Design Method for Online Totally Self-Checking Comparators Implementable on FPGAs

Yusuke Kanno, *Member, IEEE*, Tadanobu Toba, Kotaro Shimamura, *Member, IEEE*, and Nobuyasu Kanekawa, *Senior Member, IEEE*

***Abstract*— In this article, we propose a method for designing online totally self-checking (TSC) comparators for TSC systems implementable on field-programmable gate arrays (FPGAs). This method can be used to conduct exhaustive online diagnostics of each lookup table (LUT), which involves mapping the fundamental components of the comparator, with a small number of test patterns by directly measuring output of each LUT. Our method drastically reduces the number of test patterns for exhaustive diagnosis on the order of the input number n [$O(n)$] (n is the input number to the comparator) while maintaining 100% coverage, even if we only know the specifications of the LUT without knowing its detailed structure. FPGAs will be easily applicable to systems that require high dependability. To confirm the soft error rate (SER) in a static random-access memory (SRAM)-based FPGA, we also conducted an experiment involving a single-event upset (SEU) caused by neutron radiation. For this experiment, we designed an FPGA implementation of 1575 identical dual-modular-redundant TSC comparators. The experiment was conducted for 10.4 h, and 34 errors were observed regarding such failures in comparator function. The evaluated SER for the TSC comparator with the proposed method was 0.055 FIT at sea level of New York City.**

***Index Terms*— Comparator, field-programmable gate array (FPGA), neutron radiation, single-event upset (SEU), soft error rate (SER), totally self-checking (TSC).**

I. Introduction

With the progress in Industries 4.0, systems, which need to be highly dependable, especially regarding safety applications such as for automobiles, railway systems, chemical plants, and avionics, have become sophisticated due to the acceleration in the use of Internet-of-Things (IoT) technology (involving connecting many sensors) and artificial intelligence (AI) technology (provides highly efficient processing of data gathered using the IoT technology). The processors for such a system need to perform well; however, the production of such processors is limited due to the particularity of applications in highly secure systems. Therefore, high-mix-low-volume development is naturally required for these processors in many cases.

Under such circumstances, using field-programmable gate arrays (FPGAs) is attractive for the development of processors for such systems for economic reasons. The ever-increasing demand for higher performance and functionality embedded in a single chip requires high-performance MOSFETs and more room for implementing logical functions in a single LSI chip; thus, such LSIs require advanced fabrication process technology. However, the development cost of these customized chips is high. Using an FPGA reduces this fabrication cost and provides high-performance capability for data processing and state-of-the-art control methods. Therefore, design methods that achieve high dependability by applying FPGAs to these systems become more important.

To achieve high dependability, redundant codes [1], such as parity, M-out-of-N code [2], and two rail-logic and arithmetic codes [3]–[6], are widely used. Several types of self-checking computer systems have been presented [1], [7]–[11] based on using these redundant codes. To develop self-checking LSIs, another possible method is duplicating functional blocks and verifying the output of these blocks using a self-checking data-compare mechanism through *ad hoc* design. Therefore, intrachip redundancy is commonly used for achieving dependability [12], [13]. Totally self-checking (TSC) comparators with an online fault injection method for intrachip redundancy were proposed [14], [15]. These TSC comparators can conduct self-diagnosis without any modification in the functional blocks, i.e., applying encoding such as error-correction code (ECC), to the functional block. This reduces both development cost and the design period for developing LSIs.

There are issues in applying such comparators to FPGAs. Unlike hard-wired LSIs, FPGAs can implement any functional block by using lookup tables (LUTs). The configuration of the function on an FPGA has many degrees of freedom regarding the connection of each LUT, even though the comparator is small. Therefore, compared to application-specific integrated circuits (ASICs), this high flexibility of FPGAs requires extra test patterns for self-diagnostics when a self-checking comparator is applied to a TSC system. Typically, the number of test patterns for diagnosis of the n-input circuit is 2^n when we treat the function of the circuit as a black-box implementation. This issue is known as the explosion of the diagnosis test patterns [16]. Thus, a design method for a TSC comparator

Manuscript received February 19, 2019; revised May 29, 2019 and August 16, 2019; accepted September 6, 2019. Date of publication January 9, 2020; date of current version February 25, 2020. *(Corresponding author: Yusuke Kanno.)*

Y. Kanno is with the Center for Technology Innovation—Electronics, Hitachi, Ltd., Kokubunji 185-8601, Japan (e-mail: yusuke.kanno.wy@hitachi.com).

T. Toba is with the Center for Technology Innovation—Production Engineering, Hitachi, Ltd., Yokohama 244-0817, Japan (e-mail: tadanobu.toba.ee@hitachi.com).

K. Shimamura and N. Kanekawa are with the Center for Technology Innovation—Controls, Hitachi, Ltd., Hitachi 319-1292, Japan (e-mail: kotaro.shimamura.tx@hitachi.com; nobuyasu.kanekawa.ef@hitachi.com).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2019.2946180

implementable on FPGAs, that is, to reduce the number of test patterns without reducing test coverage, is needed.

Another concern is that a static random-access memory (SRAM)-based FPGA is vulnerable to soft errors caused by a single-event upset (SEU) regarding radiation of cosmic rays [17]. Particularly, neutrons are recognized as a major cause of soft errors for microelectronics on ground. An FPGA for Rad-Hard use [18] is available, but this type of FPGA is for special use; thus, the availability on the market is very limited. Our goal is to use the standard commercial off-the-shelf (COTS) FPGAs for highly dependable systems. To develop such a system, a method for mitigating this problem systematically and quantitative confirmation of the effect of SEU are necessary.

This article is organized as follows. In Section II, we describe the related work that clarifies the issues of applying a conventional method of FPGA implementation, and then, we propose the design method for TSC comparators implementable on FPGAs with an online fault injection method. In Section III, we describe the device under the test (DUT) design that we developed for measuring the soft error rate (SER) per TSC comparator designed with the proposed method. In Section IV, we describe the experimental environment and results. Finally, we conclude this article in Section V.

II. TSC Comparator Implementable on FPGA

A. TSC and TSC Comparator (Related Work)

TSC is indispensable for highly reliable systems, especially regarding safety applications [12], [13], [19]. TSC is defined as follows [2].

Definition 1: A circuit is self-testing if, for every fault from a prescribed set, the circuit produces a noncode output for at least one code input.

Definition 2: A circuit is fault secure if, for every fault from a prescribed set, the circuit never produces an incorrect code output for code inputs.

Definition 3: A circuit is TSC if it is both self-testing and fault secure.

There are two possible approaches to develop TSC LSIs that comply with Definitions 1–3 [14]:

(a) implementing the whole circuit through *ad hoc* design for TSC;
(b) duplicating functional blocks (b-1) and verifying the output of these blocks (b-2) using a TSC data-compare mechanism through *ad hoc* design (b-3).

Approach (a) has shortcomings in which all the circuits must be newly designed and implemented in an *ad hoc* process with extremely strict design restrictions. On the other hand, approach (b) can enable the development of TSC logic, but only if the data-compare mechanism has an *ad hoc* design for fail-safe; in other words, a conventional design of functional block can be easily used for a duplicated functional block. Therefore, development cost and time can be greatly reduced.

