Low Overhead Internal Scrubbing Technique for Virtex-5 Configuration Upsets

Prasanth C
M. Tech. Student
Department of Electronics and Communication Engineering
SAINTGITS College of Engineering, Kerala, India

Riboy Cheriyan
Associate Professor
Department of Electronics and Communication Engineering
SAINTGITS College of Engineering, Kerala, India

Abstract

Field Programmable Gate Array is an innovative way of implementing hardware designs. Capability of customization after manufacturing of these devices allows to comply with new needs or to change an existing design. This has lead to their wide use in various sectors including safety critical applications and harsh radiation environments. But the main disadvantage of these devices is the susceptibly to radiation events. A radiation event can alter either the configuration memory or the user memory. An error in the configuration memory of an FPGA leads to a transient effect followed by a permanent effect. This may alter the implemented logic and can cause serious problems in mission critical applications. This paper presents a simple and fast method to correct single event upsets occurring in the configuration memory of Virtex-5 FPGAs. A Finite State Machine based controller is used for controlling the scrubbing process. A Syndrome decoder is used to locate the single bit error location inside a frame and they are corrected within a dual port BRAM. Internal Configuration Access Port is used to read the configuration frames and write the corrected frames back to the configuration memory. The system is implemented in Virtex-5 XC5VLX110T device and compared with the existing method for hardware utilization and the time to detect and correct single bit errors.

Keywords: Configuration Memory, Reconfiguration, Scrubbing, Single Event Upsets

I. INTRODUCTION

Field Programmable Gate Arrays (FPGAs) are programmable semiconductor devices, which consists of Configurable Logic Blocks (CLBs) connected through programmable interconnects. These FPGAs can be programmed and reprogrammed to the intended functionalities. These devices require a configuration memory to store the information in the form of a bit-stream, which is used to program the device. Every FPGA relies on the underlying programming technology, which is used to control the programmable switches. There are mainly three programming technologies: SRAM based, anti-fuse based and flash based. Out of these technologies, SRAM based technology is most commonly used, because of the infinite reprogramming capability.

SRAM based FPGAs are increasingly being used for mission critical and reliable systems. But, due to the high integration density of these SRAM cells, these devices are prone to faulty behavior caused by the cosmic rays and artificial radiation. Due to this radiation exposure, either the user memory or configuration memory can be changed, usually in the form of a bit flip. These are usually called as Single Event Upsets (SEUs). A SEU is a type of Single Event Transient (SET) characterized by a change of state in a memory cell caused by highly energized particles (e.g., protons and neutrons) or electro-magnetic radiation striking a sensitive node in a microelectronic device [1]. In space applications, SEU can result in loss of information, functional failure, or loss of control.

SEU has a peculiar effect in FPGAs when a particle hits the user's combinational logic. In an ASIC, the effect of a particle hitting either the combinational or the sequential logic is transient; the only variation is the time duration of the [2]. A fault in the combinational logic is a transient logic pulse in a node that can disappear according to the logic delay and topology. Faults in the sequential logic manifest themselves as bit flips, which will remain in the storage cell until the next load. But, in the case of an SRAM based FPGA, the combinational as well as the sequential logic are implemented by customizable logic cells, based on the bit stream loaded in the configuration memory. So, an SEU affecting the configuration memory changes the implemented logic, which means that it has a transient effect followed by a permanent effect [3]. So, some kind of periodic error correction is very important in highly reliable systems.

The mechanism used to detect and correct SEU should be very fast enough, otherwise the system-error latency would be high. Usually these mechanisms are implemented in the FPGA fabric along with the logic. So, the hardware overhead of error detection and correction mechanism should be as small as possible. This paper describes a SEU detection and correction mechanism which has higher speed of operation and less hardware overhead as compared to the existing systems.

A soft-core processor is a hardware description language (HDL) model of a specific processor (CPU) that can be customized for a given application and synthesized for an ASIC or FPGA target. In many applications, soft-core processors provide several advantages over custom designed processors such as reduced cost, flexibility, platform independence and greater immunity to obsolescence. Nios II, MicroBlaze, PicoBlaze and Xtensa are the leading soft-core processors provided by Altera, Xilinx and Tensilica respectively.

Various methods have been developed to detect and correct the SEU sensitivity of the configuration memory. TMR is a classic fault tolerant method, which repeats the same circuits three times and outputs their values with a majority vote. This method can correct any type of error that occurs in one of the three modules, with few overheads on system delay. However the main drawbacks of TMR are that it needs 3-4 times the area and it cannot recover from an SEU without stopping the device. Duplication with comparison (DWC) reduces one set of TMR devices, but it still has a double area overhead and cannot output correct values on its own when a soft error occurs.

