Designing Fault-Tolerant Techniques for SRAM-Based FPGAs

Fernanda Gusmão de Lima Kastensmidt
State University of Rio Grande do Sul

Gustavo Neuberger, Renato Fernandes Hentschke, Luigi Carro, and Ricardo Reis
Federal University of Rio Grande do Sul

Editors' note:
FPGAs have become prevalent in critical applications in which transient faults can seriously affect the circuit's operation. This article presents a fault tolerance technique for transient and permanent faults in SRAM-based FPGAs. This technique combines duplication with comparison (DWC) and concurrent error detection (CED) to provide a highly reliable circuit while maintaining hardware, pin, and power overheads far lower than with classic triple-modular-redundancy techniques.

—Dimitris Gizopoulos, University of Piraeus; and Yervant Zorian, Virage Logic

ICs ARE SENSITIVE to upsets that occur in aerospace. More recently, ICs have also become sensitive to upsets at ground level because of the continual evolution of fabrication technology for semiconductors. Drastic device shrinkage, power supply reduction, and increasing operating speeds significantly reduce noise margins and thus reliability because of the internal noise sources that very deep-submicron ICs face.[1] This trend is approaching a point at which it will be infeasible to produce ICs that are free from these effects. Consequently, fault tolerance is no longer a matter exclusively for aerospace designers; it's important for the designers of next-generation ground-level products as well.

FPGAs are popular for design solutions because they improve logic density and performance for many applications. SRAM-based FPGAs, in particular, are highly flexible because they are reprogrammable, allowing onsite design changes. However, because the reprogrammability leads to a high logic density in terms of SRAM memory cells, SRAM-based FPGAs are also sensitive to radiation and require protection to work in harsh environments.[2]

Our high-level fault tolerance technique combines time and hardware redundancy to cope with upsets in SRAM-based FPGAs. This technique reduces the number of I/O pads, and therefore power dissipation, in the interface compared to the well-known triple modular redundancy (TMR) solution. Our goal is to reduce hardware overhead (which is three times more in TMR than the original area of the unprotected design) to close to twice the original area, maintaining the same reliability and consequently reducing power dissipation. We've evaluated our technique in two types of circuits: multipliers and digital filters.

Radiation effects on SRAM-based FPGAs

A radiation environment contains various charged particles, generated by sun activity, that interact with silicon atoms, exciting and ionizing the atomic electrons.[3] At ground level, neutrons are the most frequent causes of upsets.[4] When a single heavy ion strikes the silicon, it loses its energy through the production of free electron-hole pairs, resulting in a dense ionized track in the local region. Protons and neutrons can cause a nuclear reaction when passing through the material. The recoil also produces ionization, generating a transient current pulse that can cause an upset in the circuit.

A single particle can hit either the combinational or the sequential logic in the silicon.[5] When a charged particle strikes a memory cell's sensitive nodes, such as a drain in an off-state transistor, it generates a transient current pulse that can mistakenly turn on the opposite

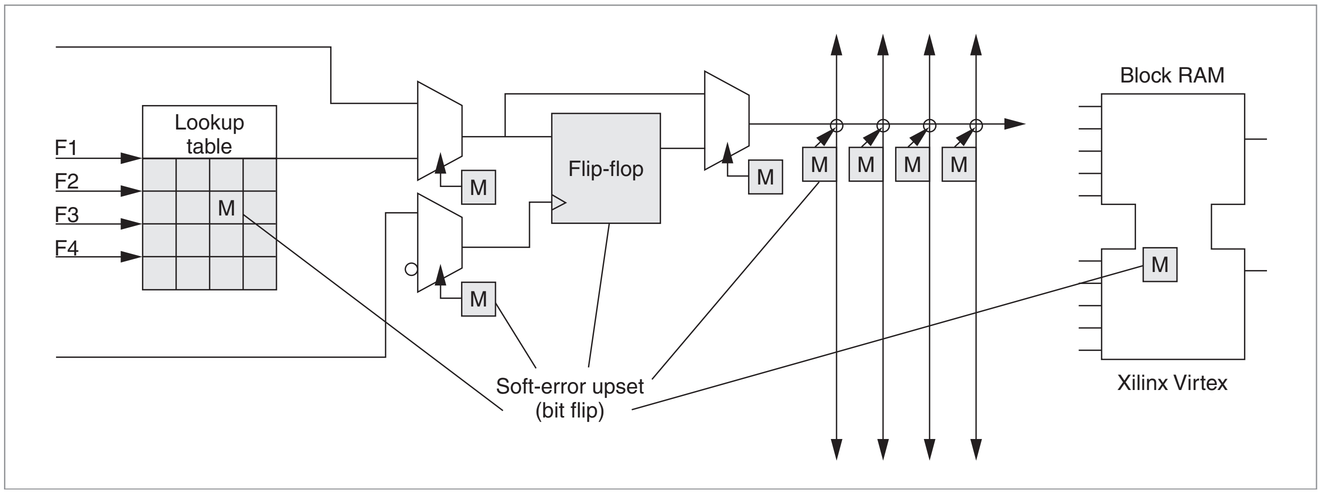


Figure 1. Bits sensitive to single-event upsets (SEUs) in the configurable-logic-block tile schematic. Inputs F_1 through F_4 are the four 1-bit input signals of the lookup table. M is the configuration memory cell.

transistor's gate. The effect can invert the stored value—that is, produce a bit flip in the memory cell. This effect is called a single-event upset (SEU) or soft error, and it's a major concern in digital circuits. When a charged particle hits the combinational logic block, it also generates a transient current pulse. This phenomenon is called a single-event transient (SET).

In FPGAs, an upset has a peculiar effect when it hits the combinational and sequential logic mapped into the programmable architecture. For example, consider SRAM-based FPGAs such as those from Xilinx's Virtex series, one of the most popular series of programmable devices on the market. Virtex devices include a flexible, regular architecture comprising an array of configurable logic blocks (CLBs) surrounded by programmable I/O blocks, all interconnected by a hierarchy of fast and versatile routing resources.[2] The CLBs provide the functional elements for constructing logic; the I/O blocks provide the interface between the package pins and the CLBs. A general routing matrix interconnects the CLBs. This matrix includes an array of routing switches located at the intersections of horizontal and vertical routing channels. Virtex devices also dedicate 4,096-bit memory blocks called block-select RAMs, clock delay-locked loops (DLLs) for clock-distribution delay compensation and clock domain control, and two tristate buffers associated with each CLB.