The fault-detection coverage with approach (b) greatly depends on the coverage of the TSC data-compare mechanism

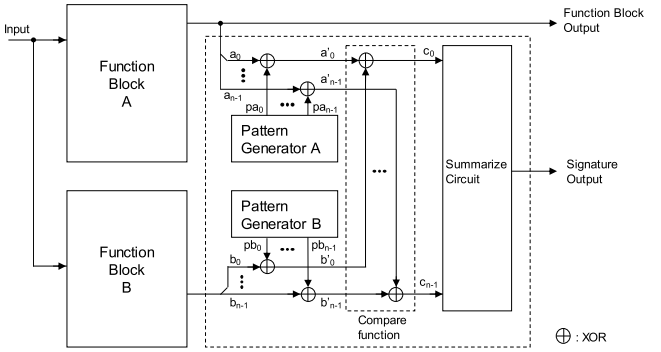


Fig. 1. TSC LSI architecture for approach 2. Two identical function blocks, verification of output from those function blocks, and diagnostics function in comparator (TSC comparator) are included.

(b-3) in the comparator. Generally, 100% coverage for detecting faults in each function block may not be guaranteed because some functions are not in use or not relevant to output. However, the faults that are relevant to output, which are the most important for the system, are detected by duplicating function blocks (b-1) and verifying the output of both blocks in the comparator (b-2); thus, the fault-detection coverage regarding function blocks A and B is 100% with conventional logic-diagnosis methods. The remaining TSC requirement for approach (b) is the TSC function for a comparator (b-3). We call this comparator having this function a TSC comparator.

An example of a TSC LSI architecture is shown in Fig. 1 [14], [15], [19]. Two n-bit inputs (a_0-a_{n-1} and b_0-b_{n-1}) from both dual-modular redundancy (DMR) functional blocks A and B are compared in this TSC comparator. This comparator uses an online fault injection method. The TSC function is provided by adding a diagnostic circuit that consists of a test-pattern generator and injector circuit that typically consists of an exclusive OR (XOR) circuit. This test-pattern injector circuit injects signals (pa_0–pa_{n-1} and pb_0–pb_{n-1}) defined as diagnosis patterns, which are generated from the test-pattern generator, into these input signals, which are compared in the comparator. The summarize circuit summarizes the results of the outputs of compare function (c_0–c_{n-1}) and outputs a signature signal. The overhead (gate counts based on two-input NAND) of this circuit is Reset/Clock ($0.357n$), pattern generator (D-FF, $15n$), pattern injector (XOR, $2.5n$), compare function (XOR, $2.5n$), and summarize circuit (eight-input OR, $0.6875n$); thus, the total overhead is $21.0625n$, which is 8.425 times larger than with a basic comparator [19].

The TSC function in this comparator is as follows. The test signal injected by the test-pattern injector circuit flips the input signal to diagnose hardware-based faults in the comparator within a diagnostic period. When functional blocks A and B are healthy, the output of the comparator is typically "low." When the test signal is injected to one of the input signals, the output of the comparator flips to "high" if the comparator is healthy. Thus, the output of the comparator signal alternates between "low" and "high" (heartbeat) when the comparator is healthy. When the comparator is not healthy, for example, the relevant circuit has a "stack-at-0" fault,

and the comparator output will not flip to "high." This TSC comparator can diagnose its health. A test pattern is designed to detect a comparison mismatch by injecting one test signal into one input one time during the injection period. Therefore, the comparator can output a heartbeat signal, which alternatively switches from "low" to "high," if both function blocks and the comparator are healthy.

B. Proposed Design Method of TSC Comparators Implementable on FPGAs

With conventional design methods [12]–[15], [19], (b-1) and (b-2) are easily embeddable in an FPGA; however, (b-3) is insufficient. Regarding (b-3), the issues with implementing comparators in FPGAs are as follows.

1) Additional test patterns are required for detecting faults in a specific failure mode that is based on the nature of the programmable logic of an FPGA when a comparator is mapped to LUTs.
2) The probability of short circuit for wiring in FPGAs is higher than that in ASICs due to the SEU in configuration memory (CRAM). There are cases in which some faults are missed because the method discussed in a previous study [12] is only considered "stack-at faults."

Fault models for mapping a circuit to an LUT in an FPGA have been reported [20], [21]. Basically, the entity of a k-input LUT is a 2^k-bit SRAM cell. We can diagnosis a k-input LUT by using the 2^k test patterns as long as we observe the LUT output [20], [21]. Short circuits of wiring between two nets [22] are also nonnegligible due to the LUT-based circuits having a large degree of freedom of implementation due to the programmability of LUTs. Typically, the number of test patterns for diagnosis of the n-input circuit is 2^n when we treat the function of the circuit as a black-box implementation. Naive implementation of a TSC comparator to an FPGA results in an explosion of test patterns [16].

To reduce the number of test patterns without reducing test coverage, the proposed method is also used to observe the outputs of each LUT that consists of a comparator function. With this method, primitive components in a comparator, such as XORs and logical ORs, are manually mapped into each LUT, and the outputs of all these LUTs are monitored. This method drastically reduces the combinations of test patterns. If we assume four inputs for each LUT, the required test pattern for an LUT is $2^4 = 16$. For a comparator, the required number of LUTs is as follows:

$$\mathrm{K_{LUT}} = \mathrm{round}\left(\frac{n}{2}\sum_{i=0}^{\infty}\left(\frac{1}{4}\right)^i\right) = \mathrm{round}\left(\frac{2}{3}n\right). \quad (1)$$

To define test patterns, we exhaustively considered fault modes, such as a stack-at fault of the input node/output node, LUT failure mode, and short-circuit effect in an internal comparator. We determined total faults per an LUT (four inputs) based on the following fault models:

1) flipping a bit in CRAM: $2^4 = 16$;
2) stack-at fault at input: $2 \times 4 = 8$;
3) stack-at fault at output: $2 \times 1 = 2$;

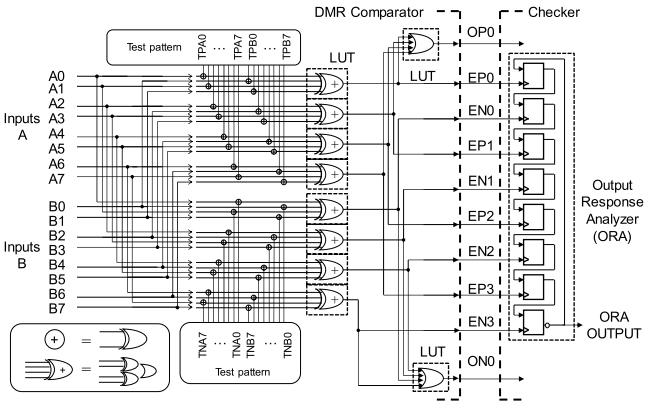


Fig. 2. Example schematic of dual-modular-redundant TSC comparator designed with the proposed method.

4) short circuit between two internal signals (each LUT has one output defined as an internal signal): $K_{\mathrm{LUT}} - 1$.

The total number of possible failures is $(25 + K_{\mathrm{LUT}}) \times K_{\mathrm{LUT}}$. However, we found that we can use the test patterns for the first case to detect faults for the second-to-fourth cases with a combination sequence of several test patterns in the first case. Thus, the total number of test patterns for a comparator is

$$p = 2^4 \times K_{\mathrm{LUT}} \quad (2)$$

which is on the order of n $[O(n)]$ (n is the input number to the comparator). As a result, this method can reduce the total number of test patterns from $O(2^n)$ (worst case) to $O(n)$, which is acceptable for real use cases of online diagnosis. Our method does not necessarily require random sampling simulations, such as Monte Carlo, to verify TSC comparator functions because the total number of test patterns is O(n); thus, we can verify all cases through exhaustive evaluation.