Scrubbing is an alternate mechanism for correcting configuration memory errors, in which periodic access of the configuration memory is used. The errors can be corrected by writing the configuration data either in a periodic way or only when an error is detected. The former method is called as open loop scrubbing and the latter one is closed loop scrubbing. Based on the way in which configuration memory is accessed, scrubbing is of two types: Internal scrubbing and external scrubbing (Berg et.al (2008)). External scrubbing use external configuration ports such as JTAG or SelectMap and it require an external radiation hardened scrubbing controller and a memory to store the configuration bits. On the other hand, internal scrubbing uses internal configuration interface such as Internal Configuration Access Port. The scrubbing is also performed internally.

A scrubbing controller is required to control all these mechanisms. Most of the existing techniques use an embedded microprocessor to control the scrubbing process. But this often leads to higher area overheads. The microprocessor indeed offers some additional options like control or debug through an RS232 interface and fault injection. But, these options are not required in actual recovery process and may even reduce reliability of the system.

Once the single bit error inside a frame is identified and corrected, the frame has to be written back to the memory location. For this, frame address should be specified to frame address register of an FPGA. Conventionally used method for frame address generation is to have separate counters for each of the five fields in the frame address register. This also increases area overhead. This paper describes a SEU detection and correction mechanism based on internal scrubbing of configuration memory, which has higher speed of operation and less hardware overhead as compared to the existing systems.

This paper is organized as follows. Section II describes the motivation in doing such a topic and details the existing methods and techniques in this area. Section III deals with the configuration memory architecture of Virtex-5 FPGA. In section IV, proposed work is explained and the implementation results are given in section V. The last section presents some conclusions and perspectives for future work.

II. MOTIVATION

Traditionally used method for SEU mitigation depends upon hardware redundancy. It can either be Duplication with Comparison (DWC) for error detection or Triple Modular Redundancy (TMR) with majority voting for masking faults. In DWC, the hardware is duplicated and the results provided by these two units are compared to detect the fault. In TMR, the hardware is triplicated and the output of the system is taken using a majority voter. Even if one of these three units fails, the majority voter selects the correct output from the other two units. But the main disadvantage of this method is that the hardware has to be triplicated, which incurs large area overhead. Also, this method is vulnerable to multiple bit upsets, which is becoming more frequent, as the device geometry becoming smaller [4].

The alternate mechanism for correcting errors in the configuration memory is called as scrubbing. In scrubbing, the configuration memory of the device is accessed in a periodic way, and if any errors are found, they are corrected. There are mainly two types of scrubbing: Internal scrubbing and external scrubbing, depending upon the way in which the configuration memory is accessed [5]. External scrubbing use external configuration ports such as JTAG or SelectMap and it require an external radiation hardened scrubbing controller and a memory to store the configuration bits. On the other hand, internal scrubbing uses internal configuration interface such as Internal Configuration Access Port. The scrubbing is also performed internally. For SEU/MBU mitigation, two strategies exist: open-loop scrubbing and closed loop scrubbing [5]. In open-loop scrubbing (blind scrubbing), the configuration data is read back from a radiation hardened golden memory and is written back through a configuration port using dynamic partial reconfiguration. The scrub rate should be faster than the expected SEU rate.
In the case of closed loop scrubbing, the configuration data is read back for SEU/MBU detection. Only if there is an error, correction process is carried out. The simplest method to detect errors in the configuration data is to read back the frames and compare them against the golden counterparts. If there is a difference, the golden copy is used to scrub the faulty frame through configuration port. Similar to open loop scrubbing, this method also requires a golden copy of the configuration data. Another method is to check the CRC computed during readback process. If an error is detected, the error is detected using the Error Correcting Codes. But, in this method only single error per frame can be corrected. Internal error detection and correction mechanism has been implemented in [6], which uses a Picoblaze processor [7] for the scrubbing control, and other circuitry and Block RAM for error correction. It uses separate program memory for the Picoblaze processor and it is not covered by the Hamming bits.

A method to protect reconfigurable soft core processors against configuration memory upsets have been described in [8]. It also uses a Picoblaze processor as the configuration engine and an external radiation hardened golden memory is used to store the configuration bits. When an error is detected and the error location is identified, it is corrected using the golden copy through partial reconfiguration. Readback and configuration is done through ICAP and Frame_ECC primitive is used for the error detection. The error detection mechanism has been replaced with a Hamming decoder in [9].