Users can quickly program a Virtex device by loading a configuration bitstream (a collection of configuration bits) into it. They can change device functionality at any time by loading in a new bitstream. The bitstream contains all the information to configure the programmable storage elements in the matrix located in the lookup tables (LUTs), flip-flops, and CLB configuration cells, and interconnections, as Figure 1 shows. All these configuration bits are potentially sensitive to SEUs; hence, we targeted them in our investigation.

In an ASIC, the effect of a particle hitting either the combinational or the sequential logic is transient; the only variation is how long the fault lasts. A fault in the combinational logic is a transient logic pulse in a node that can disappear according to the logic delay and topology. In other words, a storage cell might or might not latch a transient fault from the combinational logic. Faults in the sequential logic manifest themselves as bit flips, which remain in the storage cell until the next load. In an SRAM-based FPGA, customizable memory cells—SRAM cells (see Figure 1)—implement both the user's combinational and sequential logic. When an upset occurs in the combinational logic synthesized in the FPGA, it corresponds to a bit flip in one of the LUT's cells or in the cells that control the routing. An upset in an LUT memory cell modifies the implemented combinational logic, as Figure 2a shows. This upset has a permanent effect, and is correctable only at the next load of the configuration bitstream. This effect is similar to a stuck-at fault at 1 or 0 in the combinational logic defined by that LUT. Thus, a storage cell latches the upset from the FPGA's combinational logic, unless the FPGA uses some detection technique. An upset in the routing can connect or disconnect a wire in the matrix, as Figure 2b shows. It also has a permanent effect, which can travel to an open or a short circuit in the combinational logic implemented by the FPGA. The configuration bitstream's next load corrects this fault.

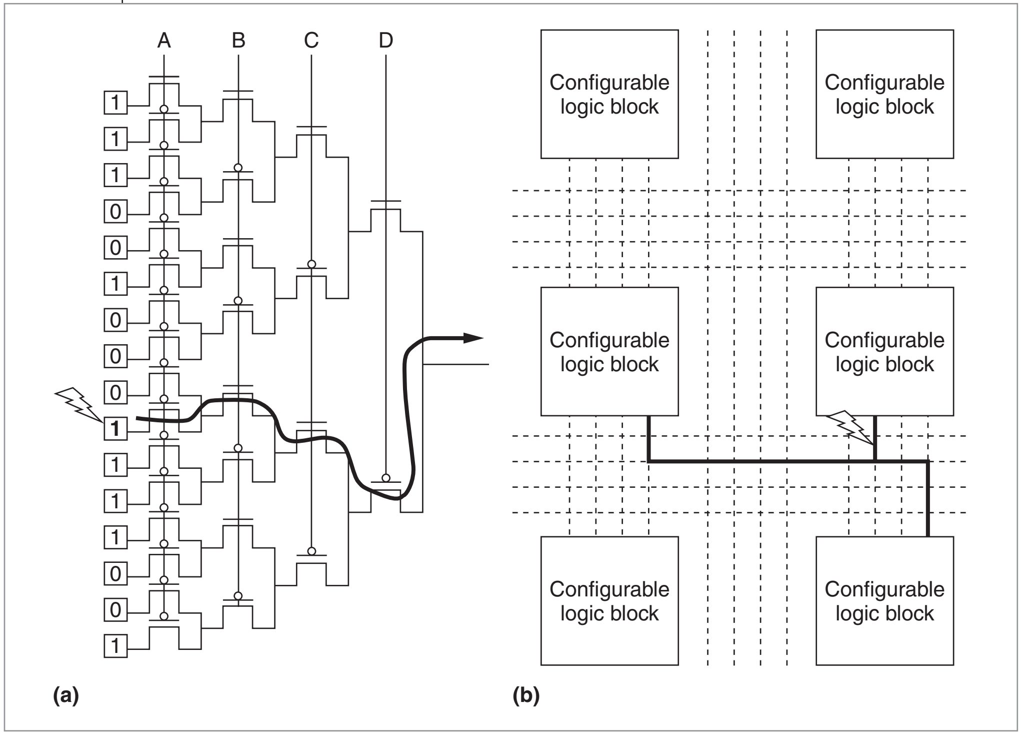


Figure 2. Example upsets in the SRAM-based FPGA architecture: an upset in the lookup table because of logic modification (a), and an upset in the routing because of an undesirable connection (b).

When an upset occurs in the user sequential logic synthesized in the FPGA, it has a transient effect because the CLB flip-flop's next load corrects it. An upset in the embedded block RAM has a permanent effect, and fault tolerance techniques must correct it. Engineers must apply these techniques in the architectural or high-level description, because the bitstream's load can't change the memory state without interrupting the application's normal operation. It's also possible to find, in the CLB, the SET upsets in the combinational logic, such as input and output multiplexers for routing control. (Rebaudengo, Reorda, and Violante also discuss the effects of upsets in the FPGA architecture.[6])

Radiation tests on Xilinx FPGAs show the effects of SEUs in the design application and prove the necessity for using fault-tolerant techniques in aerospace applications.[7] A fault-tolerant system designed into SRAM-based FPGAs must cope with the peculiarities just discussed: transient and permanent effects of an SEU in the combinational logic, short and open circuits in the design connections, and bit flips in the flip-flops and memory cells. Ohlsson et al. also analyzed the effect of neutrons in an SRAM-based FPGA from Xilinx.[8] At that time, FPGAs were not very susceptible to neutrons, but now as transistor size decreases and logic density increases, FPGAs are becoming more vulnerable.

