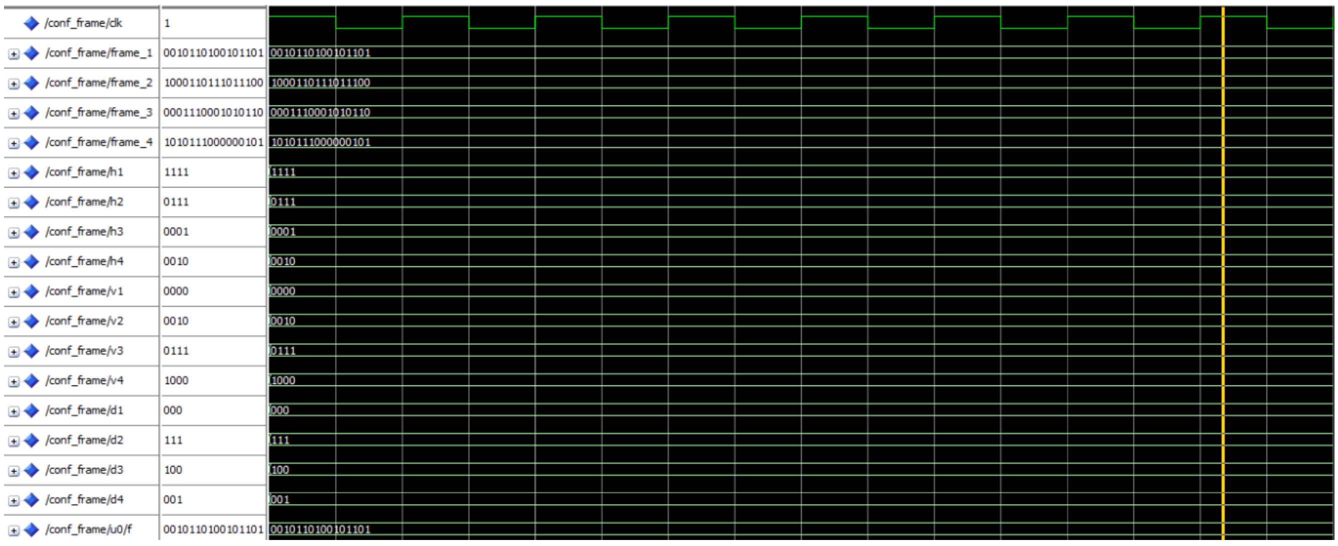
Vhdl program for configuration frames

```
library ieee;
use ieee.std_logic_1164.all;
entity conf_frame is
port(  clk : in std_logic
   );
end conf_frame;

architecture df of conf_frame is

 component parity_gen is
 port( f  : in std_logic_vector(15 downto 0);
       h  : out std_logic_vector(3 downto 0);
       v  : out std_logic_vector(3 downto 0);
       d  : out std_logic_vector(2 downto 0)
       );
 end component;

 signal frame_1:std_logic_vector(15 downto 0):="0010110100101101";
 signal frame_2:std_logic_vector(15 downto 0):="1000110111011100";
 signal frame_3:std_logic_vector(15 downto 0):="0001110001010110";
 signal frame_4:std_logic_vector(15 downto 0):="1010111000000101";
 signal h1,h2,h3,h4,v1,v2,v3,v4:std_logic_vector(3 downto 0);
 signal d1,d2,d3,d4:std_logic_vector(2 downto 0);

begin

u0:parity_gen port map(frame_1,h1,v1,d1);
u1:parity_gen port map(frame_2,h2,v2,d2);
u2:parity_gen port map(frame_3,h3,v3,d3);
u3:parity_gen port map(frame_4,h4,v4,d4);


end df;
```

The most previous scrubbing researches focus more on the improvement of the mean time to repair (MTTR) such as rapid scrubbing and fine-grained fast scrubbing [18]. Fine-grained fast scrubbing can even significantly gain a 62.32% reduction in MTTR with 10.4% area overhead. However, in this article, the hierarchical scrubbing focuses more on the benefit of the MTTR improvement, for example, MUTF. Based on the EMR [24] and circuit MTTM characteristic, we do not have to pay equal attention to all parts of the circuit. Just like we use TMR to partial circuit [4], we could use different scrubbing frequency or methods to different priorities. This is the reason why compared to other scrubbing methods like [19] and [22], the proposed scrubbing methods have a significant improvement in MUTF.