Fig. 2 shows an example of a TSC comparator designed with the proposed method for DMR with 8-bit inputs and a checker. Both 8-bit inputs, A0–A7 and B0–B7 are connected to the test-pattern injector consisting of XOR circuits. One comparator (upper comparator in Fig. 2) is diagnosed at the positive clock edge, and the other (redundant lower comparator in the figure) is diagnosed at the negative clock edge. Test patterns TPA0–TPA7 and TPB0–TPB7 are connected to the upper comparator, and test patterns TNA0–TNA7 and TNB0–TNB7 are connected to the lower comparator. The inputs for the comparators are allocated in each LUT, and the output of each LUT is connected to a clock terminal of the corresponding flip-flop (FF) in the checker. This circuit analyzes the signature signals from the comparator, which we call an output response analyzer (ORA). Outputs EP0–EP3 and EN0–EN3 sequentially switch from "low" to "high" when the test pattern is injected. Therefore, FFs in the checker propagate a "high" signal in accordance with the input of these outputs.

Example of comparator-operation waveforms is shown in Fig. 3, which is an example of fault injection operation. Each input is injected with a fault signal within an appropriate time based on a defined test pattern. Test patterns, TPA and TPB, are injected at the positive edge of the clock signal. Test patterns, TNA and TNB, are injected at the negative edge

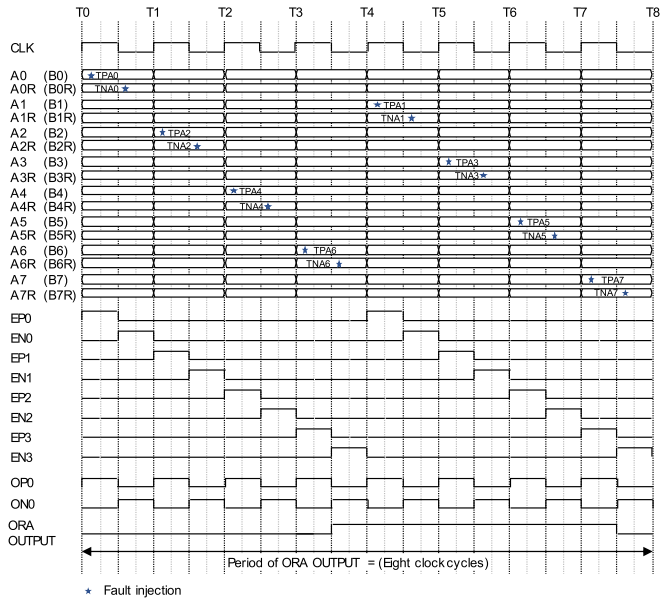


Fig. 3. Example waveforms of TSC comparator. It shows 16 out of 160 patterns. For B0–B7, the corresponding test pattern is replaced with TPB# and TNB#.

of the clock. EP0–3 and EN0–3 become "high" sequentially, as shown in Fig. 3; thus, the FF in the checker propagates the signal in accordance with the inputs of the TPA, TPB, TNA, and TNB signals, and outputs ORA OUTPUT. The period of the ORA OUTPUT is eight clock cycles. We measured this period for a later experiment by using the CHK_CNT counter. The notation OP0 is the output of the upper comparator, and ON0 is the output of the redundant comparator.

In the case of short circuit between two internal signals (EP#, EN#, OP#, and ON#), we can observe all internal signals; thus, we can find these faults by measuring each sequential signal pattern compared with the expected one. In the case of short circuit of wiring between an internal signal in the comparator and an external signal of the comparator, misdetection may occur. For example, this misdetection occurs when an external signal erroneously connected to an internal signal synchronously outputs the same value as the expected value of the internal signal. However, since the external signal is independent of the TSC operation of the comparator, the signal switches asynchronously and randomly to the internal signal. The misdetection probability decreases exponentially by performing several diagnoses that are conducted continuously online, while function blocks A and B are in use. Thus, all faults can be detected within a diagnostics period.

The proposed method can also detect faults that would be latent when diagnosing LUTs with the conventional method. For example, a TSC comparator designed with the proposed method consists of XORs in the first stage and logical ORs in the second stage, as shown in Fig. 2. We consider a case in which an input of one of the XORs in the first stage has a fault such as a short circuit between that of the other XORs (e.g., A0 and A2) and the second OR also has a fault that changed its original OR function to a different one simultaneously (e.g., although the OR function is correct to output OP0 = 1 when EP0 = 1, EP1 = 1, EP2 = 1, and EP3 = 0, the function changes and outputs OP0 = 0 as an unintended function when EP0 = 1, EP1 = 0, EP2 = 1, and EP3 = 0). In this case, TPA0 = 1, TPA4 = 1, and the remaining TPA# = 0 and TPB# = 0 are expected to output EP0 = 1, EP1 = 0, EP2 = 1, and EP3 = 0, but the outputs are EP0 = 1, EP1 = 1, EP2 = 1, and EP3 = 0 due to the short circuit of A0 and A2. The proposed method can detect this fault because each output of an LUT, i.e., EP0–3, is directly observed. It is difficult to detect this fault using the conventional design method; thus, this fault is latent. On the other hand, by using our method, since the intermediate node, such as the output of XOR, is observable, we can detect this fault in the first stage by measuring EP0–EP3 with simple LUT diagnostics.

The proposed method provides 100% coverage for internal faults and short circuit of two wires within a comparator because it tests each LUT by using the above 2^4 test patterns and monitors each LUT's output.

Regarding the generation of test patterns for diagnosis, the required number of test patterns for the TSC comparator is $2^4 \times K_{\mathrm{LUT}}$, as shown in (2). When $n = 8$, we obtain $K_{\mathrm{LUT}} = 5$ and $2^4 = 16$; as a result, the number of test patterns is 80. We use a dual-modular-redundant TSC comparator, as shown in Fig. 2; thus, the total number of test patterns is 160. To reduce hardware overhead, we determined 40 patterns for basic injecting signals considering the symmetry of four situations such as input (A/B) and evaluation clock edge (positive/negative); thus, the total number of patterns is 160 for four identical diagnoses of input A at the clock rise edge, input A at the clock fall edge, input B at the clock rise edge, and input B at the clock fall edge. This circuit can diagnose the dual rate of the system clock; thus, the total diagnosing time is 80 cycles. These test patterns are generated manually for the purpose of easy explanation of the fault-detection coverage for safety applications to a third-party certification authority. We also manually mapped these basic circuits of the comparator to LUTs for the same purpose. Test patterns are stored in on-chip RAM and fed to the comparator during its diagnosis.

Our method is applicable to the software-compare method discussed in a previous study [12] when we map the architecture in that study to an FPGA. First, the test patterns expand when the applied LUT-based design complies with our method. Second, the output of each LUT is set to register when a fault is detected. Finally, the CPU reads these registers and compares them with the expected values. Due to the observing outputs of each LUT, the number of test patterns can be reduced.

The comparators designed with the proposed method are ideally implemented in two separate chips. The comparator function is implemented in an FPGA with the main functional blocks, and the checker is embedded in another hard-wired LSI chip, such as ASIC or discrete LSIs, to reduce the SEU caused by cosmic rays. In this case, to embed a comparator with the main functional blocks, the I/O between two chips is reduced. However, operational speed is limited, so we choose an implementation in accordance with performance requirements and the requirement of standards, such as functional safety.

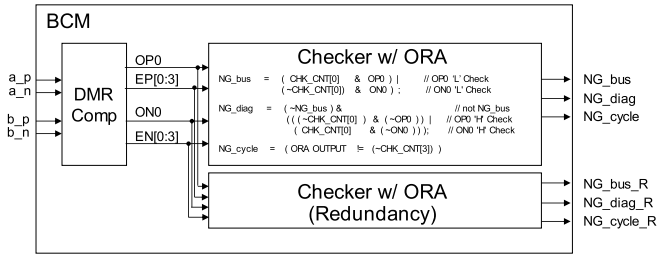


Fig. 4. Block diagram of BCM for the experiment of neutron radiation. ORA in Fig. 2 was implemented in the checker block. Checker block outputs NG_bus, NG_diag, and NG_cycle. This BCM has two checkers with ORA as redundancy.

