Reduced-Precision Redundancy for Reliable FPGA Communications Systems in High-Radiation Environments

BRIAN PRATT
L-3 Communications, Communications Systems-West

MEGAN FULLER
MICHAEL RICE, Senior Member, IEEE

MICHAEL WIRTHLIN, Senior Member, IEEE
Brigham Young University

Triple modular redundancy (TMR) is the traditional mitigation technique for field-programmable gate arrays (FPGAs) subject to single-event upsets (SEUs) in high-radiation environments. Reduced-precision redundancy (RPR) as an alternative to TMR for communications systems is demonstrated. RPR reduces the number of "catastrophic" SEUs in several simple communications receivers by over 95%, thus increasing the mean time to failure (MTTF) by 25 to 99 times, while consuming less than half of the resources that TMR does in most cases.

Manuscript received June 15, 2011; revised February 16, 2012; released for publication April 27, 2012.

IEEE Log No. T-AES/49/1/944359.

Refereeing of this contribution was handled by P. Willett.

This work was supported by the I/UCRC Program of the National Science Foundation under Grant 0801876.

Authors' current addresses: B. Pratt, L-3 Communications, Communication Systems-West, 640 North 2200 West, Salt Lake City, UT 84116; M. Fuller, Department of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139; M. Rice and M. Wirthlin, NSF Center for High-Performance Reconfigurable Computing (CHREC), Department of Electrical and Computer Engineering, Brigham Young University, 459 Clyde Bldg., Provo, UT 84602, E-mail: (mdr@byu.edu).

I. INTRODUCTION

Field-programmable gate arrays (FPGAs) are an attractive target platform for reconfigurable radios [1]. FPGAs have been used to implement communication-specific processors for well over a decade. Their ability to combine flexibility with good performance makes FPGAs popular for software-defined radios. Reconfigurable radios are also becoming more attractive for space-based applications. The ability to reconfigure the FPGA resources with an updated radio personality reduces the amount of hardware needed on the spacecraft [2].

The problem with using FPGAs in space is the presence of high-energy particles [3, 4]. These particles can alter the state of static memory cells in integrated circuits. Upsets in the FPGA configuration memory may alter the operation of the digital circuit defined by the memory state. Because most of the FPGA area is devoted to static memory cells, an FPGA is more sensitive to radiation than an application-specific integrated circuit (ASIC).

To operate reliably in space, a hardware mitigation strategy, such as triple modular redundancy (TMR), must be applied. TMR, however, is very expensive and requires three times more hardware resources than an unmitigated circuit. Motivated by the observation that an FPGA-based radio comprises mostly arithmetic operations, this paper explores the application of reduced-precision redundancy (RPR) to the problem. The metric used to evaluate the effectiveness of RPR is the bit error rate (BER) achieved by the FPGA-based radio. To fully evaluate the benefits of RPR on a communications system, the impact of ionizing radiation on BER must be well understood.

To this end Section II describes the results of a novel single-event upset (SEU) fault injection strategy that allows us to more fully characterize the impact of SEUs on the BER performance. Section III describes RPR in more detail. A detailed accounting of previous work is summarized, and it is in this context that a detailed description of our contributions to RPR make the most sense. Sections IV, V, and VI describe the application of the fault injection strategy to a binary pulse amplitude modulation (PAM) detector with perfect timing synchronization (Section V) and with a phase-locked loop (PLL)-based timing synchronizer (Section VI).

This work presents several contributions over previous work. First, RPR is applied to a real FPGA-based system and is validated with extensive fault injection experiments. Second, a system-level metric, the BER, is used to evaluate the effectiveness of RPR in the presence of radiation. Third, critical clock and reset signals in the FPGA are included in the mitigation approach to improve the BER. The results from this fault injection demonstrate that RPR

increases the mean time to failure (MTTF) of this radio by more than 20 times and at a much lower cost than traditional approaches such as TMR.