Fault tolerance in SRAM-based FPGAs

There are two ways to implement fault-tolerant designs in SRAM-based FPGAs. The first is to design a new FPGA matrix of fault-tolerant elements. These new elements can replace the old ones in the same architecture topology, or you could develop a new architecture to improve robustness. Either way, the cost will be high, though it will vary according to development time, number of required engineers, and foundry technology. Another option is to protect the high-level description using redundancy, targeting the FPGA architecture. You could use a commercial FPGA to implement the design, and apply the SEU mitigation technique to the design description before synthesizing the redundant blocks in the FPGA. This approach is far less expensive than the previous one because here users are responsible for protecting their own designs; new chip development and fabrication are not necessary. Thus, the user can choose the fault tolerance technique, and consequently control the area, performance, and power dissipation overheads.

The high-level SEU mitigation technique used most often today to protect designs synthesized in the Virtex architecture is based mainly on TMR combined with scrubbing, which places a continuous load on the bitstream. The TMR mitigation scheme uses three identical logic circuits (redundant blocks 0, 1, and 2) synthesized in the FPGA. These circuits perform the same task in tandem, with a majority voter circuit comparing corresponding outputs. Details of the TMR technique for Virtex are available elsewhere,[9] and Lima et al. present more examples.[10] The correct implementation of TMR circuitry in the Virtex architecture depends on the type of data

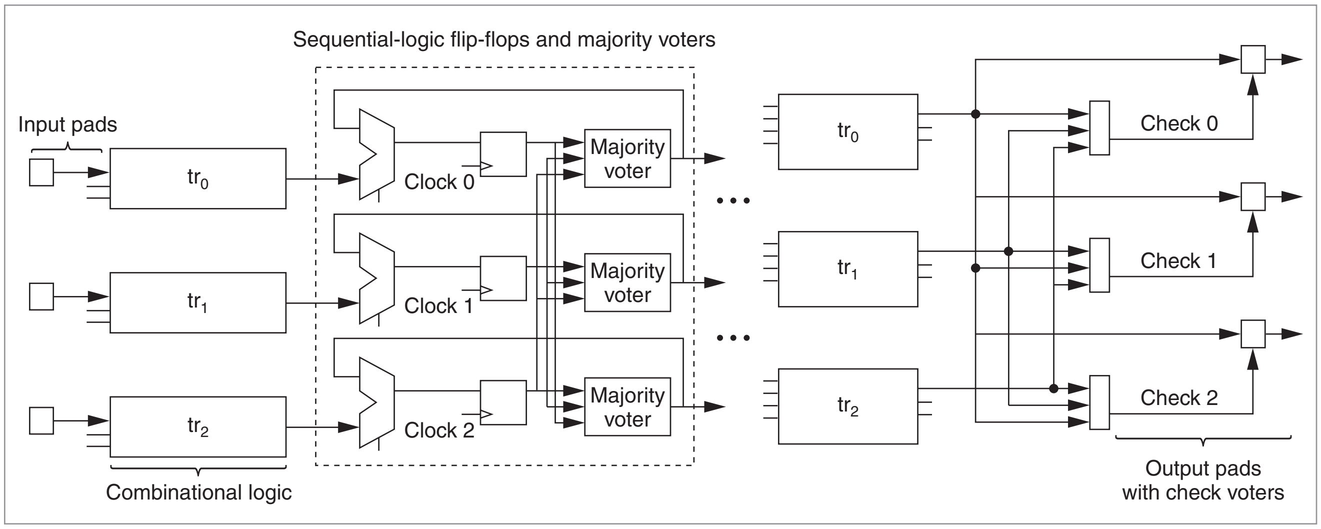


Figure 3. Triple modular redundancy for Xilinx FPGAs. The throughput logic is triplicated, represented by TMR combinational modules tr_0, tr_1, and tr_2. The registers are also triplicated and are voted on by majority voters; they also have a mechanism to correct upsets in the multiplexers.

structure you need to mitigate. Logic falls into four different structure types: throughput, state machine, I/O, and special features (select-RAM blocks, DLLs, and so on).

Throughput logic is a logic module of any size or functionality, synchronous or asynchronous, where all logic paths flow from the module's inputs to its outputs without forming a logic loop. In this case, all that's necessary is to triplicate the logic, creating three redundant logic parts (0, 1, and 2). No voters are required, because the FPGA output will be voted on later by default. State-machine logic is any structure where a registered output, at any register stage in the module, feeds back into any prior stage in the module, forming a registered logic loop. This structure is common in accumulators, counters, and any custom state machine or state sequencer in which each internal register's state depends on its own previous state. In this case, it's necessary to triplicate the logic and to have majority voters in the outputs. To ensure that a register doesn't lock on the wrong value, each redundant logic part in the feedback path has a voter so that the system can recover itself. One LUT can easily implement the majority voter. For designs constrained by available logic resources, you can implement the majority voter using Virtex tristate buffers rather than LUTs.

The primary purpose of using a TMR design methodology is to remove all single points of failure from the design. Therefore, each redundant part that uses FPGA inputs should have its own set of inputs. Thus, if an input suffers a failure, it affects only one of the redundant logic parts. The outputs are the key to the overall TMR strategy. Because full TMR generates every logic path in triplicate, it's necessary to bring these three logic paths back to a single path that doesn't create a single point of failure. You can do this by placing TMR output voters inside the output logic block. Figure 3 illustrates the TMR technique.

Scrubbing lets a system repair SEUs in the configuration memory without disrupting operations. The Virtex Select-MAP interface performs this scrubbing. When an FPGA is in this mode, an external oscillator generates a configuration clock that drives the programmable ROM (PROM) and the FPGA. At each clock cycle, new data is available on the PROM data pins. One example is the Flash PROM XQR18V04, which provides a parallel frequency of up to 264 Mbps at 33 MHz. The scrubbing cycle time depends on the configuration clock frequency and the readback bitstream size.

Previous results based on fault injection and radiation ground testing show the Virtex TMR design techniques' reliability.[8,11] However, the TMR technique has limitations, such as high area overhead, three times more input and output pins, and a significant increase in power dissipation. Many applications can accept these limitations, but some cannot.