A method to detect configuration upsets using built-in self test is described in [10]. But, this method can detect only persistent errors and no transient errors. A number of other techniques have been used to correct configuration upsets in Virtex-4 FPGAs [11], [12]. But, there are only limited numbers of techniques available for Virtex-5 FPGAs. So, this paper focuses on error detection and correction mechanism for the single event upsets in the configuration memory of Virtex-5 FPGAs.

III. VIRTEX-5 CONFIGURATION MEMORY

Configuration memory of Virtex-5 FPGA is arranged in frames that are tiled about the devices. Frames are the smallest addressable segments of Virtex-5 configuration memory space and it consists of 41 words of 32 bits each. Each FPGA is divided into a number of rows that cover up the entire fabric area. A single frame covers the height of a memory row. These rows are grouped into two halves, top and bottom. From the center, rows in each half are numbered from 0 to 9. Frames in each row are also grouped into different columns with variable size depending upon the type of column, such as CLB, DSP, BRAM, IOB etc. So, the address of a frame in the configuration memory consists of five fields: block type, top/bottom indicator, row address, major address (column address) and the minor address (frame address inside a column) [13]. All Virtex-5 FPGA commands are executed by performing reads and writes to the appropriate configuration registers.

IV. PROPOSED SYSTEM

The proposed system is completely implemented in CLBs and one block Ram in the FPGA fabric. It consists of a scrubbing controller, a dual port BRAM, a syndrome decoder and Virtex 5 primitives – ICAP and FRAME_ECC [14]. The block diagram of the system is given below.

