Hybrid Scrubber of SEM and Picoblaze for FPGA on COMET Read-out Electronics

Eitaro Hamada, Youichi Igarashi, and Kazuki Ueno

***Abstract*—When operating FPGAs in a radiation environment, SEUs induced in the configuration memory can alter the functionality of the firmware. This alteration can disrupt the correct operation of an FPGA. Even with a typical SEU mitigation design incorporated into the FPGA, unrecoverable errors can still occur, which can only be corrected by re-downloading FPGA firmware. The majority of unrecoverable errors are caused by multi-bit upsets. In this study, we developed Hybrid Scrubber Design that can correct these multi-bit upsets. The Hybrid Scrubber Design consists of the AMD Soft Error Mitigation (SEM) and the AMD microprocessor (Picoblaze). When a single-bit upset occurs, the SEM within the FPGA corrects it in a short time. The FPGA communicates with an external computer only when a multi-bit upset occurs, and then the Picoblaze corrects it. We incorporated the Hybrid Scrubber Design into the FPGA on the readout electronics for the COMET experiment. We conducted neutron irradiation tests and measured unrecoverable error rate. Compared to incorporating the SEM, which is a typical SEU mitigation design, the Hybrid Scrubber Design reduced unrecoverable error rate by 80 percent.**

***Index Terms*—COMET, SEU, FPGA, Hybrid, Neutron, Muon**

I. Introduction

SRAM-BASED Field Programmable Gate Arrays (FPGAs) have often been adopted for front-end electronics in high-energy experiments due to advantages such as programmability, low-price, and high-performance. Many FPGAs in front-end electronics operate in radiation environments. The most critical issue of these FPGAs is single-event effects (SEEs), specifically single-event upsets (SEUs) within the configuration memory. SEUs may alter the functionality of the firmware and disrupt the correct operation of the FPGA. AMD Inc. provides the Soft Error Mitigation (SEM) IP core for detecting and correcting SEU errors [1]. In this study, we define the SEU mitigation module of the SEM as the 'SEM Design'. Many experiments adopt the SEM Design because it can be easily implemented in FPGAs. However, when a multi-bit upset (MBU) in the configuration frame occurs, the SEM can detect the issue but may not be able to correct it. In such cases, the SEM process can be halted, leading to soft errors that cannot be corrected. There are other causes of soft errors similar to this. For example, when SEUs occur in critical circuits such as clock generators or the SEM, the FPGA can stop, and these errors cannot be corrected. Another potential error is SEUs occurring in flip-flops or block RAM, which are not protected by the SEM. We define these soft errors as unrecoverable errors. Unrecoverable errors cannot be repaired without re-downloading the FPGA firmware, which generally takes several tens of seconds of dead time. While there are multiple causes of unrecoverable errors, most of them are due to MBUs. In this study, we developed a new SEU mitigation design capable of correcting not only single-bit upsets (SBUs) but also MBUs. Because it can correct MBUs, which are a major cause of unrecoverable errors, it is expected to reduce the dead time associated with firmware downloads. This design comprises the SEM and the Picoblaze (AMD soft microprocessor) [2]. The SEM corrects SBUs. The FPGA communicates with an external Personal Computer (PC) to receive the addresses of upset bits in the configuration memory only when the SEM encounters an MBU that it cannot correct. Subsequently, the Picoblaze restores the MBU to its original state. In this study, we define this design as the 'Hybrid Scrubber Design'.

The shorter the time it takes to correct an upset after an SEU occurs, the higher the reliability of the FPGA, as the upset can be restored to its original state before system errors occur [3]. One of the advantages of the Hybrid Scrubber Design is that the correction process for SBUs is the same as that of the SEM Design. Because most SEUs in the FPGA are SBUs, the Hybrid Scrubber Design does not increase the time required to correct them compared to the SEM Design. The details of the correction time are explained in Section VI.

Various methods have been developed to correct MBUs in a configuration frame. Methods such as blind scrubbing and readback scrubbing have been explored [4], [5]. These methods require reference memory that stores the original configuration data (golden memory). Because a golden memory is susceptible to potential damage from neutron or gamma-ray radiation, these methods may not be applicable to high-energy experiments. Triple-module redundancy and double-module redundancy are also used as methods to correct MBUs. These methods need to duplicate identical modules within an FPGA [6]. Unless an FPGA has sufficient resources, adopting these methods can be difficult. However, our Hybrid Scrubber Design does not require golden memory. The resources of both the SEM and the Picoblaze are very small. For example, they account for less than one percent of the total resources for Artix-7 XC7A200T-2FBG676C.

We incorporated the Hybrid Scrubber Design into the FPGA for the readout electronics of the COMET (The COherent

This Manuscript was submitted on June 1, 2024.;

Eitaro Hamada, Youichi Igarashi are with Institute of Particle and Nuclear Studies, High Energy Accelerator Research Organization(KEK), 1-1 Oho, Tsukuba, Ibaraki 305-0801, Japan (e-mail: ehamada@post.kek.jp).

Kazuki Ueno are with the Department of Physics, Osaka University, Osaka 565-0871, Japan

This work was supported by World Premier International Research CenterInitiative (WPI), MEXT, Japan.