III. Design of Comparator-Evaluation Module for Measuring Soft Error

In this section, we describe a DUT for SER evaluation. High-energy neutrons, protons, muons, and so on are primarily generated by the spallation reaction of extremely high-energy cosmic rays such as protons and a helium atom nucleus when they enter the atmosphere [23]. Charged particles are halted in a relatively short range; however, neutrons produce a cascade of spallation reactions (air shower), which form terrestrial neutrons at the ground level. Thus, these secondary cosmic-ray neutrons are recognized as a major cause of soft error for microelectronics on the ground. The terrestrial neutron flux at sea level in New York City is 12.9 cm^{-2} h^{-1} ($E > 10$ MeV), which is a reference point of the SER on the Earth [24]. Since the neutron fluence is small on the ground, the probability of measuring the SER is also small. One way to increase the error rate is to conduct an accelerated life test (ALT) [25], which is often used to irradiate a chip with higher intensity particles that can cause errors. We applied this method to this experiment.

To evaluate the SER of a TSC comparator designed with our method and implemented on an SRAM-based FPGA against neutron radiation, we designed a DUT with modules for this comparator and an ORA implemented on an FPGA. For effective analysis, control of the neutron-radiation fluence is essential [17]. Although we conducted this experiment with the ALT, if the irradiation fluence is much stronger, the frequency of multiple event upsets increases, causing difficulty in analysis. To obtain meaningful results from a machine time of several hours, at least 100 events are necessary for measuring the events for the DUT. We estimated that at least 1000 identical DMR comparators were required for the experiment.

Fig. 4 shows a block diagram of a basic comparator module (BCM). We designed a DMR comparator and two checkers with an ORA. Conceptually, the single checker is designed in the hard-wired LSI external to the FPGA, as shown in Fig. 2. However, in this experiment, we used an off-the-shelf evaluation board with an FPGA; thus, checkers must be embedded in the same FPGA. To easily identify the error type, we adopted DMR for the checker.

In this checker, three signals, NG_bus, NG_diag, and NG_cycle, are generated using the output of the comparator. The NG_bus signal signifies that the input bus of the comparator is healthy and is generated as

$$\text{NG\_bus} = ((\text{CHK\_CNT}[0] \ \& \ \text{OP0}) | (\sim \text{CHK\_CNT}[0] \ \& \ \text{ON0})) \quad (3)$$

where CHK_CNT[0] is the output of the least significant bit (LSB) in the 4-bit counter. This logic checks the output of the comparator, i.e., OP0 is low in the high state of the clock and ON0 is low in the low state of the clock. Our DMR comparator consists of two identical comparators. One comparator diagnoses its circuit at the positive edge of the clock signal, and the other diagnoses its circuit at the negative edge of the clock signal, as described in Section II. Thus, our DMR comparator carries out self-diagnosis for double the clock frequency.

The NG_diag signal signifies the health of the comparator and is generated as

$$\text{NG\_diag} = (\sim \text{NG\_bus}) \ \& \ ((\sim \text{CHK\_CNT}[0] \ \& \sim \text{OP0}) | (\text{CHK\_CNT}[0] \ \& \sim \text{ON0})). \quad (4)$$

This DMR comparator injects test signals during the diagnosing period. As a result, the output is expected to be high if the comparator is healthy.

The NG_cycle signal signifies the health of the ORA output

$$\text{NG\_cycle} = (\text{ORAOUTPUT!} = (\sim \text{CHK\_CNT}[3])) \quad (5)$$

where CHK_CNT[3] is the output of the most significant bit (MSB) of the 4-bit counter. As mentioned earlier, since this DMR comparator diagnoses at double the clock frequency and the input signal to the ORA is 8 bits; an 8-bit counter that operates at double the clock frequency is needed. For signal OUTPUT ORA, the "high" level propagates sequentially through each FF in the ORA in accordance with dual-rate toggling due to the EP0-3 and EN0-3 signals. As a result, signal OUTPUT ORA alters at the rising edge of the clock for every four clock cycles.

A schematic of the DUT that includes 75 comparator-evaluation modules (CEMs) with seven triplet comparator modules (TCMs) is shown in Fig. 5. This DUT had a total of 1575 DMR TSC comparators and associated checkers that include the ORA. Each TCM has three basic BCMs. The TCM outputs three types of signals, NG_bus, NG_diag, and NG_cycle, and redundancy signals, NG_bus_R, NG_diag_R, and NG_cycle_R. All signals from each BCM are drawn to the top hierarchy in parallel, the total number of signals is 3150 for each signal type, and each signal consists of 18 bits. These signals are OR each other in accordance with the same number of TCMs. The selector circuit selects a CEM number (CEM #) to identify the position where failure occurred. If multiple errors occurred simultaneously, only data from the lowest CEM # are stored in the register due to the limitation of the design specifications.

Fig. 6 shows the top view of the circuit for this experiment. All comparators in the DUT had the same inputs for easy control. We focused on the SER of the comparators against neutron radiation; thus, we designed the input signals as dc inputs. These input signals were connected to a test-pattern

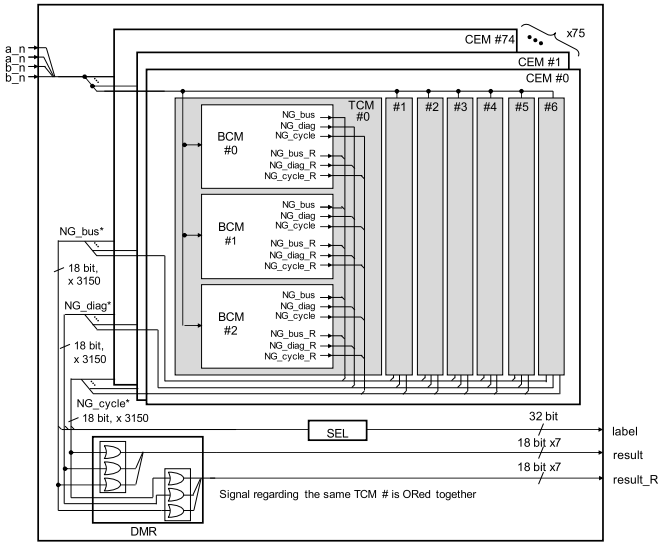


Fig. 5. Schematic of DUT module that includes 75 CEMs with seven TCMs. Each TCM has three BCMs.

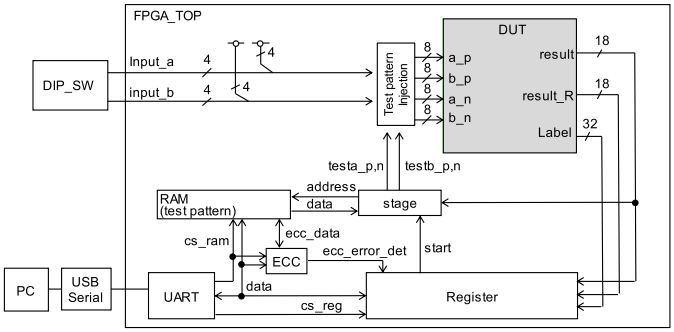


Fig. 6. Top view of circuit evaluated in this experiment.

injection circuit, and this circuit flipped input signals when the test patterns were injected. The measured data were stored in a register. The test patterns stored in an on-chip RAM were protected by ECC, and each test pattern was injected via the injector circuit after reading from the RAM. When an ECC error occurred, the error information was stored in the register. Stored data in the register were sent to a PC after the error occurred.