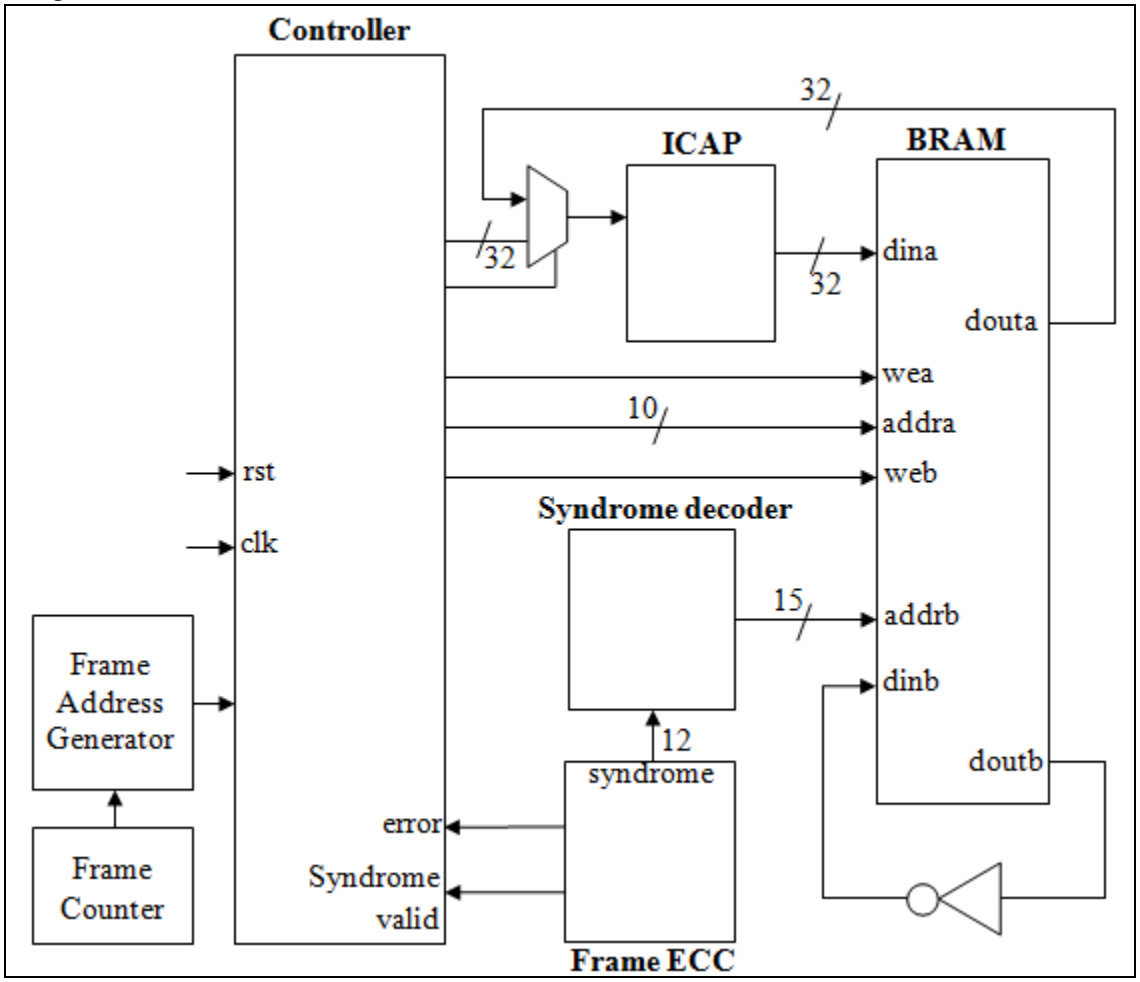

Fig.1: Block diagram of the System

A. Controller:

The scrubbing controller is implemented as a Finite State Machine. This controller initiates readback procedure and all the configuration commands are send to the FPGA through the Internal Configuration Access Port (ICAP).

The scrubbing controller initiates readback by sending appropriate commands in the bit reversed format to the corresponding configuration registers through ICAP. All the commands are sent using Packet 1 header. After the initialization, the required number of frames to be read from the FDRO register is specified as the word count in the packet header, and the starting frame address is specified in the Frame Address Register. For this Packet 2 hearer is used. This will start fetching frames word by word and they are stored in a dual port BRAM.

In order to find out single bit error in a frame, FRAME_ECC primitive of Virtex-5 is used. As each word is read from memory, the FRAME_ECC primitive calculates a syndrome value using all bits in the frame including the ECC bits. A Virtex-5 frame consists of 1312 bits organized as 41 words of 32 bits. Out of these 1312 bits, 12 bits correspond to parity bits and other 1300 bits refer to frame data. The first read out frame will be a pad frame and it is discarded. Only valid configuration frames are stored in memory and error checking is carried out.

At the end of each frame, syndrome_valid signal is asserted for one clock cycle and then we can sample the syndrome bits. If the frame bits are same as the original programmed values, then the syndrome bits are all 0's. The error status is decomposed from the syndrome bits as follows:

Table – 1
Error Status Decomposition

S[11]	*S[10:0]*	*Status*
0	*= 0*	*No error*
1	*≠ 0*	*Single bit error*
1	*= 0*	*Single bit error in last parity bit*
0	*≠ 0*	*Double bit error*

If there is a single bit error, then the error location is given by the syndrome bits 10:0. Double bit error is not correctable using this method.

B. Dual port BRAM:

One of the important components in this method is a dual port Block RAM. Port A and Port B of this dual port BRAM are configured for reads and writes of different width. Both read and write operations are fully synchronous to the supplied clock, but Port A and Port B can operate fully independent and asynchronous to each other accessing the same memory array. Each frame read from the configuration memory using ICAP is stored in this BRAM. So, it is configured as 1024 x 32 bit RAM. This is implemented as one column of 36 Kb BRAM in Virtex-5. Port A of this BRAM is controlled by the scrubbing controller. This controller provides the address, where each word of a frame is to be stored. So, port A is configured for 32 bit reads and writes. But, port B is configured for single bit reads and writes. This port provides the singe bit error correction capability. The address pins of port B are controlled by a syndrome_decoder, which process the syndrome value produced by the FRAME_ECC primitive and locates the position of the error in the entire frame. The output of Port B is inverted and is given to the port input. So, when the single bit error position is given by the syndrome_decoder, that bit is inverted and written back to the same memory location. This port is enabled only when the error is detected.

C. Syndrome Decoder:

When entire words of a frame is read back, the syndrome_valid signal goes high, indicating that the error status can be sampled. In case of a single bit error in the frame data, S[10:0] indicates the flipped address space from 704 to 2047. These addresses correspond to the location of first bit and last bit in the frame respectively. In order to convert the syndrome value to the actual range of the frame bits i.e. 0 to 1312, some additional logic should be incorporated. For this, following equation is used:

$$bit\_index = \{S[10:5] - 6'd22 - S[10], S[4:0]\} \quad (1)$$

This is equal to subtracting 704 decimal, if the syndrome value is less than 1024 decimal or subtracting 736 decimal, if it is greater than 1024. If S[10:0] is 0 or a power of two, then the error is in the parity bits. Parity bits are stored in locations from 640-651.

Decoding of the syndrome value is performed by the syndrome_decoder block. The bit_index obtained from syndrome_decoder is given as the address of Port B. at the same time, scrubbing controller asserts write_enable signal for Port B. Hence the faulty bit in the frame is flipped and then entire frame is written back to the frame address at which the error was detected. At this point, the multiplexer select pins are so adjusted that ICAP receives input from the Port A output. When the entire frame is written back to the appropriate location, readback process is performed again from the next frame address, up to the last frame and the entire process is continued.