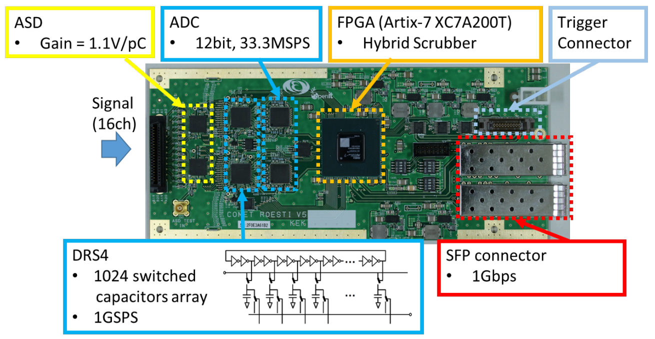

Fig. 1. Photograph and Description of ROESTI.

Muon to Electron Transition) experiment. This is discussed in Section II. To understand the structure of our Hybrid Scrubber Design, it is necessary to understand the SEM and the Picoblaze. Their structures are explained in Sections III and IV, followed by an explanation of the structure of the Hybrid Scrubber Design.

II. COMET Experiment and ROESTI

The COMET experiment at J-PARC aims to search for the muon to electron (μ–e conversion) conversion process in a muonic atom. This process violates charged lepton flavor and is highly suppressed in the Standard Model (SM) including neutrino oscillation, with a predicted branching ratio of less than $\mathcal{O}(10^{-54})$. However, theoretical models beyond the SM predict a branching ratio of $\mathcal{O}(10^{-15})$ for COMET Phase-I [7] and $\mathcal{O}(10^{-17})$ for COMET Phase-II. Therefore, the discovery of μ–e conversion would provide clear evidence for new physics. To suppress background and achieve the goal sensitivity, we adopt a straw tube tracker for the electron detector [8]. We have developed a readout electronics board called ROESTI, which precisely reads the signal from the detector [9]. The ROESTI consists of 16-channel signal input connector, ASD [10], DRS4 [11], ADC (AD9637, Analog Devices), FPGA (XC7A200T-2FBG676C, AMD Inc.), trigger connector, and Small Form-Factor Pluggable (SFP) connector. Figure 1 shows a photograph and a description of the ROESTI. The ASD amplifies and shapes the detector signal. The DRS4 and the ADC digitize analog signals with high-speed and high-accuracy.

To prevent degradation of the detector signal, the ROESTI must be located near the detector. According to simulations, the total neutron fluence in a 150-day physics measurement is estimated to be 1×10^{12} neutrons/cm^2, 1 MeV equivalent, with a safety factor of 5–10 in the regions where the ROESTIs are installed in the COMET Phase-I [12].

Before incorporating the Hybrid Scrubber Design, the FPGA in the ROESTI adopted the SEM Design. Fig. 2 shows a diagram of the FPGA design of the ROESTI at that time, which displays only parts related to the SEU mitigation. The Data I/F module receives digitized data from the ADC. Subsequently, waveform data is created and then sent to the Network I/F. The Network I/F incorporates SiTCP, a hardware-based TCP and UDP processor for Gigabit Ethernet [13]. The SiTCP sends the waveform data to an external PC over TCP/IP.

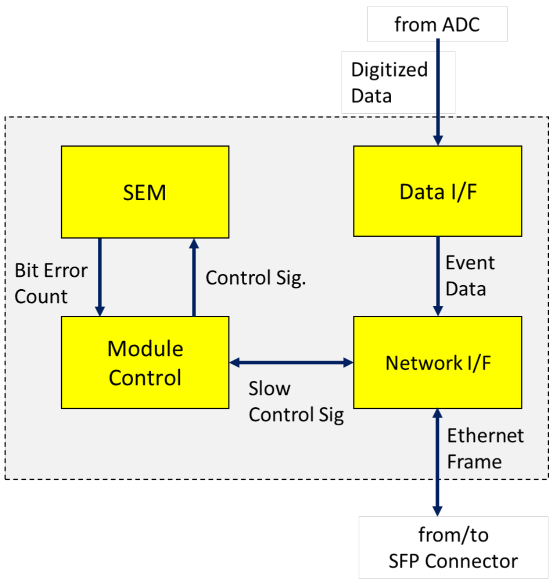

Fig. 2. Diagram of conventional design for ROESTI.

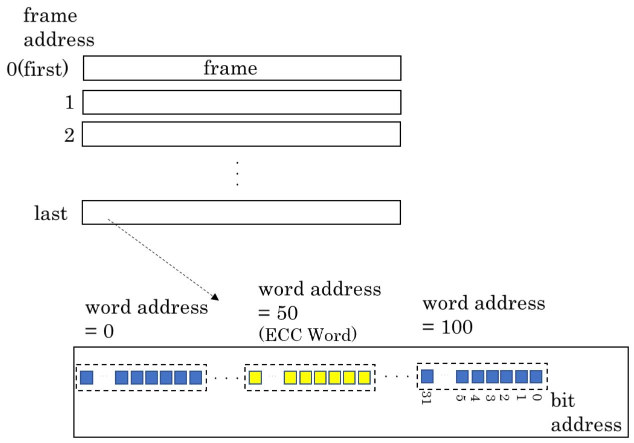

Fig. 3. Structure of AMD 7-series configuration memory.

The SEU counts are sent from the SEM to the Module Control and stored in a register. The value of that register can be sent to the external PC over UDP/IP through the Network I/F. Control signals necessary for controlling the SEM can also be sent over UDP/IP.

III. Structure of SEM

The configuration memory is organized as an array of frames [14]. Each frame in the AMD 7-series FPGA is assigned a unique frame address and configured with 3232 bits (equal to 101 words of 32 bits each), as shown in Fig. 3. Each word and bit is assigned a word address or a bit address. Each frame is protected by an Error Correction Code (ECC), and the entire array of frames is protected by a Readback Cyclic-Redundancy Check (CRC).