The frame partition result is shown in Fig. 14. The blue part is the used frames but not the prioritized essential frames. It includes 383 frames and occupies 93.87% in OPEN_FX. The orange part denotes the prioritized essential frames but not the critical frames while the yellow part denotes the critical frames. Different from the MEM_CTRL, the OPEN_FX has enough blue frames to provide MTTD decreased, thus even the proposed multi frequency scrubbing method has significant improvement.

If the frames MTTM is smaller than the limitation of the ICAP port speed, the scrubbing method cannot detect and recover the SEUs in such a short time. This is the reason why the hierarchical scrubbing still has some uncover SEUs especially in MEM_CTRL and the USB_FUNC. It is better to add a soft error mitigation structure like TMR to the circuit corresponding to these frames, which extends the MTTM. Another way is to decrease the frequency of the test circuits, which would also extend the MTTM the proposed scrubber only occupies less than 1% of the total FPGA resources. Soft Error Mitigation IP is the SEU mitigation provided by Xilinx which uses the conventional traversal scrubbing method.



```
/conf_frame/clk          1
/conf_frame/frame_1      0010110100101101
/conf_frame/frame_2      1000110111011100
/conf_frame/frame_3      0001110001010110
/conf_frame/frame_4      1010111000000101
/conf_frame/h1           1111
/conf_frame/h2           0111
/conf_frame/h3           0001
/conf_frame/h4           0010
/conf_frame/v1           0000
/conf_frame/v2           0010
/conf_frame/v3           0111
/conf_frame/v4           1000
/conf_frame/d1           000
/conf_frame/d2           111
/conf_frame/d3           100
/conf_frame/d4           001
/conf_frame/u0/f         0010110100101101
```

Fig 13.Configuration Frame Waveform

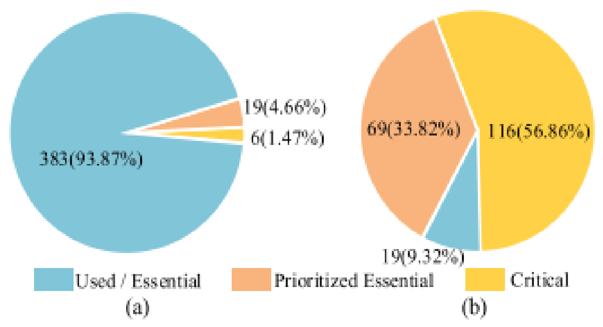


Fig14. Different circuits frames partition according to MTTM. (a) OPEN_FX frames. (b) MEM_CTRL frames.

The IP function here is also set to just detect and correct single-bit errors. Multifrequency scrubbing 1 means the first method which just stores the scrubbing sequence generated by software directly in BRAM while multifrequency scrubbing 2 means generating the sequence in real time by hardware. However, it should be noted that in addition to the overhead of the scrubber, it still has some other resources usage, which will depend on the user circuit.

For rapid scrubbing, it is necessary to increase the overhead associated with user circuit redundancy. For multifrequency scrubbing, extra BRAM is used for storing the Weight if the circuit scale is bigger. For the impact on the circuit frequency, the rapid scrubbing has little impact on the frequency of the original circuit based on the timing report [19]. The reason for the dropping of the circuit frequency is that we reroute the design and use comparators to collect the error vector that may increase the critical path of the original circuits. However, the multifrequency scrubbing method is independent of the original circuits and has no impact on the original circuit performance. The scrubber frequency is decided by the ICAP port with 100 MHz

VI. CONCLUSION

Thisarticle proposes anewhierarchical scrubbingtechnique thatcanmake fulluseoftheSEU and MBU sensitivity.Theself-adaptive MTTD enables an effective scrubbing for different frame priorities based on rapid scrubbing and proposed multifrequency scrubbing, which greatly benefits the circuit reliability by adjusting the scrubbing rate for every configuration frame based on the model.

The reliability of the system from 1.56 to 146.93.

REFERENCES

[1] E. Dupont, M. Nicolaidis, and P. Rohr, “Embedded robustness IPs for transient-error-free ICs,” *IEEE Des. Test. Comput.*, vol. 19, no. 3, pp. 54–68, May2002.

[2] C. Shao, H. Li, J. Fang, and Q. Deng, “An error location and correction method for memory based on data similarity analysis,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 10, pp. 2354–2364, Oct.2019.

[3] R. Parhi, C. H. Kim, and K. K. Parhi, “Fault-tolerant ripple-carry binary adder using partial triple modular redundancy (PTMR),” in *Proc. IEEE Int.Symp.CircuitsSyst.(ISCAS)*,May2015,pp.41–44.