D. Frame Address Generator:

After the single bit error in the frame is corrected, it has to be written back to the configuration memory, for this we need to know, from which address the faulty frame was read. As already mentioned, the address of a frame consists of five fields: block type, top/bottom indicator, row address, major address and the minor address. In order to write back the frame to the correct memory location, all these five fields has to to be correctly generated. A single PGA consists of several numbers of frames. For example, a XC4VLX110T FPGA consists of 23712 frames. But the address space is not continuous from 0 to 23711. The address of each frame depends upon those five fields. Conventionally used method for frame address generation is to have five different counters, one for each filed in the frame address generator. But, this method requires higher device utilization, as it employs five counters. And also the different types of blocks are arranged in different manner in different FPGAs. So, it is very difficult to decide the count value for each of these counters. This makes thee frame address generation a complex one.
In the proposed system, a single counter is implemented, which counts the frames , as it read back from the configuration memory. When a frame is identified to be faulty, this frame number can be used to generate the correct frame address using an algorithm. For this, knowledge about the arrangement of frames in the configuration memory of an FPGA is important. In order to understand the total number of frames and the arrangement of the frame in the fabric area, Rapid Smith tool is used. From the tool, the 23712 frames of XC5VLX110T is known to be arranged in 8 rows, 4 in each half. The number of frames in each roe is same. So, it is clear that a single row consists of 2964 frames. Each of these rows is divided into 66 columns. The number of frames in a particular type of column is known, but the arrangement of these columns is usually unknown and is provided by the RapidSmith tool. Based on these information, an algorithms have been developed, which can convert the frame number to the corresponding frame address. The main advantage of this technique is that, it requires only one counter, instead of the five in conventionally used techniques.

V. RESULTS AND DISCUSSIONS

The proposed system for correcting single event configuration upsets in Virtex-5 devices is implemented in XC5VLX110T device and the functioning of various components have been simulated. The simulation results of various components are shown below.

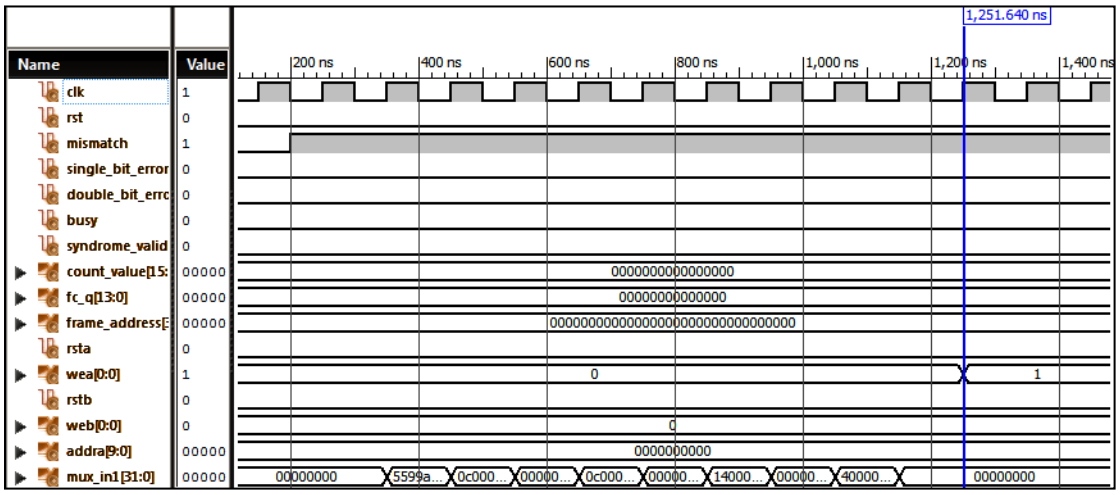

Fig. 2: Controller

Figure 2 shows the operation of a controller. When there is a mismatch, it starts sending commands in the bit swapped format to the appropriate configuration registers. For this, ICAP control signal are also generated accordingly. After the initialization of readback process, each frame will be read from the memory through ICAP and these frames are stored in the memory. So, the port A write enable signal of the dual port BRAM is also asserted. Before changing the ICAP from write mode to read mode, the ICAP select pin should be deasserted, otherwise an abort operation will occur.