Each frame consists of 101 words; one of which is allocated as an ECC word that assists in the detection and correction of errors within the configuration frame. When the frame is read, an ECC syndrome is calculated from the configuration memory data that includes the ECC word. In this calculation,

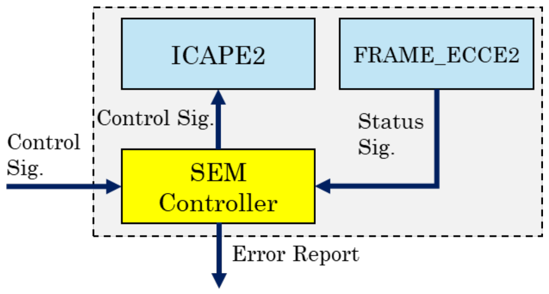

Fig. 4. Diagram of SEM.

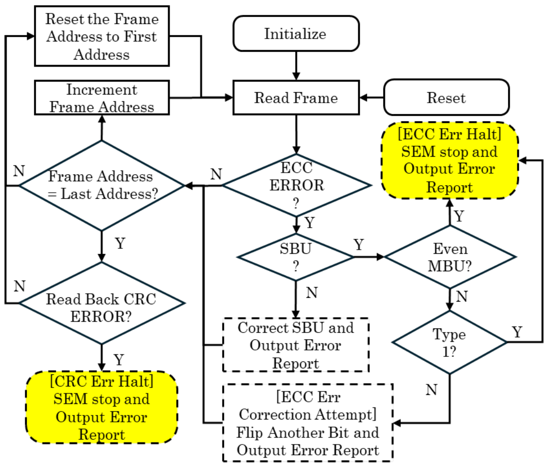
Reset the Frame Address to First Address
Initialize
Increment Frame Address
Read Frame
Reset
[ECC Err Halt] SEM stop and Output Error Report
Frame Address = Last Address?
ECC ERROR ?
SBU ?
Even MBU?
Read Back CRC ERROR?
Correct SBU and Output Error Report
Type 1?
[CRC Err Halt] SEM stop and Output Error Report
[ECC Err Correction Attempt] Flip Another Bit and Output Error Report
N
Y

Fig. 5. Flowchart of error detection and correction processes by SEM. The yellow squares indicate where the error detection and correction processes stop. The dashed squares indicate where the SEM outputs Error Report.

a Single-Error Correction, Double-Error Detection (SECDED) Hamming code is used.

When reading each frame, the CRC code is also calculated. After passing through the frame with the final frame address, the precomputed 32-bit CRC checksum, which serves as a reference value, is compared with the obtained CRC checksum. If they differ, an error is detected, although the address of the frame in which the error occurs is unknown.

Fig. 4 shows a block diagram of the SEM. The status signals of the Frame ECC and the Readback CRC are output through the FRAME_ECCE2 primitive and sent to the SEM controller. The status signals include ECCERROR and CRCERROR, which are asserted when an ECC or CRC error is detected, respectively. In addition, the status signals include the address indicating where the error occurred. The ICAPE2 is an FPGA primitive that provides read and write access to registers within configuration memory. When an ECC or CRC error occurs, the SEM controller outputs the Error Report, which includes the frame address of the frame where the error occurred. If the error can be corrected, the SEM controller sends control signals to the ICAPE2 to execute the partial reconfiguration and correct the error.

Fig. 5 shows a flowchart of the error detection and correction processes by using the SEM. Table I lists the number of upset bits in one frame, detected errors, and final state. After initialization or reset, the error detection and correction processes begin from the frame with an address of 0. Each frame is read, and the presence of an ECC error is checked. Once the frame with the last address is read, the presence of a CRC error is checked. If no errors are detected, then the error detection and correction processes return to the beginning. When a frame with an SBU is read, an ECC error is detected. The ECCERROR signal on the FRAME_ECCE2 is asserted, and the frame, word, and bit addresses are sent to the SEM controller. After receiving them, the SEM controller sends a control signal to the ICAPE2 to correct the SBU, and the ECC error is corrected. When a frame with an MBU is read, the error detection and correction processes vary depending on whether the number of upset bits is even or odd. If a frame with an even MBU is read, an ECC error is detected. The ECCERROR signal on the FRAME_ECCE2 is asserted, and the frame address is sent to the SEM controller. The SECDED code cannot correct this error. The SEM state transitions to 'ECC Err Halt' and the error detection and correction processes stop. When a frame with an odd MBU is read, an ECC error is detected. The subsequent processes can be categorized into two types, Type 1 and Type 2. The calculation of the ECC syndrome determines which type is selected [15]. In the case of Type 1, the error correction and detection processes are the same as when an even MBU occurs. The SEM state transitions to 'ECC Err Halt', and the error detection and correction processes stop. In the case of Type 2, the situation is similar to the error detection and correction processes of an SBU. However, in the 'ECC Err Correction Attempt' state, the SEM attempts to correct the ECC error and fails. Here, the frame address of the frame where the error occurred is sent from FRAME_ECCE2 to the SEM controller. Simultaneously, the wrong word address and bit address are sent to the SEM controller. Because the SEM controller sends a control signal to flip that bit, it leads to an increase of one bit error. After the last frame is read, a CRC error occurs and the CRCERROR signal on the FRAME_ECCE2 is asserted. The SEM state transitions to 'CRC Err Halt' and the error detection and correction processes stop.

If 'Enhanced Repair' mode is selected in the SEM, an additional CRC-based algorithm is applied at the frame level. By incorporating this algorithm, adjacent double-bit upsets can also be corrected. As mentioned in Section II, before incorporating the Hybrid Scrubber Design, the FPGA in the ROESTI adopted the SEM Design using the Enhanced Repair mode. On the other hand, the Hybrid Scrubber Design does not use that. The reasons for this decision are explained in Section VI.