We used Xilinx's Virtex 7 FPGA (XC7VX485T-2FFG1761, VC707 Evaluation Kit) [26]. The hardware resources for implementing our circuit in the FPGA are listed in Table I. FPGA TOP includes DUT and Exc. DUT, and DUT includes CEM and Exc. CEM. As shown in Fig. 6, Exc. DUT includes RAM, registers, and other circuits. In this design, we implemented 12 kb of control registers, 1.2 kb of test pattern registers, and 32 kb of registers for logging data to both Slice Reg and LUTRAM. We did not use BRAMs. The total number of slices in XC7VX485T was 75 900, and our design used 97% of all slice resources. As shown in Table I, wiring resources used 4726 slices, which was 6.4% of all resources. Note that this design is for SER evaluation of comparators, so the overhead of wiring was large. However, the overhead can be reduced because one or two comparators at most are implemented on an FPGA for real use cases.

TABLE I
FPGA RESOURCES FOR THIS IMPLEMENTATION EVALUATION

Modules	Slices	Slice Reg	LUTs	LUTRAM
FPGA_TOP	10/73434	8/95956	31/83340	8/3593
Exc. DUT	1208	1380	2597	443
DUT	267/72226	75/94576	799/80743	0/3150
Exc. CEM	4726	1	2068	0
CEM	6/900	0/94576	21/1049	0/42
TCM	0/127	0/181	0/150	0/6
BCM	0/42	0/61	0/50	0/2
DMR Comp	0/10	0/0	0/10	0/0
Checker	17/17	31/31	19/19	1/1

Number in each cell denotes (# of use of current level) / (total # including that of lower level hierarchy modules). Typical values for CEM, TCM, BCM, DMR Comp, and Checker are shown in this table.

TABLE II
SPECIFICATIONS OF RCNP OF OSAKA UNIVERSITY

Test Facilities	Osaka Univ, Research Center for Nuclear Physics
Beam type	Spallation beam
Neutron energy	Energy: 1 – 400 MeV Proton beam current: 1 μA Beam shape: Circle (108 mm ϕ)

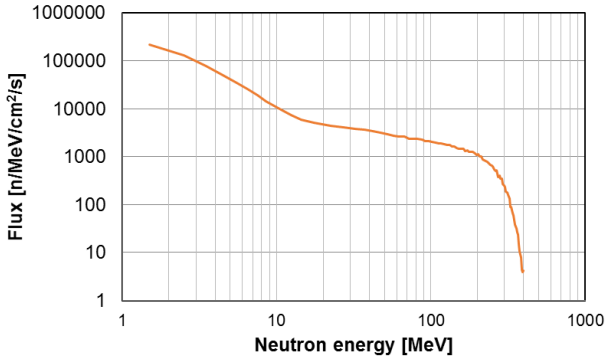


Fig. 7. Neutron flux during the experiment.

IV. EXPERIMENT AND RESULTS

We conducted the abovementioned experiment at the Research Center for Nuclear Physics (RCNP) of Osaka University. The specifications of the RCNP facility are summarized in Table II [27]. The neutron flux is shown in Fig. 7.

The soft error mechanism is explained as follows. When a nucleus in the device collides with a ballistic neutron, a nuclear spallation reaction, in which the nucleus breaks into secondary fragments, can take place with a certain probability. Similar to alpha-ray soft error, when the storage node (diffusion layer) is hit by a secondary ion, a certain amount of electrons/holes produced along the ion track are collected at the nodes, typically by the funneling mechanism [28] and/or the drift-diffusion process. An SEU occurs when charge collected at the node exceeds the critical charge over which the data "1 (high)" in the node changes to "0 (low)."

With measurements using spallation neutrons, one can derive an averaged neutron SEU cross section that can be defined as follows [24]:

$$\overline{\sigma_{\mathrm{SEU-dev}}} = \frac{N_{\mathrm{SEU}}}{\Phi_{\mathrm{spec}}}. \tag{6}$$

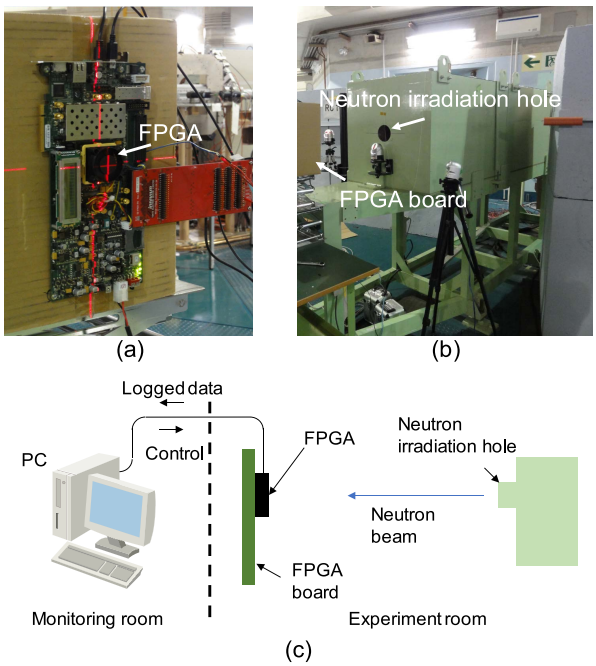


Fig. 8. Experimental setup. (a) Target FPGA board with comparator modules. Embedded chip was Xilinx Virtex 7 FPGA. (b) Neutron irradiation hole at RCNP facility. (c) Block diagram of the experimental setup.

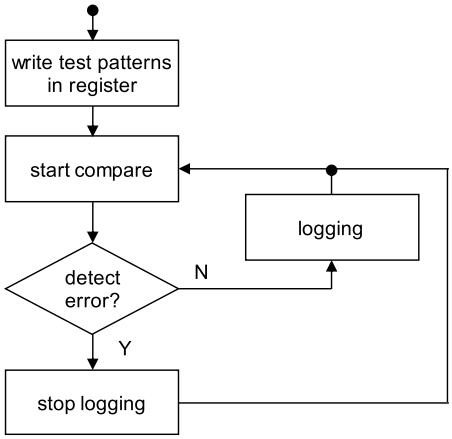


Fig. 9. Evaluation procedure of logging failure during the experiment.

TABLE III
ERROR TYPE CLASSIFICATION

Error Type	Description
E1	Diagnosis error (DE) (NG_diag)
E2	Pulse cycle error (PCE) (NG_cycle)
E3	Output bus error (OBE) (NG_bus)
E4	DE + PCE
E5	DE + OBE
E6	DE + PCE + OBE

In (6), the averaged cross sections depend on the entire spallation neutron spectrum, where N_{SEU} denotes the total errors detected, Φ_{spec} is the fluence of neutrons over the spectrum from $E > 10$ MeV in units of cm^{-2}, and $\overline{\sigma_{\mathrm{SEU-dev}}}$ is the spallation SEU cross section in units of cm^2/device.

A. Experimental Setup

The experimental setup is shown in Fig. 8. Fig. 8(a) shows the target FPGA board with the DUT module, and Fig. 8(b) shows the neutron radiation hole. Fig. 8(c) shows a block diagram of the experiment setup. The FPGA was in front of a neutron irradiation hole and connected with a host PC via a serial connection, as shown in Fig. 6. The serial connection was used to configure the FPGA and collect log data stored in registers on the FPGA to the PC.

The evaluation procedure of the data-logging method used in this experiment is shown in Fig. 9. First, a test pattern was loaded from an external ROM to the on-chip RAM. After the test pattern was loaded in the RAM, diagnosis of the comparator began. The data of the comparison results were logged continuously until an error occurred. When an error occurred, data logging stopped and the stored data in the register were sent to an external PC. After sending the data, the DUT was reset and data logging resumed. All data logged in the register were sent to the PC after an error occurred, so we could analyze these data after the experiment.