II. FPGA RADIOS IN SPACE

FPGA-based radios must deal with the effects of radiation when used in space systems. This section describes the environment that faces FPGAs in space and the main techniques that have been used in the past to combat the effects of radiation. It also describes the specific effects of radiation on FPGA-based digital radios and how mitigation strategies might be optimized for these types of systems.

A. Single Event Upsets

SRAM-based (static random access memory) FPGAs consist of a large array of memory cells. These memory cells hold both user data and configuration data that define the operation of the circuit. Charged particles affect these cells by occasionally inverting the contents of a particular cell. Such an event is called an SEU [5]. A corrupted memory cell may alter either the user data or the FPGA configuration [6]. For an SRAM-based FPGA, most of the memory cells are found within the configuration memory. The FPGA configuration memory defines the operation of the routing, logic, I/O, and other internal functions of an FPGA design. Upsets within the configuration memory are most problematic because these upsets alter the function of the FPGA circuit. For example, an upset within the configuration memory may change the routing or the logic of the FPGA and cause the FPGA design to operate incorrectly.

To protect an FPGA design from SEUs, several fault tolerance techniques are typically used. First, the upsets themselves are periodically repaired to prevent upset accumulation. This is accomplished using a technique known as configuration scrubbing. Configuration scrubbing is a periodic check and correction of the configuration memory of the FPGA. An external radiation-hardened memory can be used to store a master copy of the FPGA configuration to aid in this correction. Accompanying configuration scrubbing is some form of hardware redundancy. Hardware redundancy techniques involve the use of additional, redundant hardware to mask the effects of SEUs that occur in the FPGA configuration memory. If the scrubbing rate is sufficiently high relative to the SEU rate, the accompanying redundancy technique only needs to mask one SEU at a time [7].

The most popular redundancy technique for mitigating configuration SEUs on FPGAs is TMR [8]. TMR uses three copies of the circuit and voting to choose the correct output. Figure 1 shows a simplified block diagram of a digital filter protected with TMR. As long as two of the three modules are operating correctly, the final output is correct. TMR is popular because it is straightforward to implement and provides very effective protection for any type of design.

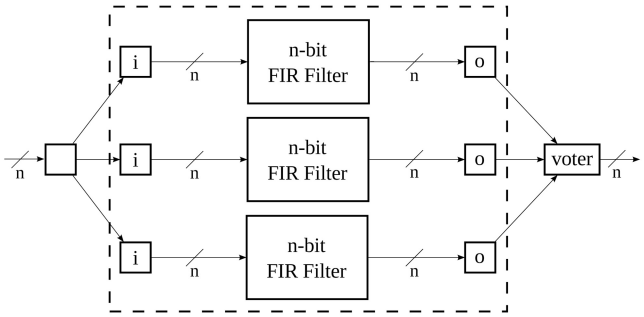

Fig. 1. Simplified block diagram of n-bit FIR filter protected with TMR. Portion surrounded by dotted box is implemented on FPGA.

Although TMR is very effective at protecting FPGA designs from SEUs, TMR requires three times the hardware resources and consumes roughly three times the power. In light of the high cost of TMR, more efficient alternatives to TMR have been explored [9–12]. These techniques exploit application-specific properties of the design to protect the design with fewer resources than TMR. These techniques also are less costly because they do not fully protect the design against SEUs. These techniques provide a trade-off between mitigation effectiveness and FPGA resource requirements. Unfortunately, these techniques are only effective for specific circuit types, such as state machines. There is a need for more efficient SEU mitigation strategies for general arithmetic circuits and for digital signal processing systems. A detailed investigation of the relationship between SEU and BER performance for an FPGA-based digital radio is described in the next section. The results of this investigation show that more efficient SEU mitigation is possible with a technique known as RPR.