TABLE I
THE NUMBER OF UPSET BITS AND DETECTED ERRORS BY SEM

the number of upset bits	detected error	final state
SBU	ECC Error	(corrected & not stop)
MBU (even)	ECC Error	ECC Err Halt
MBU (odd) Type 1	ECC Error	ECC Err Halt
MBU (odd) Type 2	ECC Error + CRC Error	CRC Err Halt

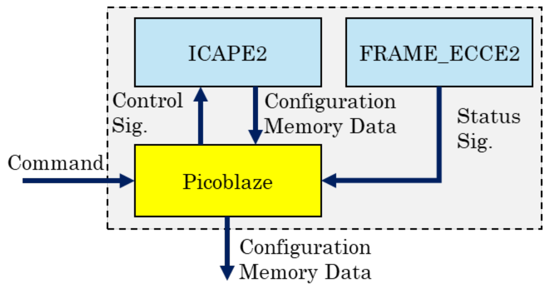

Fig. 6. Diagram of Picoblaze.

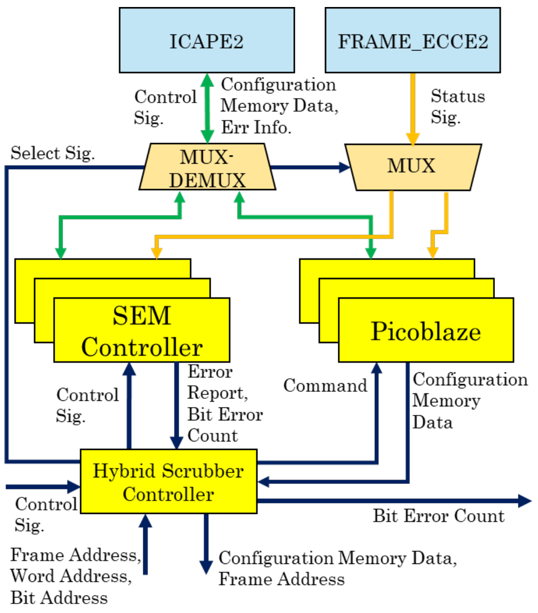
ICAPE2
FRAME_ECCE2
Control Sig.
Configuration Memory Data, Err Info.
Status Sig.
Select Sig.
MUX-DEMUX
MUX
SEM Controller
Picoblaze
Control Sig.
Error Report, Bit Error Count
Command
Configuration Memory Data
Hybrid Scrubber Controller
Control Sig.
Bit Error Count
Frame Address, Word Address, Bit Address
Configuration Memory Data, Frame Address

Fig. 7. Block diagram of Hybrid Scrubber Design.

IV. Structure of Picoblaze

The Picoblaze is a soft microprocessor provided by AMD Inc. Several programs are available for the Picoblaze, including the ’ICAP Controller,’ which enables reading and writing to the FPGA’s configuration memory while it is in operation. Fig. 6 shows a block diagram of the Picoblaze and its peripheral circuits. Similar to the SEM, the status signals are sent from the FRAME_ECCE2 to the Picoblaze. The Picoblaze consists of a processor, program RAM, and a data buffer. The program RAM stores the contents of the ’ICAP Controller’ program. By sending READ or WRITE commands along with the frame address to the Picoblaze, the Picoblaze generates control signals to the ICAPE2, enabling the processing of reads and writes on one frame of the configuration memory.

V. Structure of Hybrid Scrubber

Fig. 7 shows the block diagram of the Hybrid Scrubber Design. The Hybrid Scrubber Controller is the module that controls the entire system. To build a more robust system, we triplicated the modules for the SEM controller and the Picoblaze. As explained in Sections III and IV, both the SEM and the Picoblaze communicate with the ICAPE2 and FRAME_ECCE2. Because only one ICAPE2 can be integrated into a single device, these signals are connected to either the SEM or the Picoblaze through the multiplexer and the demultiplexer. The same structure is applied to the FRAME_ECCE2. Under normal circumstances, the SEM is connected to both the ICAPE2 and the FRAME_ECCE2, operating as shown in the flowchart in Fig. 5. The Picoblaze is connected to the ICAPE2 and FRAME_ECCE2 only when an error that the SEM cannot correct occurs, and it performs the correction. The select signal for the multiplexers and the demultiplexers are determined to enable this condition. We implemented a Hybrid Scrubber Design on the FPGA of the ROSETI as shown in Fig. 8. Compared to the conventional FPGA in Fig. 2, the SEM is replaced by the Hybrid Scrubber. The Hybrid Scrubber exchanges signals with the Network I/F to communicate with a PC. In Section III, we mentioned that when an MBU occurs, the SEM transitions to the ’ECC Error Halt’ or ’CRC Error Halt’ state and stops. When these situations arise, error correction processing by the Picoblaze and the PC starts.

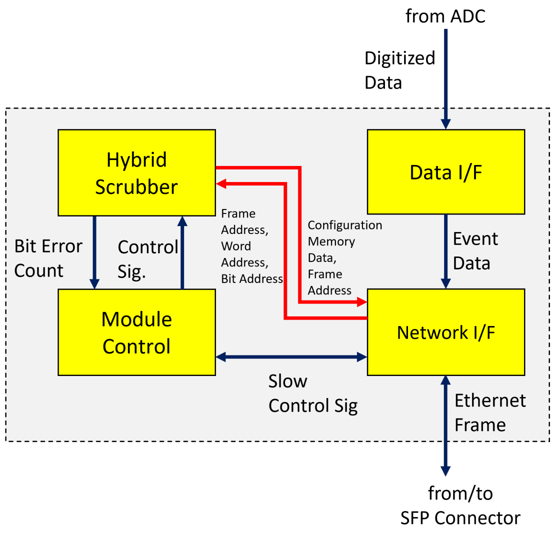

Fig. 8. Block diagram of ROESTI firmware with Hybrid Scrubber Design.