B. Experimental Results

First, we considered the dependence of error types and components regarding measured errors. All error types measured during this experiment are listed in Table III. As mentioned in Section III, we designed three signals to detect errors, NG_bus, NG_diag, and NG_cycle. There were a total of six combinations of these errors because if NG_bus and NG_cycle are detected simultaneously, NG_diag should also be detected, that is, this type of error is under the same condition that detects all three signals simultaneously. We observed no errors regarding the case in which NG_bus and NG_cycle were detected and NG_diag was not detected. When an error in the external circuit that we were not interested in occurred, the corresponding "invalid data" were set to the register. A diagnosis circuit detected this irregularity, and then, data logging stopped. Therefore, we can easily and automatically distinguish the data of interest for the corresponding error detection and other irregular invalid data.

Fig. 10 shows the error-occurrence frequency categorized by components and error type. Components in which errors occurred were categorized as checker (CHK), comparator (CMP), input of comparator (INPUT), and others, and the case of multiple errors occurring was also considered. We did not obtain results, indicating that some logged data were the invalid data mentioned earlier, except for the seven operational errors that were due to circuit hang-up on the PCB board and misoperation. We confirmed that every result consisted of valid data and all errors associated with the circuit of interest were detected.

The frequency of error detection in the DMR comparators was 34 and that in the checkers was 112. The ratio of this frequency in the comparators to that of checkers was 1:3.3. Regarding the area for the DMR comparator and checkers of a BCM in the FPGA, the DMR comparator required ten slices in the FPGA layout, and the checkers required 34 slices

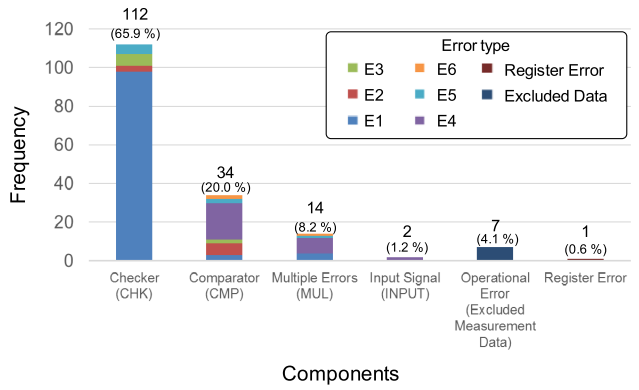


Fig. 10. Error-occurrence frequency classified by each error type.

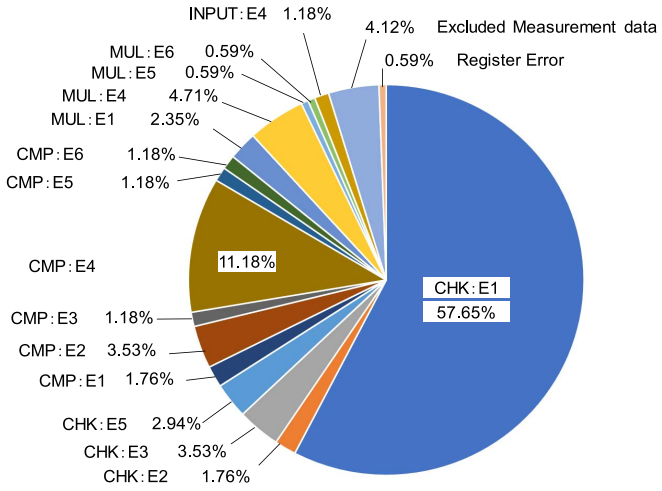


Fig. 11. Ratios of detected errors classified by components and reasons. No errors were detected in CHK:E4, CHK:E6, MUL:E2, MUL:E3, INPUT:E1, INPUT:E2, INPUT:E3, INPUT:E5, and INPUT:E6.

(typical value). Therefore, the ratio of the area of comparators to that of the checkers was 1:3.4. Both ratios were similar; thus, the error-occurrence frequency was proportional to the area. This is the consequence of uniformity regarding the SER in an FPGA.

Fig. 11 shows the ratio of detected errors classified by components and reasons, which is a breakdown of the results in Fig. 10. The most frequent reason was the reporting of a single failure in the checker with ORAs. This is caused by an inconsistency between DMR ORAs due to an accidental flip of relevant CRAM bits or the storage node in FF along the connection path. The excluded measurement data were the operational error data in Fig. 10. Note that we detected the case of CMP:E2 (3.53%), which would be latent with conventional methods.

All events detected during this experiment are shown in Fig. 12 and classified by cause in chronological order. The measured CEM # associated with the measured error events is shown in the chronological order in Fig. 13. These results show that the measured error occurred uniformly in FPGA.

C. SER Evaluation

Next, we calculated the SER. We referred to the JEDEC standard for this experiment [24]. The neutron fluence during

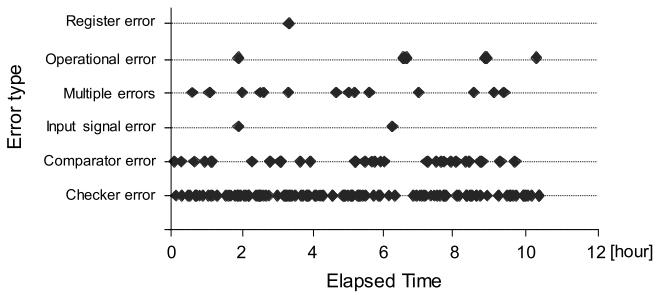


Fig. 12. Chronological order of each measured error type.

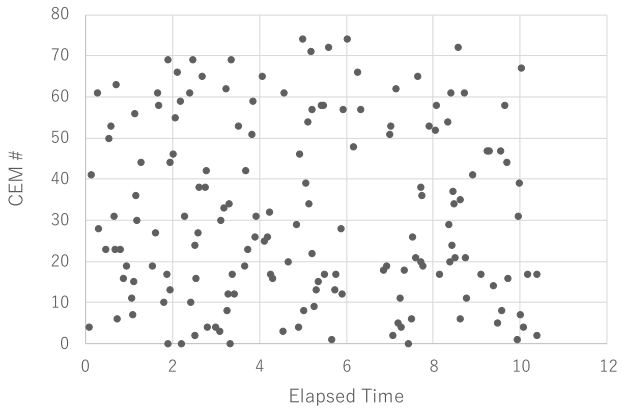


Fig. 13. Chronological order of CEM number for detecting faults.

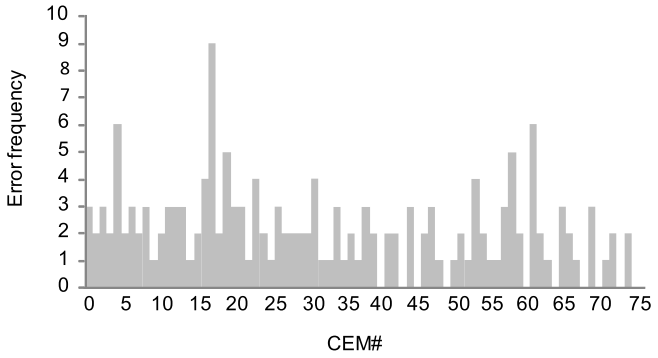


Fig. 14. Error-occurrence frequency of each module.

the experiment was $\Phi_{spec} = 5.27 \times 10^9$ n/cm^2, which is 3.91×10^7 times greater than that at sea level in New York City, and the FPGA was irradiated for 10.4 h. Hence, the amount of neutrons radiated in the experiment was about the same as that radiated in 4.06×10^8 h at sea level in New York. We obtained an SER $= (163/(5.27 \times 10^9)) \times (10^9 \times 0.0036 \times 3600) =$ 401 FIT/DUT at sea level in New York City for our DUT. We omitted the excluded measurement data in Fig. 10 for this calculation.