B. The Impact of SEUs on BER Performance

To learn more about the effects of SEUs on a reconfigurable radio, a novel fault injection experiment was created. SEUs within a reconfigurable radio were emulated by intentionally modifying individual configuration bits within the configuration memory [13]. As expected many configuration upsets caused the FPGA circuit to deviate from its proper operation. The effect of these upsets on the radio, however, varied significantly. In some cases, a given configuration upset had a significant negative effect on the radio. Most configuration upsets, however, had a limited effect on the operation of the radio.

The figure of merit used to evaluate the behavior of the impact of SEUs radio is the BER. Some of the configuration upsets caused little or no change

in the BER, while others had a considerable impact on the BER performance. Representative examples of the BER curves generated from these experiments are shown in Fig. 2.

Four general classes of SEUs are identified from these experiments. The SEU classes were defined as follows.

1) A Class 1 SEU causes almost no perturbation in the BER performance of the receiver. The measured loss is less than 0.1 dB as compared with the theoretical BER curve. The SEUs in this class are those that impact the low-order bits in the arithmetic computations (either the value of the filter coefficient or the multiply-accumulate operation). Depending on the matched filter implementation details, class 1 SEUs are 30%–77% of the total number of SEUs.

2) A Class 2 SEU degrades the BER performance in the same way that a source of additive noise degrades performance. This effect can be thought of as either an implementation loss or, curiously, as a noise figure. Class 2 SEUs are those that impact the memory cells that define the middle-order bits of the filter coefficients and the middle-order bits of the outputs of the arithmetic units. Depending on the matched filter implementation details, class 2 SEUs are 17%–64% of the total number of SEUs.

3) A Class 3 SEU produces an unusably-high BER floor.[1] SEUs impacting the memory cells that define the high-order bits of computation, as in the filter coefficients and the outputs of the arithmetic units, are the main causes of SEUs in this category. These SEUs are considered "catastrophic." Depending on the matched filter implementation details, class 3 SEUs are 3%–4% of the total number of SEUs.

4) A Class 4 SEU produces a BER of 1/2. These SEUs are also catastrophic and are caused by faults in the memory cells that define the clock distribution network, the global reset signal, the most significant bit (MSB) of matched filter output, and the threshold comparator in the decision block. Depending on the matched filter implementation details, class 4 SEUs are 2%–4% of the total number of SEUs.

The main observation from these results is that the not all SEUs impact the radio in the same way. For example, class 1 and class 2 SEUs are not critical. The errors induced by these SEUs are easily correctable by standard techniques, such as increased link margin or error-control coding. The class 3 and class 4 SEUs are critical, and redundancy must be used to improve the performance of these systems. This observation influences how the redundancy techniques are applied to a reconfigurable radio. Rather than applying TMR on a reconfigurable radio, a lower cost, selective mitigation approach can be used that protects against the class 3 and class 4 SEUs.

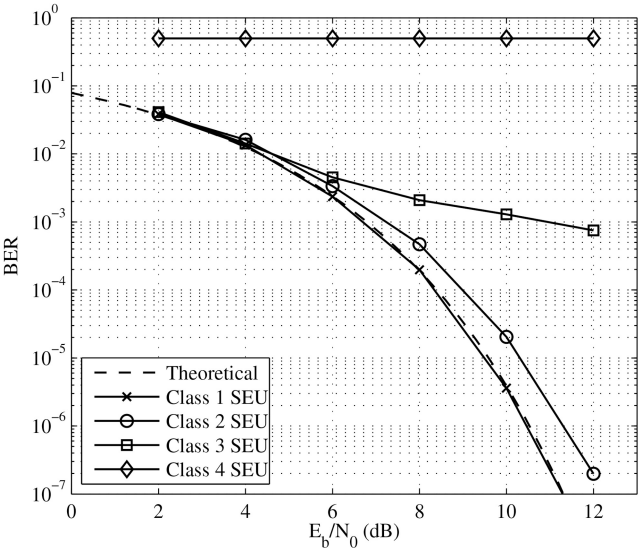

Fig. 2. Representative examples of bit-error-rate versus E_b/N_0 curves from each of the four SEU classes described in Section II-B. For reference, the theoretical bit error rate curve for the AWGN environment is also included. The system was binary PAM using a square-root raised cosine (SRRC) pulse shape with 100% excess bandwidth. The matched filter was an FIR filter using combinational logic.