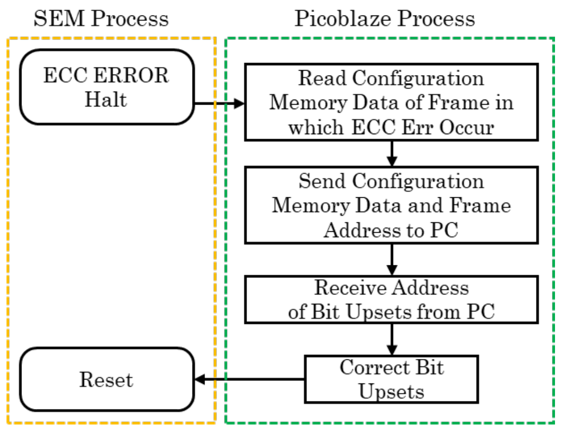
SEM Process
Picoblaze Process
ECC ERROR Halt
Read Configuration Memory Data of Frame in which ECC Err Occur
Send Configuration Memory Data and Frame Address to PC
Receive Address of Bit Upsets from PC
Correct Bit Upsets
Reset

Fig. 9. Flowchart of ECC error correction by Hybrid Scrubber Design.

A. Correcting ECC Error

Fig. 9 shows a flowchart of the ECC error correction process. The ’ECC Error Halt’ in this flowchart corresponds to the ’ECC Error Halt’ in the flowchart of the SEM as

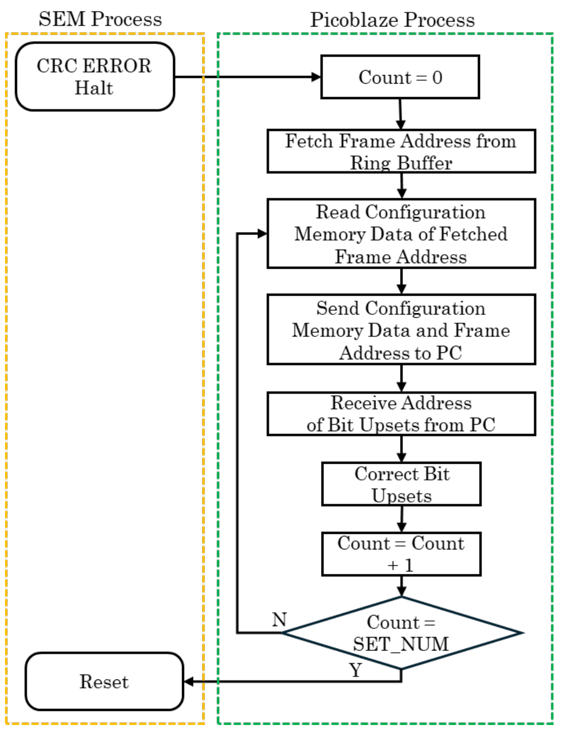
SEM Process
Picoblaze Process
CRC ERROR Halt
Count = 0
Fetch Frame Address from Ring Buffer
Read Configuration Memory Data of Fetched Frame Address
Send Configuration Memory Data and Frame Address to PC
Receive Address of Bit Upsets from PC
Correct Bit Upsets
Count = Count + 1
Count = SET_NUM
N
Y
Reset

Fig. 10. Flowchart of CRC error correction by Hybrid Scrubber Design.

shown in Fig. 5. When the SEM state transitions to 'ECC Error Halt', the correction process of the SEM stops. Simultaneously, the Error Report is sent to the Hybrid Scrubber Controller. This includes the frame address of the frame in which the error occurred. The SEU Scrubber Controller switches the select signal, connecting the Picoblaze to ICAPE2 and FRAME_ECCE2. The Picoblaze reads the configuration memory data of the frame in which the ECC error occurs. This data and the frame address are sent to the PC via the Network I/F over TCP/IP. Although the Network I/F also transmits waveform data, it is configured to prioritize the transmission of configuration memory data and the frame address. The storage device embedded in the PC contains the original complete configuration memory data. After receiving the configuration memory data and frame address, the original configuration memory data of the received frame address is fetched from the storage device by the PC. The original data is compared with the received data in the PC, searching for upset bits. The frame, word, and bit addresses of all the upset bits are then sent back to the FPGA over TCP/IP. When the Hybrid Scrubber Controller receives them, it sends control signals to the Picoblaze. The Picoblaze reverses the value of all the upset bits, allowing the ECC error to be corrected. Finally, the SEM controller is reset, and the select signal is switched to connect ICAPE2 and FRAME_ECCE2 to the SEM. After the reset, the SEM restarts from the 'Read Frame' state as shown in Fig. 5 and returns to its normal state.

B. Correcting CRC Error

As shown in Table I, when odd MBUs of Type 2 occur, a CRC error is detected. In this case, after reading the frame where the error occurred, the SEM transitions to the 'ECC Err

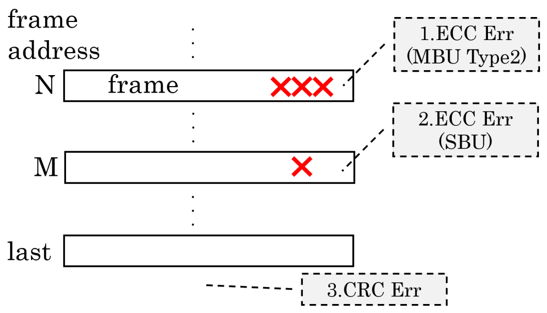

Fig. 11. Example of case where MBU and SBU occur almost simultaneously.

