Fail-Safe Logic Design Strategies within Modern FPGA Architectures

Jim Plusquellic
Electrical and
Computer Engineering
University of New Mexico
Albuquerque, New Mexico 87131
Email: jimp@ece.unm.edu

Andrew Suchanek
and Tom J. Mannos
Sandia National Laboratories
Albuquerque, New Mexico 87123
Email: asuchan@sandia.gov
and tjmanno@sandia.gov

***Abstract*—Fail-safe computing refers to computing systems that revert to a non-operational safe state when a fault occurs. In this paper, we investigate circuit level techniques as mitigations for implementing fail-safe computing processes on field-programmable gate arrays (FPGAs). The propagation of fault effects through FPGA primitives including lookup tables (LUTs) and programmable interconnect points (PIPs) is assessed within an FPGA architecture created using an open source tool. The analysis reveals additional vulnerabilities exist within reconfigurable architectures over those in equivalent fail-safe application specific integrated circuit (ASIC) versions; thus requiring a more elaborate network of redundant circuits and checking logic. An ASIC version of a fail-safe monitoring circuit is designed and compared to the equivalent circuit requirements within an FPGA. A compact fail-safe circuit design technique called DEsign for Fail-safe in reCONfigurable systems (DEFCON) is proposed. The benefits and limitations associated with an FPGA-based fail-safe circuit structure with alarm capabilities, as well as simulation and formal analyses are presented and discussed.**

I. Introduction

Fail-safe systems need to incorporate redundancy and self-checking capability, while still optimizing the speed, power and area of a design. The default action on detection of a fault is to sound an alarm, halt the system and drive the outputs to a safe operational state. A simple example is the relay logic in a fire control system, where the design must sound the alarm when a fire occurs, but must also do so under any condition that would de-activate the fire alarm system, such as a broken wire. In more complex system architectures, the detection of a failure that leads to an unsafe state usually requires the insertion of redundant components.

Redundancy simplifies the task of fault detection by utilizing comparators, which monitor the outputs of the redundant circuits at runtime and sound the alarm when a difference is detected. The primary drawback of redundancy is the significant increase in the size of the design and, given the large overhead, duplication with compare (DWC) [1] [2] is preferred over triple-modular redundancy (TMR), which limits the functionality to detection only, and requires external actions at the system level to correct the error(s).

The challenges associated with building fail-safe systems make it difficult to use automated circuit design flows that implement fail-safe properties while minimizing area overhead. In this paper, we investigate fail-safe design on FPGAs at the circuit level using an open source FPGA synthesis tool called OpenFPGA [3] [4]. FPGA artifacts that enable reconfigurability increase the difficulty of ensuring the redundant components are independent, e.g., the possibility of fault propagation through unused programmable resources needs to be considered. Moreover, failure mechanisms, such as upsets to the configuration state, can change logic functions and routing characteristics and therefore the self-checking capabilities of the monitoring circuit are more complex.

We first present a fail-safe DWC circuit as an ASIC and investigate the changes required to integrate a similar circuit into an FPGA. We utilize open-source FPGA design tools because the analysis of fault propagation paths requires detailed knowledge of the underlying reprogrammable circuit structure and it is considered proprietary and not available to end users in commercial FPGAs.

II. Background

A DWC technique is proposed for FPGAs in [1] [2] for detecting upsets in state, i.e., bit values stored in flip-flips (FFs), block RAM (BRAM) and configuration memory. The method duplicates the functional design and incorporates self-checking comparators that flag differences in duplicated signal components, e.g. primary outputs and those within feedback paths of the design.

A fault-tolerant technique that utilizes the dual outputs of 6-input LUTs to duplicate a logic function is proposed in [5]. Based on the effectiveness of the logic masking, the two duplicated outputs are either ANDed or ORed together in a downstream LUT, as a means of masking 0-to-1 and 1-to-0 single event upsets (SEUs), respectively. For example, a 5-input function is fully replicated in the lower 32 bits of the configuration memory bits (CMB) of the LUT. An integer linear program is proposed to finalize the optimal duplication and encoding scheme that attempts to minimize fault rate.

A robust technique for mitigating SEUs is proposed in [6] that leverages the dual outputs of 6-input LUTs and the carry chain. Logic functions are decomposed into two subfunctions, which are combined in the carry chain using AND or OR gates by controlling the carry-in to 0 and 1, respectively.