C. SEU Mitigation for a Reconfigurable Radio

The results from these experiments suggest that a more efficient SEU mitigation is possible for reconfigurable radios by providing selective SEU mitigation. The goal of such a mitigation approach is to protect only those configuration memory cells corresponding to class 3 and class 4 SEUs. The configuration memory cells associated with class 1 and class 2 SEUs can be ignored. Because the class 3 and class 4 SEUs account for only 7% of the FPGA circuit defining the FPGA radio, the potential resource savings of such a mitigation approach are substantial.

Using these assumptions we identified an existing redundancy technique that could be tailored to FPGA-based communications systems. This technique, known as RPR, corrects errors in the most significant bits of computation while ignoring errors in the lower order bits [14]. This is similar to the idea of applying TMR only to these more critical bits. By adding the protection of the global clock and reset signals, RPR could fill the role of protecting these FPGA-based systems with less hardware overhead than TMR.

In this paper we show that RPR is effective at significantly reducing the number of catastrophic SEUs in several simple communications systems.

[1]Note that our simulations ran only long enough to estimate BERs greater than 10^{-6} with any useful reliability. It could be the case that many of the class 2 SEUs really do have a BER floor somewhere below 10^{-6}. A case could be made that these class 2 SEUs should be class 3 SEUs. Given the fact that most modern digital communication systems use some form of error control coding and that any useful error correcting code can easily correct random errors at the rate of 10^{-6} or less, there is little merit in determining if such low BER floors exist.

This paper demonstrates the effectiveness of RPR by using two case studies. The first shows that RPR is effective for several variations of a matched filter detector, using only feed-forward logic, in a binary PAM detector. The second uses RPR to reduce the cost of mitigation on a more complex receiver with a symbol timing synchronization loop, which demonstrates RPR's effectiveness on a system with recursive processing.

The results presented in this paper show a significant reduction in the number of catastrophic SEUs (by over 95%) by using this technique at a cost of one-quarter that of TMR in some cases, and about equal that of TMR in others. This reduction corresponds to a 20–40-fold increase in the MTTF. Thus, we show that it is possible to use RPR to significantly increase the reliability of a communications system at a much lower cost than the traditional TMR approach. The comprehensive experimental results presented for these test systems suggest that RPR could be beneficial in other systems as well.

III. REDUCED-PRECISION REDUNDANCY

RPR is a redundancy technique similar to TMR that requires less hardware overhead by using reduced-precision (RP) arithmetic in two of its three replicas. It takes advantage of the fact that RP arithmetic can be a good estimate of computations that use higher precision. While TMR protects the entire circuit and provides an error-free output, RPR simply limits the error at the output of a module. RPR has an advantage over TMR when it is able to sufficiently limit the magnitude of the SEU-induced noise at a lower hardware cost.

RPR is not suited to protect any type of circuitry as TMR is. Operations that can be approximated with less hardware than the standard module are candidates for RPR. RPR has, generally, been used to protect arithmetic operations. In addition the approximation and the decision hardware required to choose the final output must not exceed the cost of TMR, otherwise, any advantage of RPR is lost.

A. Related Work

RPR has previously been shown to be effective in correcting errors in other types of circuits under various error conditions. Shim, et al. introduced RPR as part of a power-reduction technique for ASIC-based digital signal processing (DSP) systems [14, 15]. In those papers, RPR was used to correct errors induced by the voltage over-scaling (VOS) power-reduction technique. Later, Shim and Shanbhag expand the technique to consider soft errors in ASIC systems in [16]. A recent paper by Reviriego, et al. used a modified version of Shim's RPR to protect an ASIC-based adaptive filter from the effects of soft errors [17]. By taking advantage of the self-correcting nature of the adaptive filter, they could reduce the cost of RPR for this specific type of system.