Correction Attempt' state, and an increase of one-bit errors is observed as mentioned in Section III. In the case of SBUs, the SEM can successfully correct the upset bit and correct the ECC error. In either case, the frame address in which an ECC error occurs is stored in a ring buffer. This is because it is not possible to determine the frame address of the frame with MBUs in the case of the CRC error. Fig. 10 shows a flowchart of the CRC error correction process. The 'CRC Error Halt' in this flowchart corresponds to the 'CRC Error Halt' in the flowchart of the SEM as show in Fig. 5.

When the SEM state transitions to the 'CRC ERR Halt', multiple addresses are fetched from the most recently saved frame addresses in the ring buffer and the correction process for those frames begins. The reason for correcting multiple frames is to handle situations such as the example shown in Fig. 11, where an SEU occurs in another frame almost simultaneously with the occurrence of an MBU. In this example, when a CRC error is detected, the last address stored in the ring buffer is the address of the frame where an SBU occurred, and the address of the frame where an MBU occurred is stored second. By fetching multiple addresses and correcting the frames for those addresses, it is possible to correct the frame where an MBU occurred. The number of frames fetched matches the value stored in the SET_NUM register. The default value of this register is four. The process flow for a CRC error correction of one frame is the same as an ECC error correction. This procedure is repeated for the number of frame addresses retrieved from the ring buffer. Similar to the ECC correct, after the reset, the SEM restarts from the 'Read Frame' state as shown in Fig. 5 and returns to its normal state.

C. Errors with Multi-Frames

MBUs occurring across multiple frames are also possible. The Hybrid Scrubber Design can correct these errors. As an example, we assume that a double-bit error occurred in a frame with the frame address of N, and a triple-bit error occurred in the frame with the frame address of N+1, causing a CRC error. When the frame with the frame address of N is read, it is corrected as mentioned in Section V-A. After reset, the frame reading process of the SEM restarts from the frame address of 0. When the frame with the frame address of N+1 is read, it is corrected as mentioned in Section V-B.

The reset latency is crucial. This is because a long latency may prevent the proper correction of CRC errors. We assume

TABLE II
PERFORMANCE OF HYBRID SCRUBBER DESIGN

number of upset bit	detected error	detection +time +correction time (ms)	detection +time +correction time +reset time (ms)
SBU	ECC Err	0.9 to 28.2	
even MBU &odd MBU (Type 1)	ECC Err	5 to 33.3	88 to 116.3
odd MBU (Type 2)	ECC Err +CRC Err	20 to 75.6	103 to 158.6

that a double-bit upset and a triple-bit upset occur, similar to the previous example. If the reset following the ECC error correction caused by the double-bit error is prolonged, it may lead to the occurrence of many SBUs during that time. Frame addresses of these SBUs are stored in the ring buffer. If the number of these frame addresses exceeds the value stored in the SET_NUM register, the correction process for the frame with a triple-bit upset is not performed. If the SEM is set to 'Enhanced Repair' mode, the reset latency is 9.9 seconds. Such lengthy latency may lead to these issues. If the SEM is not set to this mode, the reset latency is 83 milliseconds, and such issues are not expected to occur. Therefore, we do not set the 'Enhanced Repair' mode in the SEM of the Hybrid Scrubber Design.

VI. PERFORMANCE TEST

We measured the time taken for the error correction process in the Hybrid Scrubber Design. In this test, the PC and the ROESTI communicated directly via optical communication. The PC was an HP ProBook 6570b, Intel Core i7 – 3570M (2.90 GHz, two Cores, four threads) running CentOS Linux release 7.9. When the PC received configuration memory data and a frame address of a frame that may have encountered an error over TCP/IP, it responded with the addresses of all bits that were upset within that frame. The SET_NUM register was set to four by default. Therefore, when a CRC error occurred, the configuration memory data and frame addresses of four frames were sent to the PC. We sent a control signal to the SEM via slow control and generated a pseudo error. When the error was detected, the counter in the FPGA started at 0 and incremented by one with each system clock cycle. The count stopped incrementing once the error was corrected. The time from the error detection to the completion of the correction (correction time) was calculated using the counter value and the clock cycle. In the case of either an even MBU or an odd MBU of Type 1, an ECC error occurs. The correction time for this case was from five to six milliseconds. In the case of an odd MBU of Type 2, ECC and CRC errors occur. The correction time for this case was from 20.0 to 21.0 milliseconds. The correction process for four frame addresses was executed, resulting in a correction time approximately four times that of an even MBU. The time from error occurrence to detection (detection time) was calculated based on the SEM system clock of 66.7 MHz and

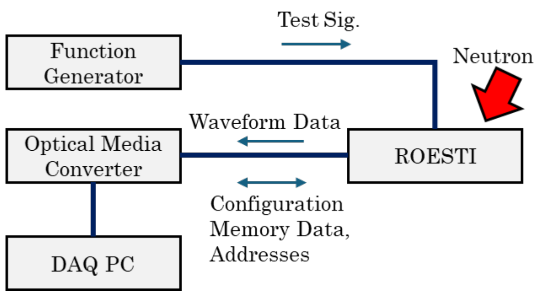

Fig. 12. Setup for neutron irradiation test.