The technique does not require changes to the placement and routing, unlike [5] where downstream AND and OR functions are needed, therefore reducing routing congestion while adding resilience to SEUs.

The authors of [7] determined that CMB for routing resources represents 90% of the total number of CMBs in the FPGA, and are therefore are highly vulnerable to SEUs. Moreover, they found that about 10% of the SEUs that upset routing CMBs produce multiple short and open faults on PIPs that cannot be corrected by TMR because the single fault assumption on which TMR is based is violated. They propose a reliability-oriented place and route algorithm that prevents multiple errors from a PIP fault impacting TMR effectiveness.

The authors of [8] characterize SEU faults in FPGA LUTs and their interconnects. LUT SEUs only introduce a fault when that cell is selected, but configuration logic blocks (CLBs) have intra-CLB routing that is typically fully connected and MUX-based, so SEUs will always cause an irrelevant signal to be selected and a fault introduced. Inter-CLB routing is typically interconnected via bidirectional pass transistors. SEU disconnection (open) faults are typically modeled as temporary stuck-at-0/stuck-at-1 faults because FPGA architecture will pull them low/high to prevent crowbar current in downstream gates. SEU short faults bridge two adjacent wires and are the most complex to characterize since they are created when two drivers drive opposite values and downstream gates interpret them as different logic signals. Nets that fanout complicate this further, providing the opportunity for multiple faults to be injected.

Modern FPGAs possess very large configuration bitstreams, which increases the amount of time required to find errors using traversal scrubbing techniques. A rapid scrubbing technique is proposed in [9] which utilizes the mapping between critical circuits protected with DWC and configuration frames.

In this paper, we address the challenging issue of designing and implementing fail-safe circuits within the fabric of an FPGA. The reconfiguration capability of an FPGA adds complexity to fail-safe design strategies by introducing additional fault conditions and fault propagation paths, via CMBs in the wiring, PIPs (connections and switches) and CLBs. A DWC technique similar to that proposed in [2] is used, but optimized to minimize routing resources in the FPGA. Moreover, the proposed DEFCON scheme provides quad-redundancy in the XOR-based comparators, enabling continued system operation if a fault occurs in the fail-safe monitor itself, with no additional overhead. Last, a circuit level analysis using the netlists provided by OpenFPGA is carried out which considers fault effects and propagation at a finer level of granularity than the work described in [2].

III. Fail-Safe Design in Custom-Integrated Circuits

When considering fail-safe at the lower layers of the design hierarchy, e.g., logic gates and routing networks, a fail-safe design strategy needs to ensure the structural integrity of the components is maintained. One approach to accomplishing this goal is to replicate the system component, e.g., microprocessor using a dual-modular-redundancy (DMR) scheme, and add circuit monitors that trigger an alarm when output signals from the redundant system components differ. However, this approach does not protect against faults that occur in the monitors themselves. Full protection requires a specialized monitor that is self-checking.

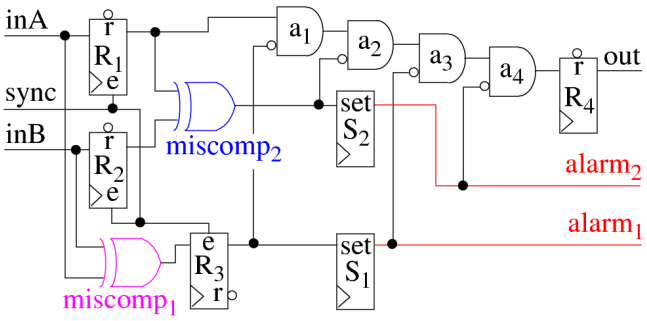

Fig. 1: ASIC schematic of the duplication-with-comparison (DWC) fail-safe circuit monitor.

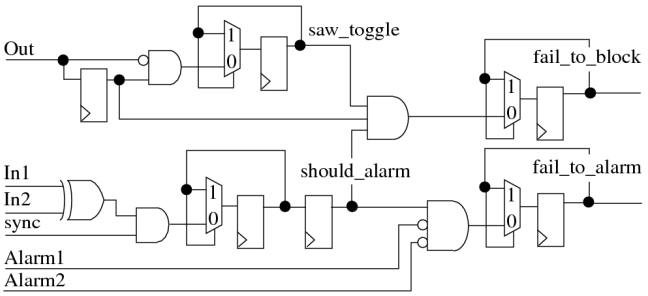

Fig. 2: OneSpin test wrapper.