The effects of soft errors in FPGA-based systems are different from those described in [14]–[17]. In an ASIC soft errors affect only the data passing through the system. In an FPGA such errors affect not only the data passing through the system, but also the configuration of the system itself. Consequently, the effects of SEUs on FPGAs are worse than on ASICs.

Snodgrass [18] studied a different variation of RPR for FPGA systems and demonstrated it on a CORDIC module (coordinate rotation digital computer). These results showed that RPR could be applied to an FPGA-based system to limit high-magnitude errors in a radiation environment, though the critical clock and reset signals were not included in the protection scheme. Sullivan [19] presented MATLAB simulations of this alternate variation of RPR (showing that RPR has promise for correcting SEUs in general arithmetic operations) and reported on some of the estimated hardware costs of RPR.

Although previous work has demonstrated that RPR is able to reduce the negative impact of SEUs on numerical computations, none of this work has investigated the benefits of RPR on the performance of a communication system. This paper measures the effects of soft errors on the BER of a communications receiver. RPR is applied to two different styles of communication receivers, and the BER of the receivers protected by RPR are compared against a receiver without RPR. In addition to RPR cricital clock and reset signals are included in the mitigation approach to maximize the benefits of RPR. Extensive fault injection experiments are performed on every sensitive configuration memory cell in the target FPGA design. The results from these experiments provide concrete evidence that RPR is an effective way of protecting communication circuits from the effects of SEUs.

B. Implementation

The RPR method used in this paper is that introduced by Shim and Shanbhag in [16]. Figure 3 shows a block diagram of an n-bit finite impulse response (FIR) filter (a filter with n-bit registers and coefficients) protected with RPR. The figure shows that the inputs to the filter are triplicated, as with TMR, and that the second and third replicas of the circuit are implemented with RP (k-bit, where $k < n$) FIR filters. Note that the decision blocks and outputs can be triplicated as well to avoid single points of failure in those modules. The outputs of the three identical decision blocks are voted on, as in the TMR system.

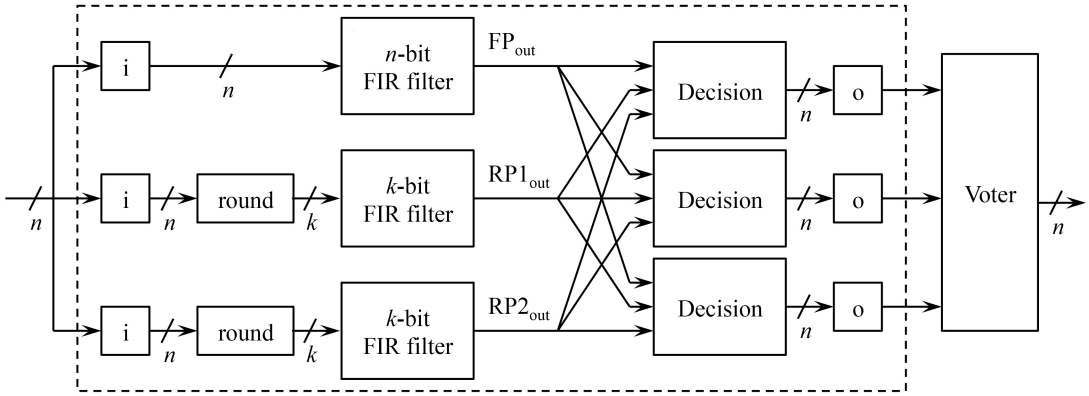

Fig. 3. Block diagram of an n-bit FIR filter protected with RPR using k-bit RP filters ($k < n$). The portion surrounded by the dotted box is implemented on the FPGA.