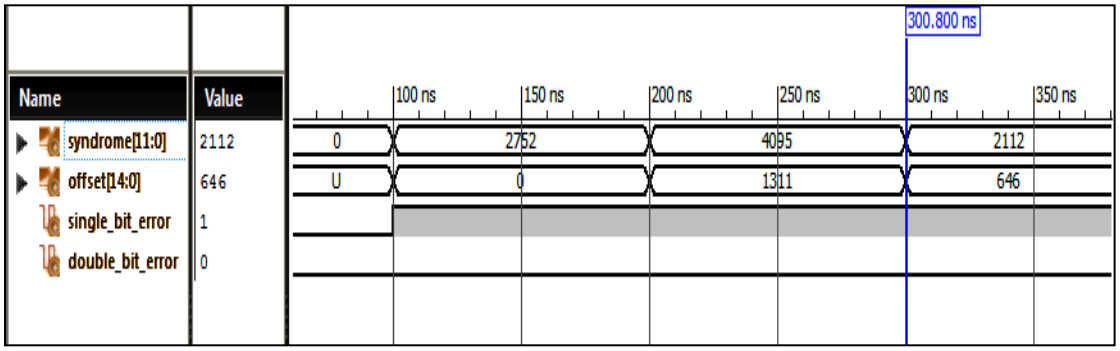

Fig. 3: Syndrome Decoder

Figure 3 shows the operation of syndrome_decoder. After a single frame is readback completely and they are stored in the memory, the FRAME_ECC primitive of Virtex-5 provides a syndrome_valid signal, indicating that we can check the value of 12 bit syndrome signal, based on which we can determine whether the frame is faulty or not. In the case off a single bit error, this syndrome signal provides the location of the error, in the range 704 to 2047. But, we are storing a frame in the memory from 0 to 1311. So, the syndrome decoder takes the 12 bit syndrome signal as the input and converts this syndrome signal to an offset value in the range 0 to 1311. This syndrome decoder also outputs whether the frame contains single bit error or double bit error. Even in the case of single bit error in the parity bits, the syndrome decoder correctly provides the location of the error.

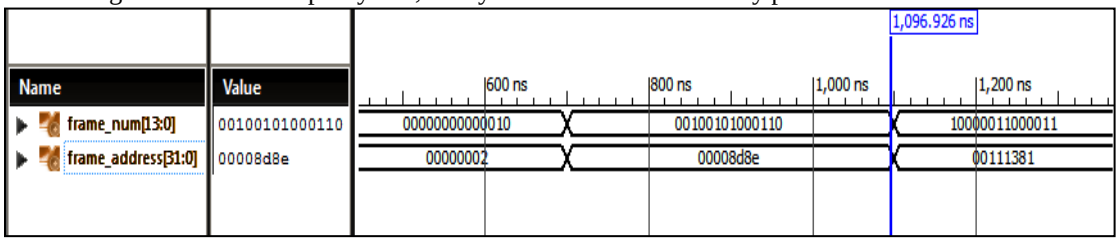

Fig. 4: Frame Address Generator

Figure 4 shows the operation of the frame address generator. When the faulty bit position is identified and it is corrected, the entire frame must be written back to the corresponding memory location. For this, we should know the correct address from which the frame was read. The frame address generator takes the output of the frame counter as the input and converts this frame number to the frame address by generating all the five fields included in the frame address.