As an example, Fig. 1 shows a schematic of a DWC circuit with self-checking capabilities. The DWC design is a simple comparator to illustrate the concept, with two redundant masters, *InA* and *InB*, writing data to a common slave, *Out*, over a serial interface. The serial data is sampled at synchronization points, i.e., when *sync* is asserted, and compared. If *InA* and *InB* both agree, then the data from one of the inputs passes through all four blocking AND gates to *Out*. Otherwise, it is blocked, and an alarm is triggered.

The proposed circuit structure detects faults that occur both within the system components driving the inputs and internally. For example, if a fault occurs within one of the system components and the inputs differ, then the XOR gate driving the input of the *Miscomp1* FF will assert. On the next clock, the *Miscomp1* FF output will be asserted, forcing a zero on the *Out* signal (the fail-safe state of this signal) via the chain of AND gates shown along the top of the figure. The *Alarm1* output will also be asserted after two clock cycles.

The remaining schematic components are designed to detect faults within the monitor itself. For example, if the XOR gate driving the input of *Miscomp1* is faulty, then the redundant XOR gate *Miscomp2* will detect a difference on the inputs (if one occurs) and force *Out* to 0, and then assert *Alarm2*.

Note that the redundancy does not guarantee correct oper-

ation of the circuit, only that the fail-secure property is maintained under any single fault. Once the circuit has detected an alarm, both alarm signals remain asserted and the output remains blocked until the asynchronous *rstN* signal is asserted low, resetting the circuit.

A formal fault analysis tool suite, called OneSpin [10], is used to carry out a fault analysis of this circuit. Because the circuit employs detection only and not correction, fault analysis will result in many false positives if just comparing the outputs to the fault-free case. We therefore create the wrapper shown in Fig. 2 to detect unauthorized events and assert one of two flag signals.

- Signal *fail_to_block* asserts if the output is able to toggle following a miscompare, indicating the data is not being blocked.
- Signal *fail_to_alarm* asserts if neither Alarm1 nor Alarm2 assert and remain asserted following a miscompare.

Under normal conditions, these signals will never assert. We confirmed this is the case using OneSpin's DV-Verify, which proves that when no fault is present, it is not possible for the logic of the circuit to assert either of the two signals. Only by forcing one of these unauthorized events to occur in simulation were we able to see them assert.

With this logic in place, OneSpin's Fault Propagation Analysis (FPA) application confirms that no fault on any of the DWC internal signals can propagate to the *fail_to_block* or *fail_to_alarm* outputs. As a sanity check, we tested again with each of the blocking gates bypassed, one at a time. In each case, OneSpin detected the broken redundancy by proving that a fault on a different gate could propagate to the unauthorized event signals.

A. OpenFPGA Framework

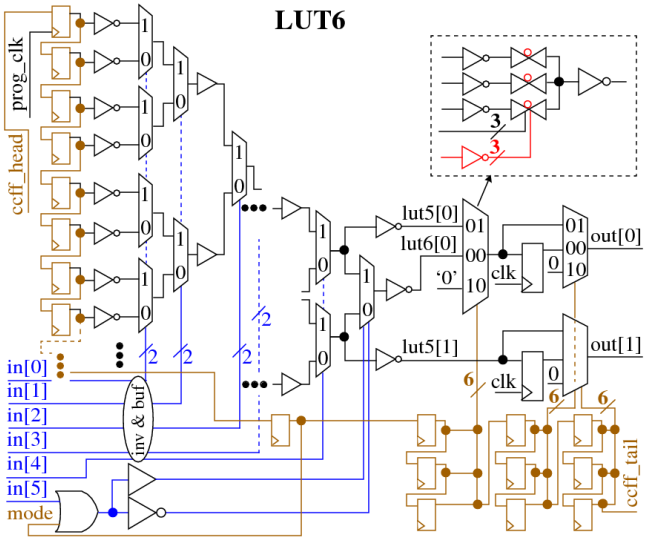

Fig. 3: OpenFPGA 6-input LUT with register logic within the CLB.