To evaluate the SER per DMR comparator, we assumed that the SER is proportional to the area in use. The total resources used to implement our DUT were 73 434 slices, and a DMR comparator used ten slices. Thus, the SER per DMR comparator was 0.055 $(= 401 \times 10/73434)$ [FIT/COMP] at sea level in New York City.

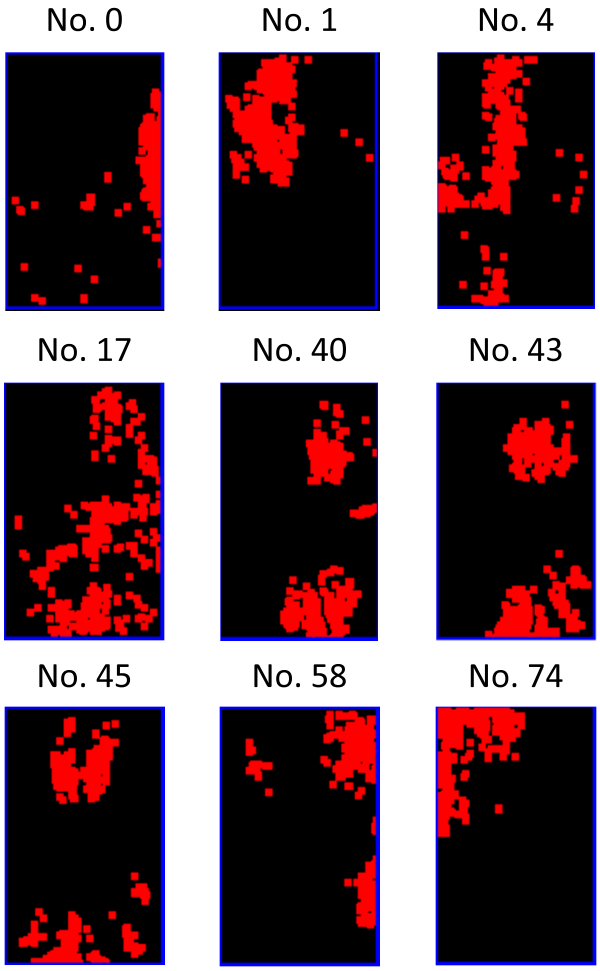


Fig. 15. Layout example for several CEMs.

D. Layout Dependence

Finally, we describe the dependence of error events for physical layout. The error-occurrence frequency for each module is shown in Fig. 14. On average, 2.16 errors were observed in each CEM. The error frequency in CEM 17 was largest in this experiment. For comparison, the layout patterns for CEMs 0, 1, 4, 17, 40, 43, 45, 58, and 74 are shown in Fig. 15. The error-occurrence frequencies were 3, 2, 6, 9, 0, 0, 0, 5, and 2, respectively. The red areas in these layout patterns are the associated slices used for comparators. Although the real connection in the FPGA is unknown to users, a layout pattern represents almost the same location in the FPGA as the actual location. Among these layout patterns, the layout pattern of CEM 17 was laid out uniformly in the FPGA chip; however, those of CEMs 40, 43, and 45 were laid out in two distinct areas. The layout patterns for CEMs 4, 17, and 58 were longitudinal and near the edge of the chip. We designed our DUT as a hierarchical structure in the register-transfer level (RTL); however, we synthesized and mapped this DUT into an FPGA in the top level of the hierarchy due to the limitations of the electronic design automation (EDA) tool. Therefore, all BCMs, TCMs, and CEMs had different layouts in each hierarchy. This is the reason for the variation in layout images.

The SEU mitigation reports from FPGA vendors do not include any dependence between the SER and LUT position in an FPGA, and we can assume the uniformity of the SER in an FPGA. Even though this uniformity assumption is true, more circuits, such as switch and repeater, are required in an FPGA; thus, this might increase the SER.

Based on the experimental results, we conclude that it is possible to design dependable systems by using FPGAs with a TSC comparator designed with the proposed method and applying an authorized design methodology defined in the functional safety standard [29].

V. Conclusion

We proposed a design method for TSC comparators implementable on FPGAs with an online fault injection method and evaluated the SER per comparator example through an experiment of neutron radiation exposure. By observing the output of each LUT directly, the proposed method was used to conduct self-diagnosis with 100% coverage in a small number of test patterns, which drastically reduced the number of test patterns to $O(n)$. Although we did not know the physical layout of the designed circuits in an FPGA, we designed a TSC system with the proposed TSC comparator designed with the proposed method that was implementable on an FPGA with a reasonable number of test patterns. Our method was extended to software-based TSC implementable on FPGAs. To evaluate the SER of such a TSC comparator in an FPGA, we used a DUT consisting of 1575 DMR comparators and measured its effectiveness against neutron radiation. The evaluated SER for a comparator was 0.055 FIT (at sea level of New York City). Thus, we confirmed the possibility of designing dependable systems, especially regarding safety applications, such as for automobiles, railway systems, chemical plants, and avionics, using FPGAs with our method and applying an adequate design methodology defined in the functional safety standard, such as IEC 61508.

Acknowledgment

The authors would like to thank Prof. M. Fukuda of Osaka University for fruitful discussions and support in the neutron experiments. They would also like to thank M. Ohkawa, N. Shimazaki, T. Hosokawa, K. Shimbo, T. Uezono, I. Kawahira, S. Abe, and T. Sakata for their technical support in designing DUT on FPGA and this experiment. They would also like to thank T. Hirotsu, M. Saen, and T. Yamada for their valuable discussions.

References

[1] W. C. Carter and P. R. Schneider, "Design of dynamically checked computers," in *Proc. Congr. Inf. Process. (IFIP)*, Edinburgh, U.K., Aug. 1968, pp. 878–883.

[2] D. A. Anderson and G. Metze, "Design of totally self-checking check circuits for m-out-of-n codes," *IEEE Trans. Comput.*, vol. C-22, no. 3, pp. 263–269, Mar. 1973.

[3] E. Fujiwara, *Code Design for Dependable Systems: Theory and Practical Applications*. Hoboken, NJ, USA: Wiley, 2006.

[4] S. Mathew, M. Anders, R. Krishnamurthy, and S. Borkar, "A 6.5 GHz 54 mW 64-bit parity-checking adder for 65 nm fault-tolerant microprocessor execution cores," in *Proc. IEEE Symp. VLSI Circuits*, Jun. 2007, pp. 46–47.

[5] J. F. Wakerly, *Error Detecting Codes, Self-Checking Circuits and Applications*. Amsterdam, The Netherlands: North Holland, 1978.

[6] T. R. N. Rao and E. Fujiwara, *Error-Control Coding for Computer Systems*. Upper Saddle River, NJ, USA: Prentice-Hall, 1989.

[7] D. A. Rennels, A. Avizienis, and M. Ercegovac, "A study of standard building blocks for the design of fault-tolerant distributed computer systems," in *Proc. FTCS*, Toulouse, France, Jun. 1978, pp. 144–149.

[8] Y. Crouzet and C. Landrault, "Design specification of a self-checking detection processor," in *Proc. FTCS*, Oct. 1980, pp. 275–277.

[9] J. Chavade and Y. Crouzet, "The PAD: A self-checking LSI circuit for fault-detection in microcomputers," in *Proc. FTCS*, Jun. 1982, pp. 55–62.

[10] M. M. Tsao, A. W. Wilson, R. C. McGarity, C. J. Tseng, and D. P. Siewiorek, "The design of C. Fast: A single chip fault tolerant microprocessor," in *Proc. 12th Int. FTCS*, Jun. 1982, pp. 63–69.