To determine the presence of an error, the decision block compares the outputs of the full-precision (FP) filter ($\mathrm{FP_{out}}$) with the outputs of the two RP filters ($\mathrm{RP1_{out}}$ and $\mathrm{RP2_{out}}$) as follows:

if (($|\mathrm{FP_{out}} - \mathrm{RP1_{out}}| > T_h$) AND ($\mathrm{RP1_{out}} = \mathrm{RP2_{out}}$))
  output $\Leftarrow \mathrm{RP2_{out}}$
else
  output $\Leftarrow \mathrm{FP_{out}}$.

In other words the FP output is used when no error is found or when the two RP modules disagree. Otherwise, the RP output is used, which provides an estimate of the correct FP output.

RPR, in the form presented here, has two main parameters that can be adjusted. The impact of these parameter settings can be understood in terms of the arithmetic error

$$\epsilon = |\mathrm{FP_{true}} - \mathrm{RP_{true}}| \tag{1}$$

where $\mathrm{FP_{true}}$ and $\mathrm{RP_{true}}$ are the outputs of the FP and RP filters, respectively, when no SEU is present. First, the size of the RP modules can be modified. In this paper the size of the RP filters is measured by the bit-width of the filter input signal, k. A larger RP filter gives a better estimate of the FP filter. This results in a better detection of errors in the FP filter and a lower ϵ. A smaller RP filter is desirable because a smaller RP filter reduces the cost of mitigation.

The second parameter for RPR is the threshold value T_h. A threshold that is too small will cause the RP output to be chosen even when there are no errors in the FP module. To prevent this, $T_h \geq \max\{\epsilon\}$ is required. On the other hand, if T_h is too large, the FP output is used even when there are significant errors in that module. In fact any error that is larger than $\max\{\epsilon\}$ must be due to an upset in the system. Consequently, T_h should be no greater than this value. In light of these bounds, T_h is set equal to $\max\{\epsilon\}$ for the experiments described in Section IV.

In addition to protecting the most significant bits of computation, Shim's RPR can be extended to protect the critical clock and reset signals in an FPGA system. In the implementation presented here, the global clock and reset inputs to the system are triplicated in addition to the data input bits.[2] A different set of global signals is fed to each of the three circuit replicas. Thus, when a single global signal is upset, only one of the three replicas is disabled. The RPR decision logic naturally works around this error. The RP modules are bypassed if one of them is disabled, and the RP output is used whenever the output of a disabled FP module falls outside the error-limiting threshold value.

IV. TEST METHODOLOGY

To demonstrate the RPR technique for communications applications, we have applied RPR to two types of communications receivers. To determine the effectiveness of RPR, we measured the effect on the BER of every possible configuration SEU in each of these systems. To measure the efficiency of RPR, we used TMR to protect the same circuits and compared the circuit area overhead. By using RPR we expected to eliminate or significantly reduce catastrophic SEUs as compared with the original designs. We also expected to see a significantly lower implementation cost than TMR.

Case study I examines a simple feed-forward binary PAM matched filter detector. This case study evaluates the RPR technique for three different receiver configurations, varying the matched filter architecture and the coefficient values. The experiments presented show that the RPR's effectiveness is not strongly dependent on either of these factors, while it can be a cost-effective alternative to TMR.

Case study II evaluates RPR on a more complex binary PAM receiver with a symbol timing synchronization PLL. This type of test is significant

[2]Some amount of skew between the three clock signals is introduced, which can often be managed by adding constraints to the circuit synthesis software, as is done with similar TMR systems.