Reducing TMR overheads by combining hardware and time redundancy

To reduce the number of pins used by the TMR approach and to deal with permanent upset effects, we present a new technique based on time and hardware

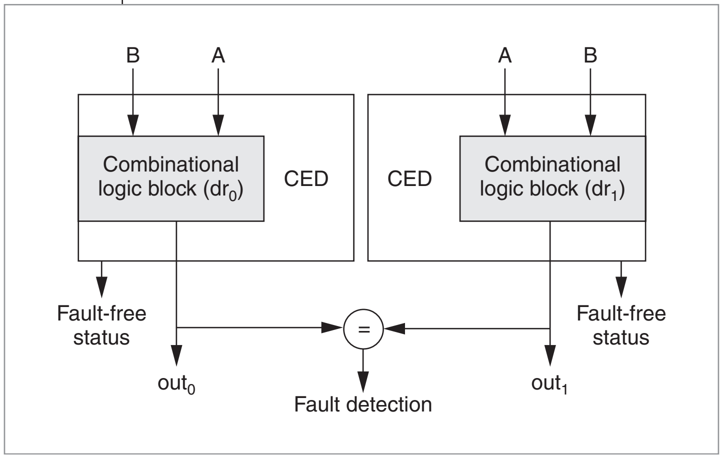


Figure 4. Duplication with comparison (DWC) combined with concurrent error detection (CED).

redundancy to protect the user's combinational logic. TMR still protects the sequential logic to avoid the accumulation of faults, because scrubbing doesn't change the content of a user's memory cell.

Lubaszewski and Courtois discussed TMR's reliability and safety compared to self-checking-based fault-tolerant schemes.[11] Their experiments indicate that the higher the module's complexity, the greater the difference in reliability between TMR and the self-checking scheme. The self-checking fault-tolerant scheme is more reliable than TMR if it does not exceed the self-checking overhead bound of 73%.

We extend the idea of using a self-checking fault-tolerant scheme to FPGAs. Our method combines duplication with comparison (DWC) with a concurrent error detection (CED) machine based on time redundancy that works as a self-checking block. DWC detects faults in the system, and CED detects which blocks are fault free. Figure 4 shows the general scheme. Two combinational logic blocks run simultaneously in the DWC technique: modules dr_0 and dr_1. A comparator in the output can detect a mismatch and signal a fault detection. If a mismatch occurs, the CED block evaluates whether the logic is fault free by analyzing the combinational logic's properties.

Researchers have proposed many methods for using CED blocks based on time redundancy to detect per-

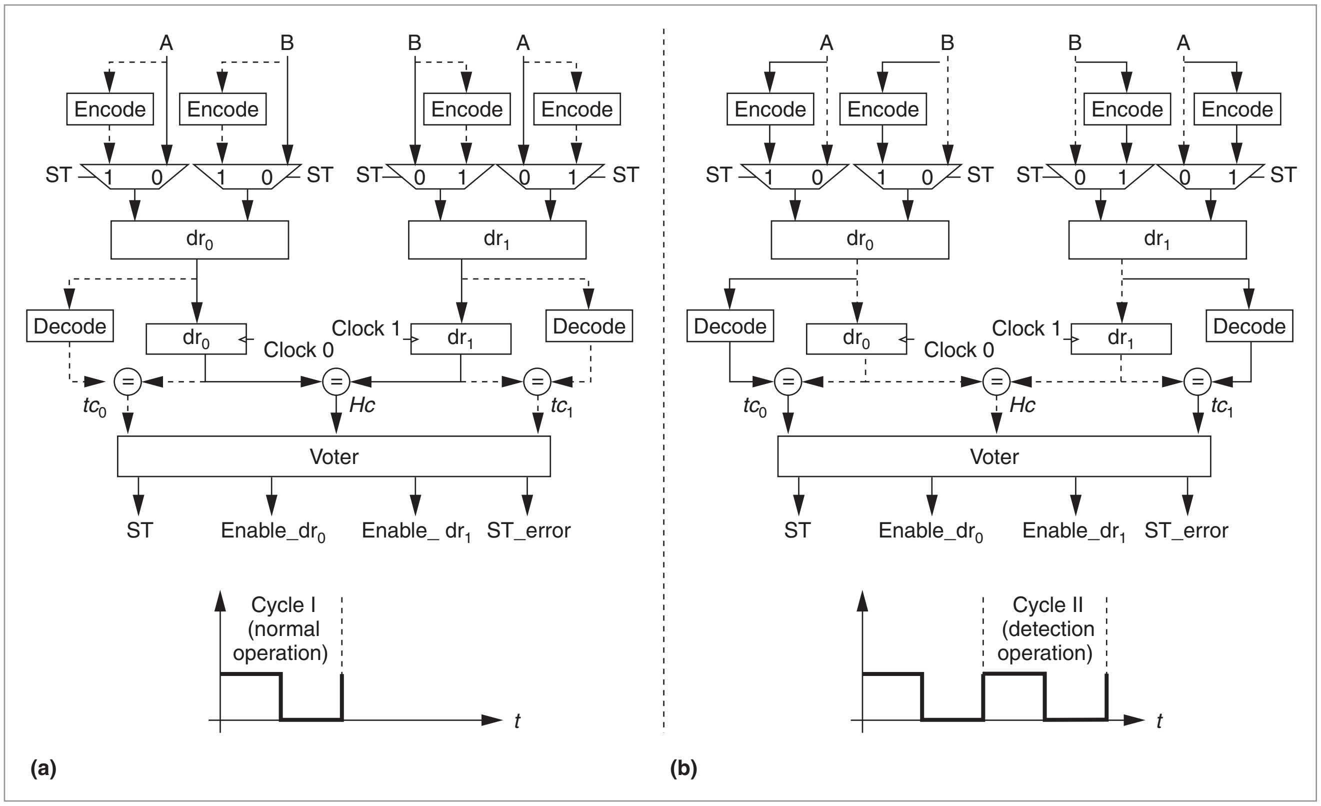


Figure 5. DWC combined with CED technique for SRAM-based FPGAs: normal (a) and fault detection (b) operation. *ST* is a state signal from the voter block that puts the system in detection operation (*ST* = 1); dr_0 and dr_1 are the combinational logic blocks; *Tc_0* and *Tc_1* are time comparisons; and *Hc* is the hardware comparison. The voter block also generates a state error signal (ST_error) and signals to enable the fault-free block (enable_dr_0 and enable_dr_1).