The OpenFPGA synthesis framework is used to create the FPGA architecture for the analysis carried out in this paper

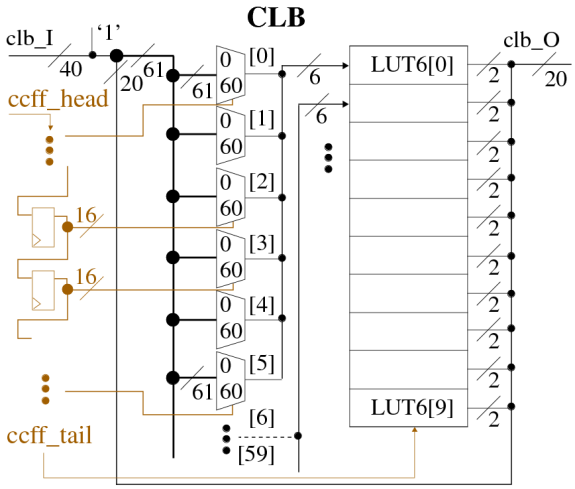

Fig. 4: CLB architecture created by OpenFPGA, with 10 LUT6 and local routing.

[3] [4]. The python-based tool flow utilizes an XML-based architectural description to specify the details of the CLBs, LUTs, FFs, routing architecture, I/O and custom hardwired IP blocks. The verilog-to-routing (VTR) CAD tool [11] is used within OpenFPGA to generate a verilog netlist. The netlist can then be processed into a layout using a standard cell library-based place-and-route (PNR) CAD tool flow.

Although the OpenFPGA synthesis tool supports a wide range of components, we utilize only the CLBs, routing components and I/O in the development of the proposed fail-safe circuit design. The specific implementation characteristics of the CLBs, which include LUTs and a local routing network, as well as the PIPs that define the global routing network, are needed to fully evaluate fault effects and fault propagation. This section describes the circuit structure created by OpenFPGA using a small example FPGA configuration test case provided in the distribution. The test case includes ten 6-input LUTs, which are enclosed within one CLB, four connection and switch blocks and a set of 32 I/O.

A schematic of the 6-input LUT (LUT6) is shown in Fig. 3. The programming bitstream configures the column of FFs on the left side with a logic function using the configuration chain (ccff_head), and other elements highlighted in brown. A sequence of transmission-gate 2-to-1 MUXs are used to implement the look-up table, with columns of amplifying buffers inserted after every two stages of MUXing. The LUT6 can be programmed to provide an upper-half LUT5 or LUT6 function on out[0] and a lower-half LUT5 function on out[1]. The two outputs can optionally be registered. The circuit structure of the LUT6 makes the lut5 outputs structurally independent except for the shared inputs.

The CLB architecture is shown in Fig. 4. The configurable routing network allows any of the LUT6 outputs to be connected locally with any of the LUT6 inputs. The 61-to-1 MUXs add 40 external inputs from other CLBs and I/O, and a constant '1', as configuration options for the LUT inputs. All routing within the CLB is fan-out free, and there are no instances of reconvergent-fanout in the circuit structures.