[4] A. T. Sheikh, A. H. El-Maleh, M. E. S. Elrabaa, and S. M. Sait, “A fault tolerance technique for combinational circuits based on selective- transistor redundancy,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 1, pp. 224–237, Jan.2017.

[5] P. K. Samudrala, J. Ramos, and S. Katkoori, “Selective triple modular redundancy (STMR) based single-event upset (SEU) tolerant synthesis for FPGAs,” *IEEE Trans. Nucl. Sci.*, vol. 51, no. 5, pp. 2957–2969, Oct.2004.

[6] X. She and N. Li, “Reducing critical configuration bits via partial TMR for SEU mitigation in FPGAs,” *IEEE Trans. Nucl. Sci.*, vol. 64, no. 10, pp. 2626–2632, Oct.2017.

[7] Y. Li *et al.*, “Feedback-based low-power soft-error-tolerant design for dual-modular redundancy,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 8, pp. 1585–1589, Aug.2018.

[8] S.-H. Kang, H.-W. Park, S. Kim, H. Oh, and S. Ha, “Optimal check- point selection with dual-modular redundancy hardening,” *IEEE Trans. Comput.*, vol. 64, no. 7, pp. 2036–2048, Jul.2015.

[9] R. Giordano, S. Perrella, V. Izzo, G. Milluzzo, and A. Aloisio, “Redundant-configuration scrubbing of SRAM-based FPGAs,” *IEEE Trans.Nucl.Sci.*,vol.64,no.9,pp.2497–2504,Sep.2017.

[10] M. S. Reorda, L. Sterpone, and A. Ullah, “An error-detection and self- repairing method for dynamically and partially reconfigurable systems,” in *Proc. 18TH IEEE Eur. TEST Symp. (ETS)*, May 2013,pp. 1–7.

[11] M. Ebrahimi, P. M. B. Rao, R. Seyyedi, and M. B. Tahoori, “Low- cost multiple bit upset correction in SRAM-based FPGA configuration frames,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 3, pp. 932–943, Mar.2016.

[12] S. Mandal, R. Paul, S. Sau, A. Chakrabarti, and S. Chattopadhyay, “Anovel method forsofterror mitigationinFPGAusingmodified matrix code,”*IEEEEmbedded Syst.Lett.*,vol.8,no.4,pp.65–68,Dec.2016.

[13] A. Stoddard, A. Gruwell, P. Zabriskie, and M. J. Wirthlin, “A hybrid approach to FPGA configuration scrubbing,” *IEEE Trans. Nucl. Sci.*, vol. 64, no. 1, pp. 497–503, Jan.2017.

[14] C. Bolchini, D. Quarta, and M. D. Santambrogio, “SEU mitigation for sram-based FPGAs through dynamic partial reconfiguration,” in *Proc. 17th Great Lakes Symp. VLSI (GLSVLSI)*, 2007, pp.55–60.

[15] M. Vavouras and C.-S. Bouganis, “Area-driven partial

reconfiguration for SEU mitigation on SRAM-based FPGAs," in *Proc. Int. Conf. ReConFigurable Comput. FPGAs (ReConFig)*, Nov. 2016, pp.1–6.

[16] G. L. Nazar, L. P. Santos, and L. Carro, "Accelerated FPGA repair throughshifted scrubbing,"in*Proc.23rdInt.Conf.Field Program.Log. Appl.*, Sep. 2013, pp.1–6.

[17] L. Pereira-Santos, G. L. Nazar, and L. Carro, "Exploring redundancy granularities to repair real-time FPGA-based systems," *Microprocessors Microsyst.*, vol. 51, pp. 264–274, Jun.2017.

[18] G. L. Nazar, L. P. Santos, and L. Carro, "Fine-grained fast field- programmable gate array scrubbing," *IEEE Trans. Very Large Scale Integr.(VLSI)Syst.*,vol.23,no.5,pp.893–904,May2015.

[19] S. Zheng *et al.*, "A rapid scrubbing technique for SEU mitigation on SRAM-based FPGAs," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp.1–5.

[20] I. Herrera-Alzu and M. Lopez-Vallejo, "System design framework and methodology for xilinx virtex FPGA configuration scrubbers," *IEEE Trans. Nucl. Sci.*, vol. 61, no. 1, pp. 619–629, Feb.2014.