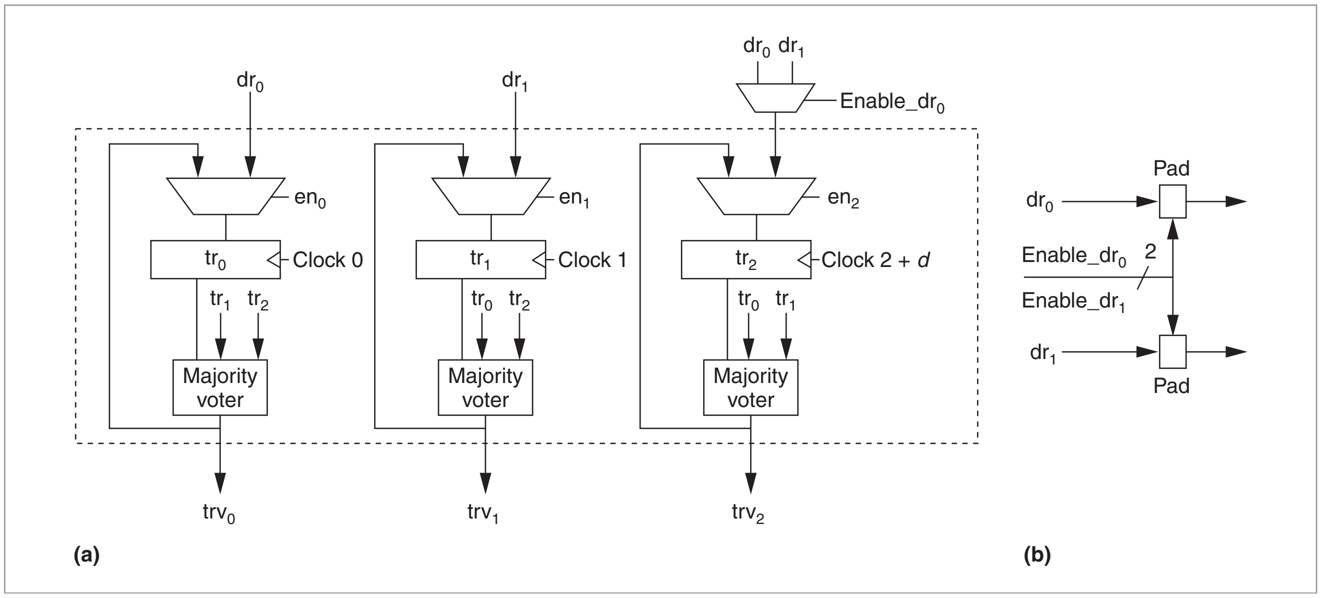


Figure 6. Example implementations when the combinational output is registered (a) or in the pad (b). Each majority voter block receives the signal from the tr_0, tr_1, and tr_2 registers. The enable_dr_0 and enable_dr_1 signals decide which fault-free blocks should pass through the logic output to the registers or to the pads. To improve reliability in routing, there are three enable signals (en_1, en_2, and en_3), each a 1-bit signal with a logic value of 0 or 1. The outputs from the majority voter blocks are trv_0, trv_1, and trv_2.

manent faults. These include bitwise inversion, recomputing with shift operands (RESO), and recomputing with swapped operands (REWSO). We implement the CED block using Patel and Fung's RESO technique.[12] This RESO method includes encoding and decoding blocks and a register.

During normal operation when time t_0, dr_0, and dr_1 are working simultaneously, the CED block stores the outputs in sample registers for further comparison, and the voter block continually compares the dr_0 and dr_1 outputs, as Figure 5a shows. If a mismatch occurs between these outputs, the output registers hold their original value for an extra clock cycle, while the CED block's RESO detects the fault. During this second clock cycle, the operands shift prior to use such that errors from permanent faults in the combinational logic are different in the first calculation than in the second. Comparing the results can identify these different errors, as Figure 5b shows. The encoding blocks are simple multiplexers, and the decoding blocks are simple connections.

For registered outputs, each output goes directly to the input of the user's TMR register. Figure 6a shows the logic scheme. Block dr_0 connects to TMR combinational module tr_0, and block dr_1 connects to module tr_1. While the circuit searches for faults, the user's TMR register holds its previous value. When the circuit finds the fault-free module, tr_2 receives that module's output, and continues receiving it until the next chip reconfiguration (fault correction). By default, the circuit starts passing the output of dr_0 to tr_2. For unregistered outputs, the circuit can drive the signals directly to the next combinational module or to the I/O pads, as Figure 6b shows.

The important characteristic of our method is that it doesn't incur a high performance penalty when the system has no faults or only a single fault. This method needs only one clock cycle in a hold operation to detect the faulty module; then it operates normally again without performance penalties. The final clock period is the original clock period plus the propagation delay of the encoders, decoders, and output comparator.

The voter block contains comparators and a small state machine to identify the operation's fault-free state or to signal an error. Figure 7 shows this logic's state diagram. The state machine's inputs are hardware comparison *Hc* and time comparisons Tc_0 and Tc_1, represented by the 2-bit signal, *Tc*. The state machine's outputs constitute a 4-bit vector (shown in Figure 7 after the slash) indicating the detection state (*ST*), the error state, enable_dr_0, and enable_dr_1. Signals enable_dr_0 and enable_dr_1 are used for the unregistered outputs (Figure 6b); when the output is registered, only enable_dr_0 is used (Figure 6a).