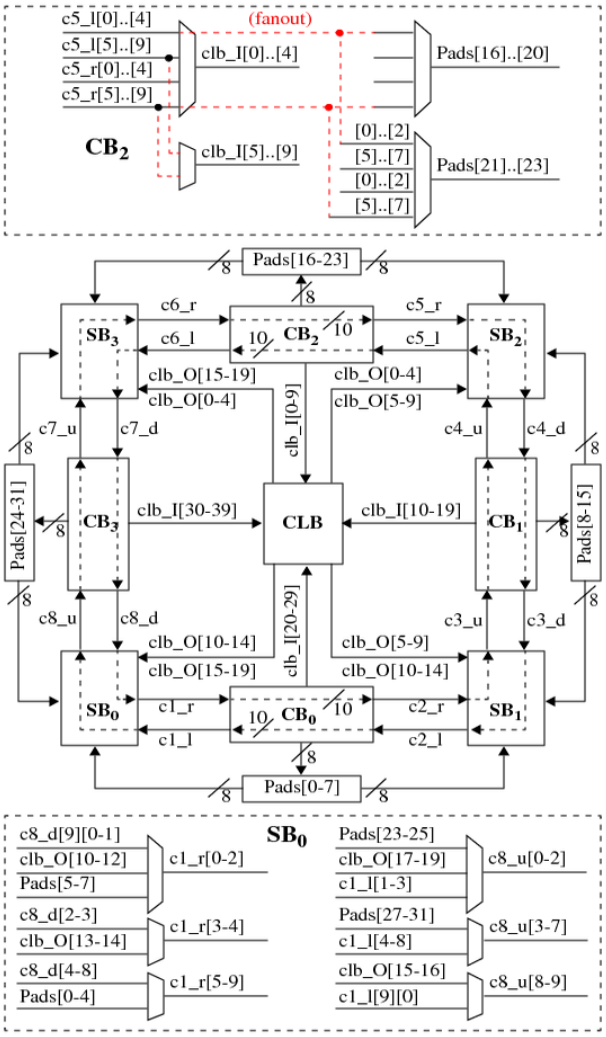

Fig. 5: Routing architecture created by OpenFPGA.

The routing architecture of the test case is shown in Fig. 5, which shows a single CLB instantiated in the center of the figure surrounded by a set of four connection blocks ($CB_{0,3}$) and switch blocks ($SB_{0,3}$). The MUXing details of the CB and SB shown above and below the routing architecture, respectively, illustrates that signal fanout is implemented in the CBs, and the CB and SB provide only a partial set of interconnect options. The I/O block includes output Pads driven from the CBs and input Pads which enter the routing network via the SBs.

B. Fail-Safe Design in an Open-source FPGA

In this section, we describe a DWC circuit optimized for an FPGA and evaluate its effectiveness in providing fail-safe operation, i.e., whether it is able to detect a fault on the outputs of the functional units it protects, and within the monitor itself. The design ensures that if a fault occurs in the functional paths or CMB, the alarm signal(s) will assert at most two clock cycles after the fault activation, and will detect faults that are active for one or more clock cycles. Another design goal of the monitor is to minimize FPGA resource utilization.

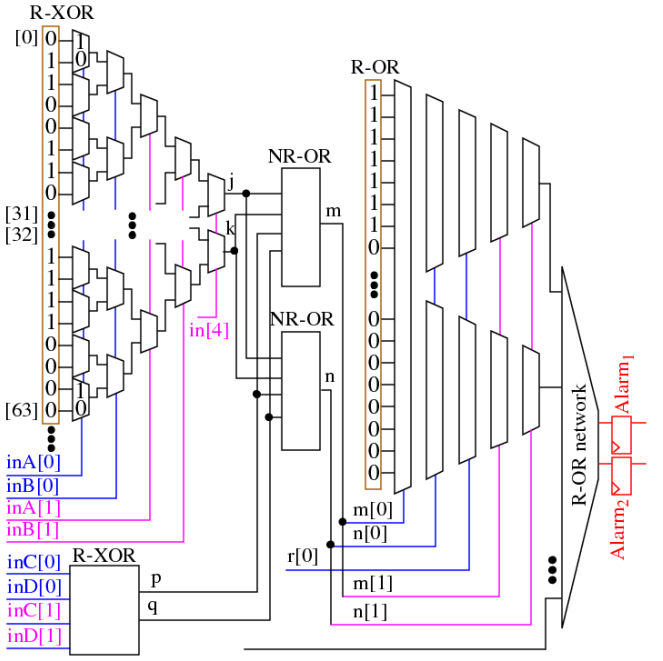

Fig. 6: Proposed DWC fail-safe circuit.

1) XOR Checker Component: A schematic of the proposed DWC circuit is shown in Fig. 6. The DWC circuit consists of a set of redundant 2-input XOR gates (R-XOR), each designed to monitor a pair of redundant signals, inA and inB. The inA and inB connect to the same outputs of a pair of redundant functional units (not shown). The functional units might be, e.g, a pair of ECUs in a vehicle and the output signals might be control signals to the anti-lock brake system or a driver assistance feature. The pair of non-redundant ORs (NR-OR) components connect to R-XOR outputs, which serve to merge the alarm signals from up to three R-XORs.

The LUT6 labeled R-XOR in the upper left of the figure is programmed to implement a special version of a 2-input XOR gate. The inA and inB signals from the functional units are fanned out locally, i.e., within the CLB shown in Fig. 4, to drive four of the LUT inputs. As mentioned earlier, the hierarchical MUX structure defines two independent logic cones for the out[0] and out[1] signals from Fig. 3, labeled here as j and k, but the inputs are not independent. For example, if a fault occurs on $inA[0]$ within the R-XOR (checker fault), both cones of logic will be affected.