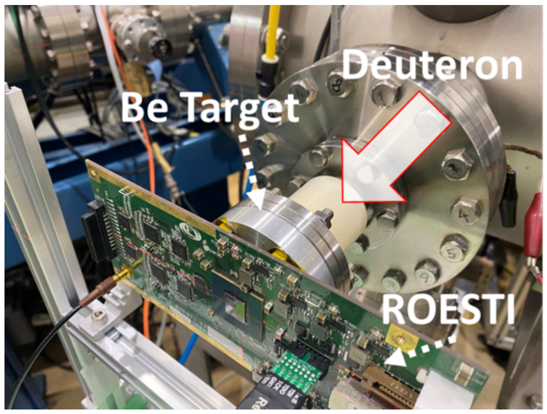

Fig. 13. Photograph of ROESTI in neutron irradiation test.

the SEM datasheet [1]. The detection time for ECC errors is less than 27.3 milliseconds. The detection time for CRC errors is less than 54.6 milliseconds. In the case of both ECC and CRC errors, the absolute minimum error detection latency is effectively zero. In the case of SBUs, the detection time is less than 27.3 milliseconds, and the correction time is 0.9 milliseconds. The time from error occurrence to correction is the sum of detection time and correction time, as listed in Table II. When an ECC or CRC error is caused by an MBU, resetting the SEM restores it to a normal state. The reset time is 83.1 milliseconds. The times, including this reset time, are also listed in Table II.

Most SEUs are SBUs. The correction process for these SBUs is the same as the correction process in the SEM Design, therefore the correction time remains unchanged. The time required to correct MBUs is only several times longer than that required to correct SBUs. This time does not reduce the reliability of the FPGA, because the frequency of MBU occurrence is very low.

VII. NEUTRON IRRADIATION TEST

The primary cause of unrecoverable errors is MBUs. As the Hybrid Scrubber Design is capable of correcting these MBUs, it is anticipated to significantly reduce the frequency of unrecoverable errors compared to the SEM Design. To verify this and demonstrate the effectiveness of the Hybrid Scrubber Design, the neutron irradiation test was performed by using the tandem electrostatic accelerator at Kobe University. This accelerator can generate a neutron beam with an energy peak of 2 MeV by ^{9}Be(d, n)^{10}B reaction with a 3 MeV deuteron beam. Fig. 12 shows the setup for the experiment.

TABLE III
RESULT OF NEUTRON IRRADIATION TEST

	SEM Design	Hybrid Scrubber Design
counts of SEUs	1.0×10^4	1.3×10^4
counts of corrected adjacent double-bit errors	0	260
counts of corrected MBUs w/o adjacent double-bit errors	0	20
counts of unrecoverable errors due to MBUs	19	0
counts of unrecoverable errors due to other errors	4	7

The ROESTI received a test signal from the function generator and sent the waveform data of that signal to the PC, once every second. The PC was an HP ProBook 6570b, equipped with an Intel Core i7-3570M processor (2.90 GHz, two Cores, four threads) running CentOS Linux release 7.9, as during the performance test. We downloaded two types of designs, the Hybrid Scrubber Design and the SEM Design, to the FPGA on the ROESTI and conducted the tests. In the case of the SEM Design, we set 'Enhanced Repair' mode for the SEM to enable the correction of adjacent double-bit upsets. The FPGA on the ROESTI was placed perpendicular to the neutron source as shown in Fig. 13. The FPGA on the ROESTI was exposed to a total neutron fluence of 4.0×10^{11} neutrons/cm^2 in the case of the SEM Design and a total neutron fluence of 5.3×10^{11} neutrons/cm^2 in the case of the Hybrid Scrubber Design.

During this irradiation test, we measured the counts of SEUs, corrected MBUs and unrecoverable errors. Table III shows the results. In the case of the SEM Design, 1.0×10^4 SEUs occurred, including errors in all bits. 19 unrecoverable errors due to MBUs were observed, and 4 unrecoverable errors due to other errors were observed. In the case of the Hybrid Scrubber Design, 1.3×10^4 SEUs occurred, including errors in all bits. 260 adjacent double-bit upsets were observed, and 20 MBUs without adjacent double-bit upsets were also observed. All MBUs were corrected. The counts of MBU accounted for only 0.5 percent of the SEU count. Among the SEUs that occurred, 99.5 percent were SBUs. This indicates that even in the case of the Hybrid Scrubber Design, most SEUs are corrected by the internal SEM of the FPGA, eliminating the need for communication with the external PC.

Fig. 14 shows the results of the unrecoverable error rate. In the case of the SEM Design, the unrecoverable error rate due to MBUs was $(4.7 \pm 1.1) \times 10^{-11}$ cm^2/neutrons, and the unrecoverable error rate due to other errors was $(1.0 \pm 0.5) \times 10^{-11}$ cm^2/neutrons. Adding them together, the unrecoverable error rate was $(5.7 \pm 1.6) \times 10^{-11}$ cm^2/neutrons. Among these, 80 percent were due to MBUs. In the case of the Hybrid Scrubber Design, the unrecoverable error rate was $(1.3 \pm 0.5) \times 10^{-11}$ cm^2/neutrons, which indicates an 80 percent decrease compared to the case of the SEM Design.