[21] S. Fouad, F. Ghaffari, M. E. A. Benkhelifa, and B. Granado, "Context- aware resources placement for SRAM-based FPGA to minimize check- point/recovery overhead," in *Proc. Int. Conf. ReConFigurable Comput. FPGAs (ReConFig)*, Dec. 2014,pp. 1–6.

[22] M. Li, S. Wang, N. Ma, and Y. Peng, "SRAM FPGAs single event upsets detection method based on selective readback," in *Proc. Prognostics Syst.Health Manage.Conf.(PHM-Harbin)*,Jul.2017,pp.1–7.

[23] S.Krishnaswamy,S.M.Plaza,I.L.Markov,andJ.P.Hayes,"Enh ancing design robustness with reliability-aware resynthesis and logic simula- tion," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design*, Nov. 2007, pp.149–154.

[24] H. Asadi and M. B. Tahoori, "Analytical techniques for soft error rate modeling and mitigation of FPGA-based designs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 12, pp. 1320–1331, Dec.2007.

[25] S. Liu *et al.*, "Comparison of the susceptibility to soft errors of SRAM- based FPGAerror correction codes implementations,"*IEEETrans.Nucl. Sci.*, vol. 59, no. 3, pp. 619–624, Jun.2012.

[26] C.-A. Mao, Y. Xie, X. Wei, Y.-Z. Xie, and H. Chen, "FPGA-based fault injection design for16K-pointFFTprocessor,"*J.Eng.*,vol.2019,no.21, pp. 7994–7997, Nov.2019.

[27] N. Jing *et al.*, "Quantitative SEU fault evaluation for SRAM-based FPGA architectures and synthesis algorithms," in *Proc. 21st Int. Conf. Field Program.Log.Appl.*,Sep.2011,pp.282–285.

[28] Y. Xie, H. Chen, Y.-Z. Xie, C.-A. Mao, and B.-Y. Li, "An automated FPGA-based fault injection platform for granularly-pipelined fault tol- erant CORDIC," in *Proc. Int. Conf. Field-Program. Technol. (FPT)*, Dec. 2018, pp.370–373.

[29] N. K. C. Das, M. S. George, and P. Jaya, "Incorporating weighted round robin in honeybee algorithm for enhanced load balancing in cloud environment," in *Proc. Int. Conf. Commun. Signal Process. (ICCSP)*, Apr. 2017, pp.0384–0389.

[30] K. W. Ross and D. D. Yao, "Optimal load balancing and scheduling in a distributed computer system," *J. ACM*, vol. 38, no. 3, pp. 676–689, Jul.1991.

[31] B. Alam, M. N. Doja, and R. Biswas, "Finding time quantum of round robin CPU scheduling algorithm using fuzzy logic," in *Proc. Int. Conf. Comput. Electr. Eng.*, Dec. 2008, pp.795–798.

[32] W. Wang and G. Casale, "Evaluating weighted round robin load balanc- ing for cloud Web services," in *Proc. 16th Int. Symp. Symbolic Numeric AlgorithmsSci.Comput.*,Sep.2014,pp.393–400.

[33] A. M. Keller and M. J. Wirthlin, "Benefits of complementary SEU mitigation for the LEON3 soft processor on SRAM-based FPGAs," *IEEETrans.Nucl.Sci.*,vol.64,no.1,pp.519–528,Jan.2017.

[34] Z. Feng, N. Jing, and L. He, "IPF: In-place X-Filling algorithm for the reliability of modern FPGAs," *IEEE Trans. Very Large Scale Integr. (VLSI)Syst.*,vol.22,no.10,pp.2226–2229,Oct.2014.

[35] L. A. C. Benites *et al.*, "Reliability calculation with respect to functional failures induced byradiationinTMRarmcortex-M0soft-core embedded into SRAM-based FPGA," *IEEE Trans. Nucl. Sci.*, vol. 66, no. 7, pp. 1433–1440, Jul.2019.

[36] L. Sterpone, M. Violante, and S. Rezgui, "An analysis based on fault injection of hardening techniques for SRAM-based FPGAs," *IEEE Trans.Nucl.Sci.*,vol.53,no.4,pp.2054–2059,Aug.2006.

[37] F. Benevenuti and F. L. Kastensmidt, "Comparing exhaustive and random fault injection methods for configuration memory on SRAM- based FPGAs,"in*Proc.IEEELatinAmer.Test Symp.(LATS)*,Mar.2019, pp.1–6.

[38] Guanghui He,Sijie Zheng , and Naifeng Jing "A Hierarchical Scrubbing Technique for SEU Mitigation on SRAM-Based FPGAs"