A workaround is to program the LUT6 function with a specific pattern that implements the 2-input XOR truth table multiple times. Here, the upper 32 CMB bits of the LUT repeat the 2-input XOR functional output pattern “0110”. Although only the first 8 cells are shown, the pattern is in fact replicated 8 times as “0110 0110 0110 0110 0110 0110 0110 0110”. The lower 32 CMB bits of the LUT are also programmed to implement a 2-input XOR but locally replicate each truth table value four times. The full sequence for the lower 32 CMB is “0000 1111 1111 0000 0000 1111 1111 0000”.

The specified configuration bit pattern removes the dependency of the inputs on both logic cones, and allows a single

LUT to implement a fully self-checking comparator, i.e., one LUT for each duplicated functional unit output signal that is monitored. Here, only inputs $inA[0]$ and $inB[0]$ affect the functional output value of the top cone, and only inputs $inA[1]$ and $inB[1]$ affect the output of the bottom cone. For example, if inA and inB are "01", i.e., a faulty condition for the functional units, and a stuck-at-1 occurs on $inA[0]$, then the bottom cone will propagate a '1' to n because $inA[1]$ and $inB[1]$ drive the inputs to an independent XOR function. Therefore, the fault cannot be masked. Also note that the proposed redundancy scheme also detects faults that occur in the CMB. For example, if a CMB flips from '1' to '0' in one of the XOR pattern sequences of either logic cone, and a second fault occurs in one of the monitored functional units, the redundant logic cone will prevent the fault from being masked. Last, the single LUT implementation reduces the routing congestion associated with the self-checking XOR gate, in comparison to the scheme proposed in [2].

As noted in Fig. 3, the $lut5[0]$ signal passes through an additional 3-to-1 MUX. The fail-safe configuration of the $in[5]$ and $mode$ signals is '1', and for the select inputs of the 3-to-1 MUX, it is "01". A fault on either $in[5]$ or $mode$ is masked and made harmless by the 3-to-1 MUX configuration. A CMB fault on the 3-to-1 MUX select signals is ignored for the faulty case "00" since the $in[5]$ and $mode$ signal assignments select the output of the upper logic cone on both the "01" and "00" inputs. If, on the other hand, a CMB fault selects the hardwired '0' on the "10" input, the fault will be masked. This is the only fault that can lead to a failure in the DEFCON scheme.

The two outputs of the LUT6 can be registered as shown on the far right of Fig. 3. The fail-safe configuration for the 2-to-1 select MUXs is 0, which allows the $lut5$ signals to bypass the registers. If a fault occurs and the registered input is selected, the occurrence of a second upstream fault will be detected but with one cycle of latency.

2) OR Merge Component: The pair of NR-OR components in Fig. 6 are designed to implement a 6-to-1 compression function. Unlike the R-XOR, the inputs fan-out to different LUTs to implement DWC functionality, which adds routing congestion. However, this configuration provides high levels of compression for the alarm signals, which reduces the number of OR gates and the overall system overhead. An alternative R-OR, shown on the right side of the figure, is designed in the same fashion as the R-XOR gate but with five inputs, i.e., m and n fanout out locally to four of the R-OR inputs with $r[0]$ driving the fifth input. The CMB is again programmed with a special version of the OR function that allows $m[0]$, $n[0]$ and $r[0]$ to control the upper logic cone output and $m[1]$ and $n[1]$ to control the bottom cone output. Similar to the analysis of R-XOR, the R-OR guarantees an alarm signal will be asserted if any incoming signal is a '1' even if an internal node is stuck-at-0 or a CMB flips from '1' to '0'.

The NR-OR and R-OR network component defines a hierarchy of 6-to-1 and 2.5-to-2 compression functions, which eventually combine all of the R-XOR redundant outputs to two alarm signals, Alarm_1 and Alarm_2. The number of gates and

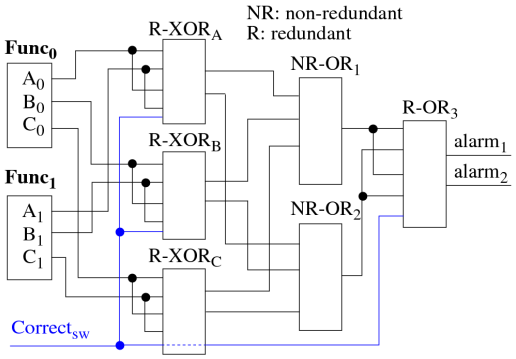

Fig. 7: Simulation model for validation with fail-safe checks on three functional unit output signal pairs.

levels in the hierarchy depends on the number of functional unit output signals monitored, and the number of NR-OR vs R-OR gates used for compression of the alarm signals.