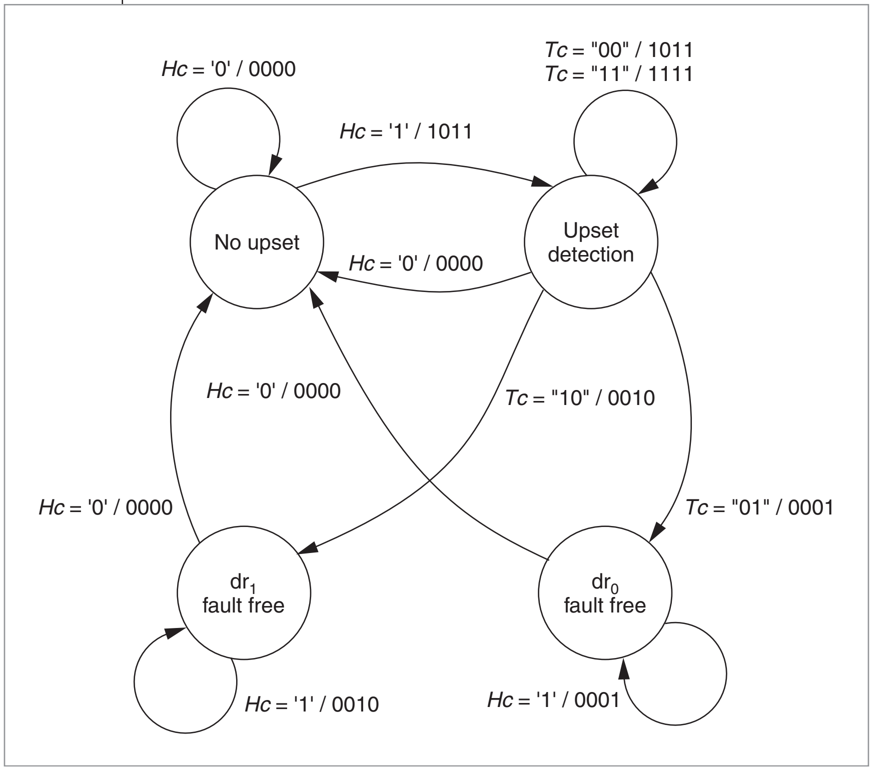


Figure 7. State diagram of the DWC-CED voter circuitry. Numbers after a slash indicate the 4-bit vector outputs. Numbers with single quotation marks indicate *Tc* or *Hc* values. Double quotation marks indicate that if the input is that value, the output is the one after the corresponding slash; if the input is another value (also indicated by double quotation marks), the output is the one after that corresponding slash.

In both TMR and our method, scrubbing corrects upsets in the user's combinational logic, and the CLB flip-flops' TMR scheme corrects upsets in the user's sequential logic. Scrubbing must be continuous to guarantee that only one upset has occurred between two reconfigurations in the design. Some constraints are necessary for our method to function properly, just as with TMR. First, there must not be upsets in more than one redundant module, including the state machine's detection and voting circuit. Consequently, we must use assigned area constraints to reduce the probability of short circuits between dr_0 and dr_1. Second, the scrubbing rate should be fast enough to avoid the accumulation of upsets in two different redundant blocks. Upsets in the detection and voting circuit don't interfere with the system's proper execution because the logic is already duplicated. In addition, upsets in this logic's latches are not critical, because they're refreshed every clock cycle. Assuming a single upset occurs per chip between scrubbing, it doesn't matter if an upset alters the correct voting, provided no upset occurs in both redundant blocks.

Experimental results

To evaluate our technique's fault coverage, we chose two arithmetic-based circuits: a multiplier and a canonical finite impulse response (FIR) digital filter. The developed tools automatically generated the multipliers and filters protected by DWC-CED. We evaluated these case study circuits in terms of fault coverage, area, performance, and power dissipation.

Fault coverage

We developed a fault coverage test system to evaluate the DWC-CED technique's robustness in the presence of upsets. The system automatically inserted structures to enable automatic fault injection in high-level descriptions, replacing all design nodes with one fault injection component, a 4-to-1 multiplexer, so that users can insert all types of faults and as many as necessary. If the multiplexer's select signal is 00, the original signal goes to the output; if the signal is 01, the output is a constant 0 (stuck-at-0 emulation); if the signal is 10, a constant 1 propagates (stuck-at-1 emulation).

For the first case study, we chose an 8-bit multiplier, along with a 9-bit multiplier to apply the RESO technique without losses in the most significant bit. We implemented multipliers using cascaded full adders. The 8-bit multiplier had 528 faulty nodes, 1,056 faults in total (stuck-at 0 or 1). The 9-bit multiplier had 675 faulty nodes, 1,350 faults in total. In both cases, the two original operands had 8 bits, resulting in 2^{16} (65,536) combinations of input vectors. We injected all combinations of faults and input vectors: 69,206,016 for the 8-bit multiplier, and 88,473,600 for the 9-bit version.

We chose a canonical digital FIR filter circuit for our second case study; the multipliers had constant coefficients, resulting in an optimized area and minimal faulty nodes. Our developed system automatically generated a 9-tap, 8-bit FIR canonical filter. The multiplier coefficients were 2, 6, 17, 32, and 38. Because of the 8-bit input, there were 2^8 (256) combinations of input vectors to test. The total number of faulty nodes in the FIR filter, including all multipliers and adders, was 4,208. We tested all possible combinations of input vectors and faults, a total of 1,077,248.

The system exhaustively injected the faults in all nodes of the test circuits for each input vector, sensitive node, and redundant blocks mult_dr$_0$ and mult_dr$_1$. The fault injection system operated with two clocks, one to control the change of input vectors, and the other to control the change of faults. A counter controlled the total number of combinations of input vectors and faults inserted in the circuit. We injected all possible combinations.

In all cycles, the voter block from the DWC-CED technique compared the outputs of modules dr$_0$ and dr$_1$. If the outputs were equal ($Hc = 0$), then a fault occurring in one of the circuits did not generate an error in the output. Therefore, for real-time operations, we could ignore this fault, and no detection operation was necessary. If a fault generated an error in the output ($Hc = 1$), the voter compared the output of dr$_1$ with the recomputing circuit's decoded output. If the outputs were not equal ($Tc_1 = 1$), the technique under test was able to detect the fault. The voter also compared the output of dr$_0$ to the recomputing circuit's decoded output. If the outputs were equal ($Tc_0 = 0$), the technique was able to detect a fault-free module.

A fault was undetected if there was a mismatch in the output of dr$_0$ and dr$_1$ ($Hc = 1$), and the technique could detect neither the faulty module (status $Tc_1 = 0$) nor the fault-free module (status $Tc_0 = 1$). An incremented counter shows the number of total undetected faults. Reading this counter from the prototype board, we calculated the percentage of undetected faults. The results in Table 1 show that all variations of RESO had good results in terms of fault coverage for arithmetic-based circuits.