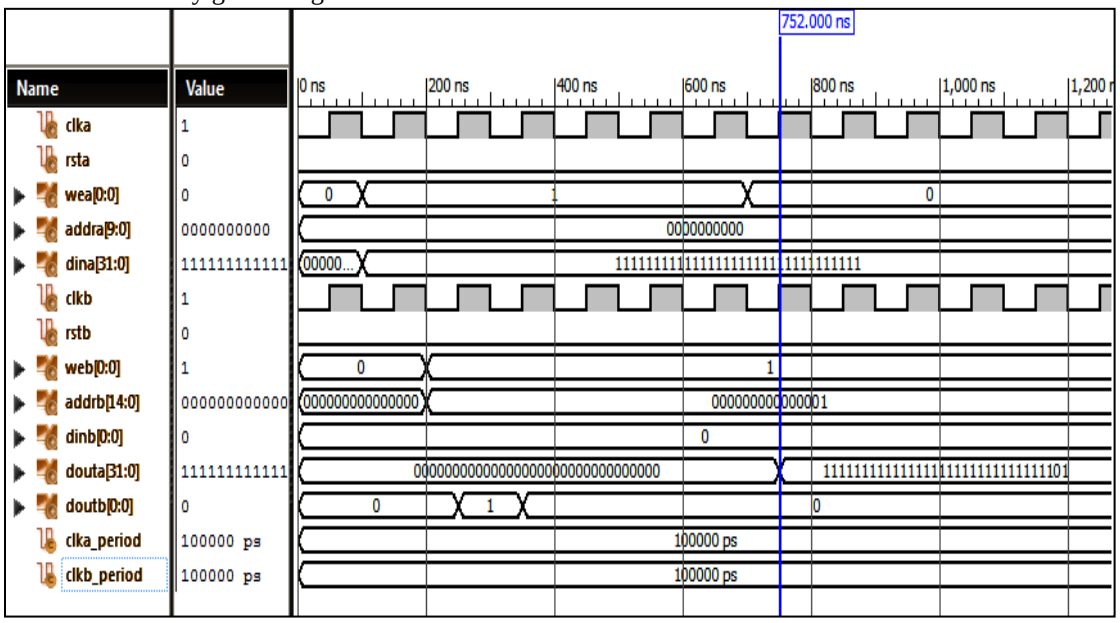

Fig. 5: Dual port BRAM

Figure 5 shows the operation of the dual port BRAM. Port-A write enable signal is asserted to store each frame in the memory. The 41 words are stored in consecutive memory locations. Address for this port-A is given by the controller. When a single bit error occurs, Port-B write enable signal is also asserted to correct that error. The exact location of this single bit error is provided by the syndrome decoder. Ehen this address is given and Port-B write enable signal is asserted, the single bit error in that

location will be corrected by inverting the output of Port-B and connecting it to the input port. Then again the Port-B write enable signal is de-asserted and the frame is written back to the configuration memory.

The main advantage of this system is that it has less hardware utilization, as compared to the existing methods. The time it takes to detect and correct a single bit error is also less compared to the other methods. The important criteria for an error detection and correction mechanism that is to be implemented in FPGA along with the target application is less hardware utilization and high speed of operation.

The central component of the system is scrubbing controller, which is implemented using a Picoblaze processor in the other systems. It takes 44 slices, 84 flip-flop registers and 1 BRAM. This embedded processor indeed provides certain advantages like control or debug through RS232 interface and fault injection. But these are not actually required in error recovery and they result in large hardware overhead. But in this method, scrubbing controller is implemented as an FSM, where the configuration commands are also included in it. It takes only 23 slices, 43 flip-flop registers and no BRAM.

The Picoblaze processor is an 8-bit processor, which means the reads and writes to configuration memory is 8-bit. All the configuration registers in Virtex-5 are 32-bit wide. Hence four clock cycles are needed to read or write a word to configuration registers, which makes the error detection and correction process slow. In the proposed system, all the memory operations are 32 bit wide, so error can be detected and corrected at a much faster rate.

This system does not require a golden copy of the configuration bit stream for single error correction. In the already existing methods, once the error is detected and located, that word is corrected by accessing the golden copy of the bit stream. But in the proposed system, when the error is detected and located, it is corrected in just one clock cycle by inverting the corresponding bit, by providing the faulty bit position to the Port B address pins. So, no additional hardware is needed to correct the single bit error, other than an inverter. Table III shows the hardware utilization comparison of the proposed scheme and reference scheme implemented in Virtex5 XC5VLX110T device with package F3316 and speed grade -1.