C. Discussion

One benefit of keeping redundant copies in the same LUT, in contrast to the architecture proposed in [1] [2], is the avoidance of CB and SB routing faults. Although a fault can upset a CMB in the CLB shown in Fig. 4, and select the wrong input for a redundant input wire to an R-XOR, the fault will be detected, in contrast to the more complex fault behavior that occurs in SBs where faults in bi-directional MUXs can lead to shorts between multiple redundant wires and lead to fault masking.

A second notable advantage of the proposed architecture is the quad redundancy that exists within each R-XOR LUT. This quad redundancy allows the fail-safe system to mitigate faults at runtime that occur in the R-XOR LUTs. If for example, only one of the Alarm signals is asserted, this is usually an indication of a fault occurring in the DWC network. The signal labeled $in[4]$ in Fig. 6 can be toggled to verify whether an R-XOR fault has occurred by switching to the redundant copy in each of the two LUT subfunctions, i.e., switch to CMB bits [16] through [31] in the upper half and [48] through [63] in the bottom half of the LUT. If the asserted Alarm signal returns to 0, then the proposed fail-safe system can begin a scrubbing operation on the R-XOR CMB while continuing to operate with full fail-safe detection capabilities in place.

D. Analysis

The faults considered in our analysis are upsets in the CMB bits. Note that stuck-open and stuck-closed PIP faults are equivalent to stuck-at faults in the CMB that control the PIPs [12]. Bridging faults between independent wire segments (not connectable via a PIP) are not considered. Moreover, the PIP components in the OpenFPGA architecture are unidirectional which eliminates the need to consider the complex fault propagation behavior that results from using bidirectional PIPs [9].

We analyze the fault detection capabilities of DEFCON by simulating faults in the structural netlist representation of the OpenFPGA architecture described above. The length of the CMB chain is 2,562 bits. The simplest approach to emulating

TABLE I: Fault Detection Results for DEFCON

State	Both	Alrm0	Alrm1	Missed	Loops	Fixable
Fault-free	16	18	18	-	0	6
Faulty A	4875	4998	4999	10	15	0
Faulty B	4875	4998	5000	9	15	0
Faulty C	4872	4996	4997	11	16	0

faults is to flip each of the CMB bits, one-at-a-time, and count the number of times each of the Alarm signals is asserted in the 2,562 simulation experiments. Unfortunately, the one-hot, pass-gate MUX implementation approach taken within the OpenFPGA framework (see callout in Fig. 3) causes a large number of undefined (X) and high impedance (Z) results making it difficult to evaluate the effectiveness of DEFCON, because, under fault conditions, wires can float, i.e., have no drivers, or can have multiple drivers.

As a work-around, we create a fault model which emulates logic gate-based MUXs, as a means of eliminating the the undefined and high impedance simulation results from one-hot circuit configurations. This increases the number of faults in the model to 5,132 because 2-level MUXs used in the netlist architecture expand the number of valid assignments. A second well known challenge with simulating FPGA designs is the presence of combinational loops, which can occur under certain fault conditions due to the flexibility in the routing network. The results presented below therefore include the number of faults that could not be assessed because of simulation failure.

The results are shown in Table I for the four test scenarios shown on the rows. Fault-free refers to consistent assignments to the two functional unit outputs, e.g., A_0 and A_1 are set to "00" or "11", while Faulty A refers to inconsistent assignments, e.g., "01" and "10". Although there are 8 possible configurations for each row in our simulation model with three functional outputs, we show results for only one configuration, e.g., A_0, A_1, B_0, B_1, C_0, C_1 = "000000" because similar results were obtained for the remaining configurations. Also note that the faulty configuration experiments introduce two faults into the model and therefore, provide data on the effectiveness of the technique when faults occur simultaneously in a functional unit and in the DEFCON monitor.