Table 1. Fault coverage of recomputing with shift operands (RESO) techniques in SRAM-based FPGAs.

Circuit	No. of injected faults	No. of detected faults	Detected faults (percent)
8-bit multiplier	69,206,016	69,176,011	99.95
9-bit multiplier	88,473,600	88,473,600	100.00
8-bit FIR filter	1,077,248	1,077,248	100.00

Area, performance, and power dissipation

To check area, performance, and power dissipation, our first test circuit was a 16-bit multiplier with a register in the output. We compared three implementations of this circuit in the XCV300-PQ240 FPGA: no fault tolerance, TMR, and our technique (DWC-CED for permanent faults using RESO). The application was to multiply a set of input numbers for 2,000 ns, with the inputs changing every 100 ns. We evaluated each circuit's power dissipation using Xilinx's XPower tool.

Table 2 shows the results in terms of area, performance, and power dissipation for these multipliers. Using our DWC-CED method, we reduced not only the number of I/O pins but also the area. The prototype board used a Virtex part with 240 I/O pins (166 available for the user). With TMR, we were unable to synthesize the (16 × 16)-bit multiplier. However, implementing the same multiplier with our technique, we could fit it into the chip and occupy less area.

In terms of performance, the standard multiplier without fault tolerance had a maximum delay of 54 ns for the specific application, the TMR version had a delay of 56 ns, and our DWC-CED method had a delay of 62 ns, representing an 11% degradation in performance. Power dissipation was less in the DWC-CED than the TMR technique, mainly because of differences in the logic, connections, and I/Os.

The second test circuit was an 11-tap, 9-bit, digital low-pass filter, shown in Figure 8. We multiplied the original

Table 2. Results for a 16-bit multiplier with a register in the output implemented in an XCV300-PQ240 FPGA.

Fault tolerance technique	Maximum delay (ns)	No. of I/O pads	No. of four-input LUTs	No. of flip-flops	Estimated power dissipation (mW)					
					Clock	Nets	Logic	Inputs	Outputs	Total
None	54	67	495	32	7	88	186	2	29	312
TMR	56	201	1,709	96	22	305	718	7	88	1,140
DWC-CED	62	169	1,706	162	22	282	542	5	83	934

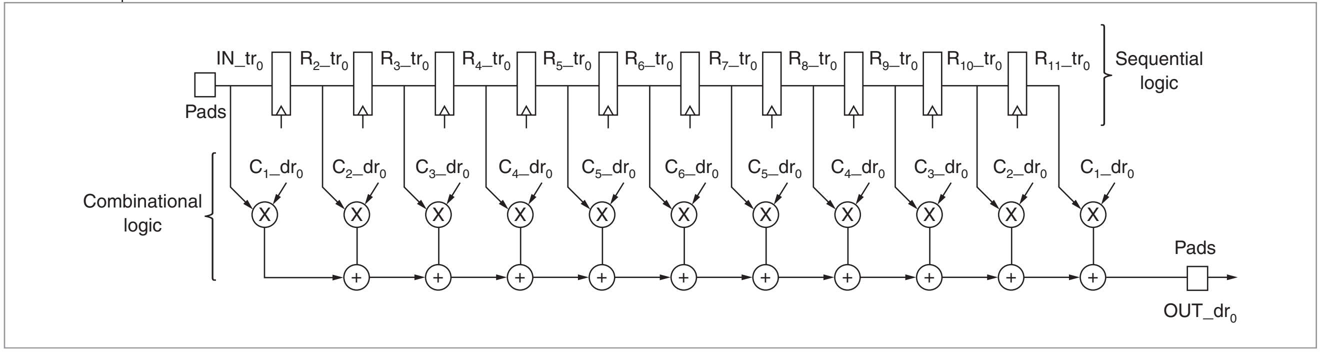


Figure 8. Digital, low-pass filter with 11 taps and 9 bits. The figure represents only one redundant block (dr_0) out of two for the combinational logic, and one redundant block (tr_0) out of three for the sequential logic. IN_tr_0 is the input to TMR combinational module tr_0; R_2_tr_0 through R_{11}_tr_0 are the registers of tr_0; C_1_dr_0 through C_6_dr_0 are constants of the filter. In these labels, dr_0 indicates that DWC protects the combinational logic such that only dr_0 and dr_1 (not shown) are necessary, and OUT_dr_0 is the output of dr_0.

Table 3. Results for a digital, 11-tap, 9-bit FIR filter implemented in the XCV300-PQ240 FPGA.

Fault tolerance technique*	Maximum delay (ns)	No. of I/O pads	No. of four-input LUTs	No. of flip-flops	Estimated power dissipation (mW)					
					Clock	Nets	Logic	Inputs	Outputs	Total
None	48	27	508	90	8	85	145	1	748	987
TMR	58	93	1,779	270	32	350	504	2	823	1,711
DWC-CED	63	75	1,738	308	25	324	530	2	19	900

* DWC-CED stores the output in registers, whereas the standard (no fault tolerance) technique and TMR do not.

coefficients calculated using Matlab (http://www.matlab.com) by a constant of 512. The final multiplier coefficients were 1, –1, –9, 6, 73, and 120.

Table 3 compares the results in terms of area, performance, and power dissipation for this digital filter implemented with no fault tolerance, TMR, and our DWC-CED technique. In this case, TMR also protected the registers, whereas the DWC-CED using RESO protected the combinational logic (multipliers and adders). The CED block resides at the outputs, where it votes on the correct pad output from dr_0 or dr_1. Results show that the FIR filter occupies a little bit less area in the FPGA when DWC-CED rather than TMR protects it. The results also show that our method uses 19% fewer pins than TMR. In terms of performance, TMR had a maximum delay of 58 ns for this test application, 20% higher than the standard (no fault tolerance) approach. Our DWC-CED technique had a maximum delay of 63 ns (8% higher than TMR) for this application.