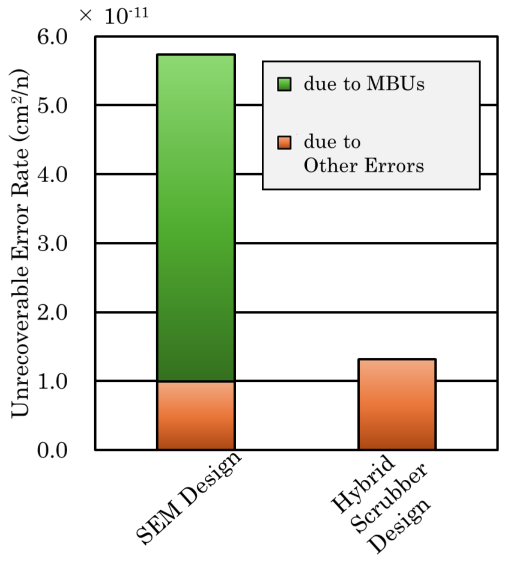

Fig. 14. Result of unrecoverable error rate.

This is attributed to the elimination of unrecoverable errors due to MBUs. This result demonstrates an 80 percent reduction in dead time caused by FPGA firmware re-downloading.

VIII. Conclusion

We developed a Hybrid Scrubber using the SEM and the Picoblaze, capable of correcting not only SBUs but also MBUs. The majority of unrecoverable errors caused by SEUs are attributable to MBUs, which can be eliminated by the Hybrid Scrubber Design. We measured the time required to correct MBUs. We confirmed that the correction time for MBUs does not reduce the reliability of the FPGA. Neutron irradiation tests were conducted and the unrecoverable error rate were measured. Compared to the SEM Design, in which unrecoverable errors occur due to MBUs, the unrecoverable error rate of the Hybrid Scrubber Design decreased by 80 percent. As a result, there was an 80 percent reduction in dead time caused by firmware re-downloads, indicating the effectiveness of the Hybrid Scrubber.

In this study, we implemented the Hybrid Scrubber Design on the FPGA of the ROESTI for the COMET experiment. In this case, Ethernet was utilized for communication with the PC. However, the Hybrid Scrubber Design can operate with other communication methods. Therefore, by appropriately designing the communication method, the Hybrid Scrubber Design can be applied in other experiments. The Hybrid Scrubber Design targets the AMD 7-series FPGA, as the Picoblaze utilizes programs designed for this series of FPGAs. By modifying the content of the program, we anticipate that the Hybrid Scrubber Design can be applied in other FPGA series, such as the AMD Ultrascale family.

Acknowledgment

The authors would like to express their sincere thanks to the staffs of The Tandem Accelerator Laboratory of Kobe University.

References

[1] Soft Error Mitigation Controller v4.1 LogiCORE IP Product Guide, AMD PG036

[2] ”PicoBlaze 8-bit Embedded Microcontroller User Guide for Extended Spartan®-3 and Virtex®-5 FPGAs Introducing PicoBlaze for Spartan-6, Virtex-6 and 7 Series FPGAs”, UG 129 June 22, 2011.,

[3] Hossein Asadi and M. B. Tahoori, ”Analytical Techniques for Soft Error Rate Modeling and Mitigation of FPGA-Based Designs”, IEEE Transactions on Very Large Scale Integration Systems, vol. 15, no. 12, pp. 1320-1331, 2007.

[4] A. Ahmed, ”New FPGA blind scrubbing technique,” 2016 IEEE Aerospace Conference, Big Sky, MT, USA, 2016, pp. 1-9,

[5] H. Michel, et al., ”Read back scrubbing for SRAM FPGAs in a data processing unit for space instruments,” 2015 NASA/ESA Conference on Adaptive Hardware and Systems (AHS), Montreal, QC, Canada, 2015, pp. 1-8,

[6] K. Kyriakoulakos and D. Pnevmatikatos, ”A novel SRAM-based FPGA architecture for efficient TMR fault tolerance support,” 2009 International Conference on Field Programmable Logic and Applications, Prague, Czech Republic, 2009, pp. 193-198,

[7] Y. Kuno and Y. Okada, ”Muon decay and physics beyond the standard model”, *Rev. Mod. Phys.*, vol. 73, pp. 151-202, Jan. 2001.

[8] H. Nishiguchi, et al., ”Development of an extremely thin-wall straw tracker operational in vacuum - The COMET straw tracker system”, *Nucl. Instrum. Meth. A*, vol. 845, pp. 269-272, Feb. 2017.

[9] K. Ueno, et al., ”Design and performance evaluation of front-end electronics for COMET straw tracker”, *Nucl. Instrum. Meth. A*, vol. 936, pp. 297-299, Aug. 2019.

[10] S. Shimazaki, et al., ”Front-end electronics of the Belle II drift chamber”, *Nucl. Instrum. Meth. A*, vol. 735, pp. 193-197, Jan. 2014.

[11] S. Ritt, R. Dinapoli and U. Hartmann, ”Application of the DRS chip for fast waveform digitizing”, *Nucl. Instrum. Meth. A*, vol. 623, pp. 486-488, Nov. 2010.

[12] Y. Nakazawa et al., ”Radiation hardness study for the COMET phase-I electronics”, Nucl. Instrum. Methods Phys. Res. A Accel. Spectrom. Detect. Assoc. Equip., vol. 955, Mar. 2020.

[13] T. Uchida, ”Hardware-Based TCP Processor for Gigabit Ethernet”, *IEEE Trans. Nucl. Sci.*, vol. 55, no.3, pp. 1631–1637, Jun. 2008.

[14] 7 Series FPGAs Configuration User Guide”, Jun. 2015.

[15] A. Stoddard, A. Gruwell, P. Zabriskie and M. J. Wirthlin, ”A Hybrid Approach to FPGA Configuration Scrubbing,” in IEEE Transactions on Nuclear Science, vol. 64, no. 1, pp. 497-503, Jan. 2017