The Fault-free results show that only 20 faults cause one or both of the Alarm signals to be asserted. Given these alarm conditions are not linked to functional unit output failures, but rather are associated with faults in the DEFCON monitor itself, a small number of fault assertions is a desirable feature. Of the 20 fault assertions, the $Correct_{sw}$ is able to instantly repair all 6 faults that occur within the R-XOR gates. In contrast, a fault on a functional unit output A_0, A_1 = "01" output is detected in 5,122 of the 5,132 cases. Therefore, DEFCON fails to detect a functional unit failure for 10 faults that occur in the monitor. For B_0, B_1 and C_0, C_1 = "01", only 9 and 11 faults are missed, respectively. The fail cases result when a stuck-at-0 fault occurs in the functional unit output routing network, preventing the downstream R-XOR checker from detecting a difference. These fail scenarios can be addressed by duplicating the functional unit outputs at the expense of doubling the number of R-XOR checkers. Note that the $Correct_{sw}$ has no effect when the functional unit outputs are faulty because it is only designed to repair monitor faults under fault-free conditions.

IV. Conclusion

In this paper, we propose a compact, fail-safe design strategy for FPGAs that is able to detect single fault occurrences at runtime in the LUTs, switches and routing of an FPGA, and forces the output state of the protected circuit to output values that represent a fail-safe condition for the system. We refer to the technique as DEFCON (DEsign for Fail-safe in reCONfigurable systems). Simulation results are presented that illustrate the effectiveness of the approach.

Acknowledgment

This R & D was supported by the Laboratory Directed Research and Development (LDRD) program at Sandia National Laboratories (LDRD # 225963). Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525.

References

[1] J. Johnson, W. Howes, M. Wirthlin, D. L. McMurtrey, M. Caffrey, P. Graham, and K. Morgan, "Using duplication with compare for on-line error detection in fpga-based designs," in *2008 IEEE Aerospace Conference*, 2008, pp. 1–11.

[2] D. L. McMurtrey, "Using duplication with compare for on-line error detection in fpga-based designs," in *Theses and Dissertations*, vol. 1094, 2006.

[3] X. Tang, E. Giacomin, A. Alacchi, B. Chauviere, and P.-E. Gaillardon, "Openfpga: An opensource framework enabling rapid prototyping of customizable fpgas," in *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, 2019, pp. 367–374.

[4] X. Tang, E. Giacomin, B. Chauviere, A. Alacchi, and P.-E. Gaillardon, "Openfpga: An open-source framework for agile prototyping customizable fpgas," *IEEE Micro*, vol. 40, no. 4, pp. 41–48, 2020.

[5] J.-Y. Lee, Y. Hu, R. Majumdar, L. He, and M. Li, "Fault-tolerant resynthesis with dual-output luts," in *2010 15th Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2010, pp. 325–330.

[6] J.-Y. Lee, Z. Feng, and L. He, "In-place decomposition for robustness in fpga," in *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2010, pp. 143–148.

[7] L. Sterpone and M. Violante, "A new reliability-oriented place and route algorithm for sram-based fpgas," *IEEE Transactions on Computers*, vol. 55, no. 6, pp. 732–744, 2006.

[8] N. Jing, J.-Y. Lee, Z. Feng, W. He, Z. Mao, and L. He, "Seu fault evaluation and characteristics for sram-based fpga architectures and synthesis algorithms," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 18, no. 1, jan 2013. [Online]. Available: https://doi.org/10.1145/2390191.2390204

[9] S. Zheng, H. You, G. He, Q. Wang, T. Si, J. Jiang, J. Jin, and N. Jing, "A rapid scrubbing technique for seu mitigation on sram-based fpgas," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 1–5.

[10] "Onespin," *https://www.onespin.com/, accessed on Sept. 23, 2022*, 2022.

[11] J. Luu, J. Goeders, M. Wainberg, A. Somerville, T. Yu, K. Nasartschuk, M. Nasr, S. Wang, T. Liu, N. Ahmed, K. B. Kent, J. Anderson, J. Rose, and V. Betz, "Vtr 7.0: Next generation architecture and cad system for fpgas," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 7, no. 2, jul 2014.

[12] J. Yao, B. Dixon, C. Stroud, and V. Nelson, "System-level built-in self-test of global routing resources in virtex-4 fpgas," in *2009 41st Southeastern Symposium on System Theory*, 2009, pp. 29–32.