[11] D. Efanov, V. Sapozhnikov, and V. Sapozhnikov, "Methods of organization of totally self-checking concurrent error detection system on the basis of constant-weight ≪1-out-of-3≫-code," in *Proc. IEEE East-West Design Test Symp. (EWDTS)*, Oct. 2016, pp. 1–9.

[12] A. Floridia and E. Sanchez, "Hybrid on-line self-test strategy for dual-core lockstep processors," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Nanotechnol. Syst. (DFT)*, Oct. 2018, pp. 1–6.

[13] X. Iturbe, B. Venu, E. Ozer, and S. Das, "A triple core lock-step (TCLS) ARM Cortex-R5 processor for safety-critical and ultra-reliable applications," in *Proc. 46th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. Workshop (DSN-W)*, Jun./Jul. 2016, pp. 246–249.

[14] N. Kanekawa, M. Nohmi, Y. Satoh, and H. Satoh, "Self-checking and fail-safe LSIs by intra-chip redundancy," in *Proc. Annu. Symp. Fault Tolerant Comput.*, Jun. 1996, pp. 426–430.

[15] N. Kanekawa, E. H. Ibe, T. Suga, and Y. Uematsu, "Fault-tolerant system technology," in *Dependability in Electronic Systems*. New York, NY, USA: Springer, 2011, Ch. 5, pp. 143–202.

[16] W. H. Kautz, "Testing for faults in combinational cellular logic arrays," in *Proc. 8th Annu. Symp. Switching Automata Theory*, Oct. 1967, pp. 161–174.

[17] N. Kanekawa, E. H. Ibe, T. Suga, and Y. Uematsu, "Terrestrial neutron-induced failures in semiconductor devices and relevant systems and their mitigation techniques," in *Dependability in Electronic Systems*. New York, NY, USA: Springer, 2011, Ch. 2, pp. 7–63.

[18] D. Ratter, "FPGAs on mars," *Xcell J.*, no. 50, pp. 8–11, 2004.

[19] N. Kanekawa and Y. Sato, "Self-checking comparator using Eigen signature patterns dedicated to wire nets," (in Japanese), *IEICE (Inst. Electron., Inf. Commun. Eng.) D-I*, vols. J79–D-I, no. 6, pp. 353–360, Jun. 1996.

[20] R. Hülle, P. Fišer, J. Schmidt, and J. Borecký, "SAT-ATPG for application-oriented FPGA testing," in *Proc. 15th Biennial Baltic Electron. Conf. (BEC)*, Oct. 2016, pp. 83–86.

[21] M. Rebaudengo, M. S. Reorda, and M. Violante, "A new functional fault model for FPGA application-oriented testing," in *Proc. 17th IEEE Int. Symp. Defect Fault Tolerance VLSI Syst.*, Nov. 2002, pp. 372–380.

[22] J. Kvasnicka, P. Kubalík, and H. Kubátová, "Experimental SEU impact on digital design implemented in FPGAs," in *Proc. 11th EUROMICRO Conf. Digit. Syst. Design Archit., Methods Tools*, Sep. 2008, pp. 100–103.

[23] T. Nakamura, M. Baba, E. Ibe, Y. Yahagi, and H. Kameyama, *Terrestrial Neutron-Induced Soft Errors in Advanced Memory Devices*. Singapore: World Scientific, 2008.

[24] *Measurement and Reporting of Alpha Particles and Terrestrial Cosmic Ray-Induced Soft Errors in Semiconductor Devices*, JEDEC Standard 89A, 2006.

[25] A. Lesea, S. Drimer, J. J. Fabula, C. Carmichael, and P. Alfke, "The Rosetta experiment: Atmospheric soft error rate testing in differing technology FPGAs," *IEEE Trans. Device Mater. Rel.*, vol. 5, no. 3, pp. 317–328, Sep. 2005.

[26] *Xilinx Virtex-7 FPGA VC707 Evaluation Kit*. Accessed: Dec. 21, 2019. [Online]. Available: http://www.xilinx.com/products/boards-and-kits/ek-v7-vc707-g.html

[27] E. H. Ibe, *Terrestrial Radiation Effects in ULSI Devices and Electronic Systems*. Hoboken, NJ, USA: Wiley, 2015.

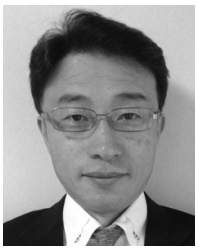
[28] C. Hu, "Alpha-particle-induced field and enhanced collection of carriers," *IEEE Electron Device Lett.*, vol. 3, no. 2, pp. 31–34, Feb. 1982.

[29] *Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems*, Standard IEC 61508-1:2010, 2010. [Online]. Available: https://www.iec.ch/functionalsafety/standards/page2.htm

Yusuke Kanno (M'05) received the B.S. and M.S. degrees in physics from Tohoku University, Sendai, Japan, in 1992 and 1995, respectively, and the Ph.D. degree from the Tokyo Institute of Technology, Tokyo, Japan, in 2009.

He joined the Central Research Laboratory, Hitachi, Ltd., Tokyo, in 1997, where he has been engaged in the research and development of high-speed and low-power circuits for embedded DRAMs and microprocessors. He was engaged in the establishment of the power gating technology for mobile systems on chip (SoCs). In 2010, he managed the research team of the high-speed and power-saving circuit technique for information and telecommunication system. In 2012, he engaged in highly reliable system by using static random-access memory (SRAM)-based field-programmable gate array (FPGA) for industrial business.

Dr. Kanno is a member of the IEEE Solid-State Circuit Society.



Tadanobu Toba received the M.S. and the Ph.D. degrees in system safety and information science from the Nagaoka University of Technology, Niigata, Japan, in 2014 and 2018, respectively.

He is currently a Senior Researcher with the Center for Technology Innovation—Production Engineering, Research and Development Group, Hitachi, Ltd., Yokohama, Japan. His current research activity is dependable technology and terrestrial neutron-induced soft-error analysis.

Dr. Toba is a member of the Institute of Electronics, Information and Communication Engineers (IEICE).

Kotaro Shimamura (M'99) received the M.S. degree in physics from Tokyo University, Tokyo, Japan, in 1990.

In 1990, he joined Hitachi, Ltd, Tsuchiura, Japan. Since then, he has been involved in the research and development of high-performance reduced instruction set computer (RISC) microprocessor for massive parallel supercomputers, fault-tolerant controller for power converters, microcontroller soft core, and fail–safe microcontroller for railway systems.

Mr. Shimamura is a member of the Association for Computing Machinery (ACM), the Institute of Electronics, Information and Communication Engineers (IEICE), the Information Processing Society of Japan (IPSJ), and the Institute of Electrical Engineers of Japan (IEEJ).

Nobuyasu Kanekawa (M'88–SM'19) received the B.Eng., M.Eng., and Ph.D. degrees in engineering from the Tokyo Institute of Technology, Tokyo, Japan, in 1985, 1987, and 1998, respectively.

He joined Hitachi Research Laboratory, Hitachi Ltd., Hitachi, Japan, in 1987, where he is currently the Chief Researcher with the Center for Technology Innovation—Controls. He was a Visiting Scholar with the Computer Science Department, University of California at Los Angeles, Los Angeles, CA, USA, from 1991 to 1992. His major publication is *Dependability in Electronic Systems* (Springer, 2010).

Dr. Kanekawa is an IEICE Fellow, the Ex-President of the Reliability Engineering Association of Japan, a member of the IFIP TC.10 and WG.10.4, International Expert of Functional Safety Standards Committees of IEC.