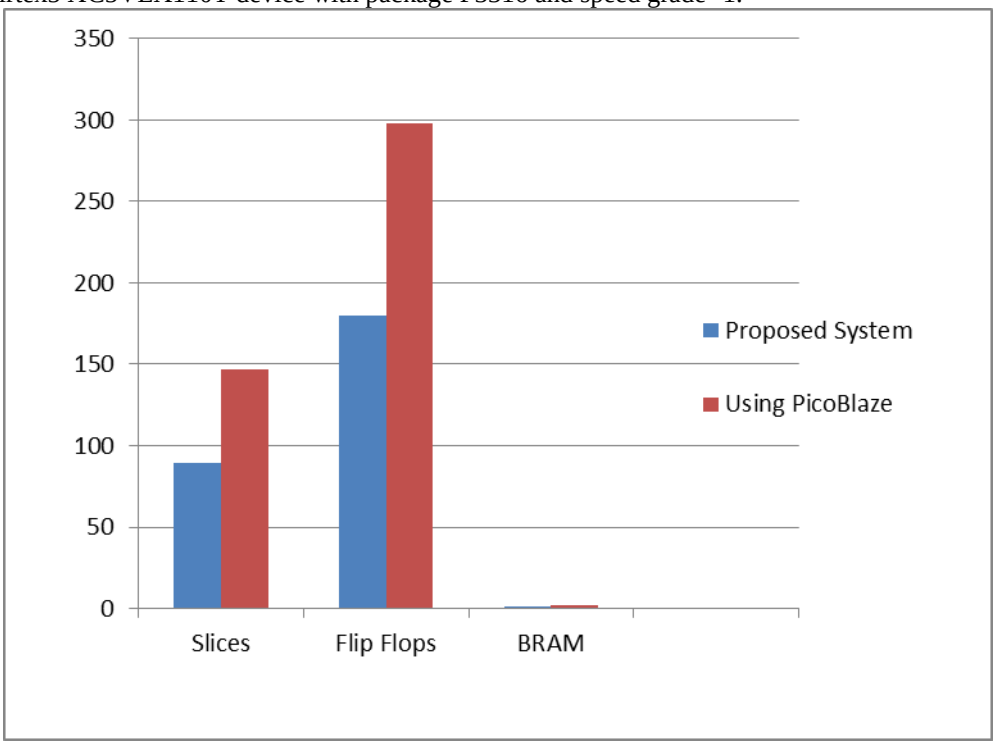

Fig. 6: Implememtation Results For Virtex-5 XC5VLX110T device

From the graph, it is clear that the hardware utilization of the proposed system is less compared to the existing system. In order to make the system fault tolerant, triple modular redundancy is used for the components, except the dual port BRAM. This BRAM is protected with built in ECC.

VI. CONCLUSION

A mechanism for the correction of single bit errors and detection of double bit errors in the configuration memory of Virtex-5 FPGA has been developed. The design has less hardware utilization, when compared to the existing methods using Picoblaze processors. The design is also able to detect and correct single bit errors, faster than that of previous methods. The internal scrubbing controller is implemented using an FSM and a dual port BRAM is used to buffer the readback frame data and to correct single bit errors. Readback and configuration are carried through ICAP and error detection is achieved through FRAME_ECC primitive of Virtex-5. The system can now correct only single bit errors and detect double bit errors. In future, the system can be made modified to correct multiple bit upsets also.

REFERENCES

[1] R. Baumann, "Radiation-induced soft errors in advanced semiconductor technologies," IEEE Trans. Device Mater. Rel., vol. 5, no. 3, pp. 305–315, Sep. 2005.
[2] B. Dutton and C. Stroud, "Single Event Upset Detection and Correction in Virtex-4 and Virtex-5 FPGAs," Proc. ISCA 24th Int. Conf. Computers and Their Applications (CATA 2009), pp. 57-62, Apr. 2009.
[3] U. Legat, A. Biasizzo, and F. Novak, "SEU Recovery Mechanism for SRAM-Based FPGAs", IEEE Transactions on Nuclear Science, voll. 59, no. 5, pp. 2562-2571, Oct 2012.
[4] C. Carmichael, "Triple module redundancy design techniques for Virtex FPGAs," Xilinx Application Notes XAPP197, 2006.
[5] M. Berg et.al. "Effectiveness of internal versus external SEU scrubbing mitigation strategies in a Xilinx FPGA: Design, test, and analysis." IEEE Transactions on Nuclear Science 4, no. 55 (2008): 2259-2266.
[6] K. Chapman, "SEU strategies for Virtex-5 devices," Xilinx Appplication Notes XAPP864 (v2.0), 2010.
[7] "PicoBlaze 8-bit Embedded Microcontroller User Guide," UG129 (v1.1.2), Xilinx Inc., 2008.
[8] H M Pahm, S Pillement, S J. Piestrak "Low Overhead Fault Tolerance Technique for a Dynamically Reconfigurable Soft-core Processor", IEEE Trans. Computers, vol 62, no. 6, June 2013.
[9] M. Safarulla, and K Manilal, "Design of Soft error tolerance technique for FPGA based soft core processors." In Advanced Communication Control and Computing Technologies (ICACCCT), 2014 International Conference on, pp. 1036-1040. IEEE, 2014.
[10] B. Dutton and C. Stroud. "Built-In Self-Test of Embedded SEU Detection Cores in Virtex-4 and Virtex-5 FPGAs." In ESA, pp. 149-155. 2009.
[11] L. Jones, "Single Event Upset (SEU) Detection and Correction Using Virtex-4 Devices," XAPP714 (v 1.5), Xilinx Inc., 2007.
[12] C. Carmichael and C. W. Tseng, "Correcting single-event upsets in Virtex-4 FPGA configuration memory," Xilinx Application Notes XAPP1088 (vl.0), 2009
[13] "Virtex-5 FPGA Configuration User Guide," UG191 (v3.2), Xilinx Inc., 2008.
[14] "Virtex-5 Libraries Guide for HDL Designs", ISE 10.1, Xilinx Inc., 2008.