The DWC-CED technique's power dissipation was considerably less than with TMR. But DWC-CED's power dissipation was also less than the standard (no fault tolerance) approach because our technique uses fewer input and output pins compared to TMR, uses less logic, and stores the output in a register, whereas the standard approach has the combinational logic going directly to the output pads. The DWC-CED technique also saves power because the output voter passes only one of the logic-registered outputs to the pads while the other one waits in the used one in case of a fault. TMR does not register the outputs but rather votes on them in the output pads, consuming more power.

WE'VE DISCUSSED only SEUs occurring in the SRAM programmable cells that are permanent until the next reconfiguration. However, a circuit operating in outer space can suffer from a total ionization dose and other effects that can provoke permanent physical damages in the circuit. We hope to explore these areas in the future. ■

■ References

1. A.H. Johnston, "Scaling and Technology Issues for Soft Error Rates," *Proc. 4th Ann. Research Conf. Reliability*,

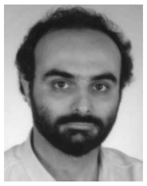
Stanford Univ., 2000; http://parts.jpl.nasa.gov/docs/Scal-00.pdf.

2. "Virtex 2.5 V Field Programmable Gate Arrays," DS003, v2.5, Product Specification, 2 Apr. 2001, Xilinx; http://direct.xilinx.com/bvdocs/publications/ds003.pdf.
3. J. Barth, C. Dyer, and E. Stassinopoulos, "Space, Atmospheric, and Terrestrial Radiation Environments," *IEEE Trans. Nuclear Science*, vol. 50, no. 3, June 2003, pp. 466-482.
4. E. Normand, "Single Event Upset at Ground Level," *IEEE Trans. Nuclear Science*, vol. 43, no. 6, Dec. 1996, pp. 2742-2750.
5. D. Alexandrescu, L. Anghel, and M. Nicolaidis, "New Methods for Evaluating the Impact of Single Event Transients in VDSM ICs," *Proc. IEEE Int'l Symp. Defect and Fault Tolerance in VLSI Systems*, IEEE CS Press, 2002, pp. 99-107.
6. M. Rebaudengo, M.S. Reorda, and M. Violante, "Simulation-Based Analysis of SEU Effects of SRAM-Based FPGAs," *Proc. Int'l Workshop Field-Programmable Logic and Applications*, IEEE CS Press, 2002, pp. 607-615.
7. E. Fuller, M. Caffrey, and P. Blain, "Radiation Test Results of the Virtex FPGA and ZBT SRAM for Space Based Reconfigurable Computing," *Proc. Int'l Conf. Military and Aerospace Applications of Programmable Logic Devices* (MAPLD 02), NASA Office of Logic Design, 2002, pp. 1-8.
8. M. Ohlsson et al., "Neutron Single Event Upsets in SRAM-Based FPGAs," *Proc. IEEE Nuclear Space Radiation Effects Conf.* (NSREC 98), IEEE Press, 1998, pp. 1-4; http://www.xilinx.com/appnotes/FPGA_NSREC98.pdf.
9. C. Carmichael, "Triple Module Redundancy Design Techniques for Virtex Series FPGA," Xilinx Application Notes 197, v1.0, Mar. 2001, p. 137; http://www.xilinx.com/bvdocs/appnotes/xapp197.pdf.
10. F. Lima et al., "A Fault Injection Analysis of Virtex FPGA TMR Design Methodology," *Proc. European Conf. Radiation and Its Effects on Components and Systems* (RADECS 01), IEEE CS Press, 2001, pp. 275-282.
11. M. Lubaszewski and B. Courtois, "A Reliable Fail-Safe System," *IEEE Trans. Computers*, vol. 47, no. 2, Feb. 1998, pp. 236-241.
12. J. Patel and L. Fung, "Multiplier and Divider Arrays with Concurrent Error Detection," *Proc. Int'l Symp. Fault-Tolerant Computing*, IEEE CS Press, 1982, pp. 325-329.

Fernanda Gusmão de Lima Kastensmidt is a professor in the Department of Digital Systems Engineering at the State University of Rio Grande do Sul in Guaíba, Brazil, and also an associate professor at the Institute of Informatics of the Federal University of Rio Grande do Sul in Porto Alegre, Brazil. Her research interests include VLSI testing and design, fault effects, fault-tolerant techniques, and programmable architectures. Kastensmidt has a BS in electrical engineering, and an MS and a PhD in computer science and microelectronics, all from the Federal University of Rio Grande do Sul. She is a member of the IEEE.

Gustavo Neuberger is a PhD student at the Institute of Informatics of the Federal University of Rio Grande do Sul. His research interests include fault tolerance, radiation effects, DFT, and SRAM memories. Neuberger has a BS in computer engineering from the Federal University of Rio Grande do Sul. He is a member of the ACM.

Renato Fernandes Hentschke is a PhD student at the Institute of Informatics of the Federal University of Rio Grande do Sul. His research interests include design automation for physical design; and algorithms for placement, routing, and congestion estimation. Hentschke has an MS and a BS in computer science from the Federal University of Rio Grande do Sul. He is a member of the ACM.



Luigi Carro is a professor in the Electrical Engineering Department and the graduate program at the Institute of Informatics of the Federal University of Rio Grande do Sul. His research interests include mixed-signal design, DSP, mixed-signal and analog testing, and fast system prototyping. Carro has a BSc in electrical engineering, an MSc in computer science, and a PhD in computer science, all from the Federal University of Rio Grande do Sul. He is a member of the IEEE and the ACM.

Ricardo Reis is a professor at the Institute of Informatics of the Federal University of Rio Grande do Sul, and the Latin America liaison for *IEEE Design & Test*. His research interests include VLSI design, CAD, physical design, design methodologies, and fault-tolerant techniques. Reis has a BSc in electrical engineering from the Federal University of Rio Grande do Sul, and a PhD in computer science and microelectronics from the Institut National Polytechnique de Grenoble, France. He is a vice president of the International Federation for Information Processing and a member of the IEEE.

■ Direct questions and comments about this article to Fernanda Gusmão de Lima Kastensmidt, PO Box 15064, Porto Alegre – RS – Brasil, 91501-970; fglima@inf.ufrgs.br.

For more information on this or any other computing topic, visit our Digital Library at http://www.computer.org/publications/dlib.