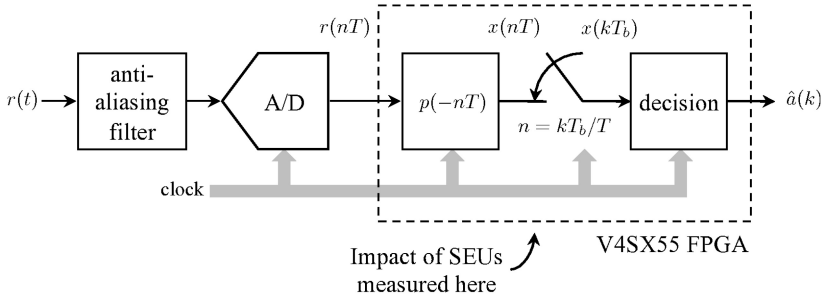

Fig. 4. High-level block diagram of system.

because recursive (or feedback) systems often have more complex error dynamics than feed-forward systems. Recursive systems with redundancy must also synchronize the redundant modules after an SEU is corrected through scrubbing. This is handled in TMR by inserting voters in feedback loops [20]. RPR decision blocks should be used in the same manner. The results of this case study show that, despite these issues, RPR can be an effective alternative to TMR for recursive systems as well.

A. Experimental Configuration

We characterized the BER performance of each system in the presence of SEUs by using fault injection [6]. The fault injection experiments were conducted as follows.

1) The design under test was targeted to a Xilinx Virtex-4 SX55 FPGA. The design is defined by a file called a "configuration bit file."
2) The bits in the configuration bit file define the contents of all the configuration memory cells in the entire FPGA. The bits defining the memory cell contents of the design under test were identified [6].
3) One of the bits in the set defined in step 2 was inverted in the original, clean configuration bit file, and the FPGA was configured using this corrupt file.
4) For this SEU a BER curve was generated using the corrupted configuration bit file.
5) For the noncatastrophic SEUs the BER curve produced by the previous step was compared with the curve for the system in the absence of upsets to estimate the performance loss at a BER of 10^{-5}.

Steps 3–5 were repeated for each of the configuration bits in the set defined in step 2. This simulated the occurrence of all possible SEUs, each being present one at a time as expected in an FPGA system with a proper scrubbing system.

An analysis circuit built within the FPGA enabled the processing of tens of millions of samples of data per upset configuration cell. With this capability the BER curve for every bit in the circuit could be constructed down to BERs of 10^{-6}. This level of detail has not been reached in any previous experiments and provides a very meaningful view of the effect of SEUs on the communications systems tested.

V. CASE STUDY I: MITIGATION OF FEED-FORWARD SYSTEM

The feed-forward system tested is illustrated in Fig. 4. A binary PAM matched filter detector is implemented in the FPGA. The anti-aliasing filter and digitizer (ADC) are outside the FPGA and are assumed to operate normally in the high-radiation environment. The downsample and decision blocks were ignored in these experiments to simplify the data collection and analysis. The filter makes up the bulk of the design in terms of configuration bits.

The following two filter architectures were considered.

1) A direct form 1 FIR filter, as shown in Fig. 5(a), was constructed directly from FPGA slices (generic logic).
2) An alternative approach, based on the built-in DSP blocks (called "dsp48" blocks), was used to design a transposed direct form 1 FIR filter, as illustrated in Fig. 5(b).

The matched filter was designed with 25 taps with symmetric coefficients and, thus, 13 multipliers. The filter pulse shape was the square-root raised-cosine (SRRC) pulse shape with excess bandwidth α using $L_p = 3$ [21]. The unmitigated filter used 16-bit registers and coefficients ($n = 16$) and operated at $N = 4$ samples/bit.

Three versions of this design were tested in order to determine any impact of these variations on the utility of RPR. These designs are labeled as follows.

1) "logic $\alpha = 1.0$" means the SRRC pulse shape with $\alpha = 1.0$ in the arrangement illustrated in Fig. 5(a).
2) "logic $\alpha = 0.25$" means the SRRC pulse shape with $\alpha = 0.25$ in the arrangement illustrated in Fig. 5(a).
3) "dsp48 $\alpha = 1.0$" means the SRRC pulse shape with $\alpha = 1.0$ in the arrangement illustrated in Fig. 5(b).

A. Mitigation Details

In the FPGAs used in our experiments, there are two choices for implementing the RP filters: the embedded DSP blocks and combinational logic.

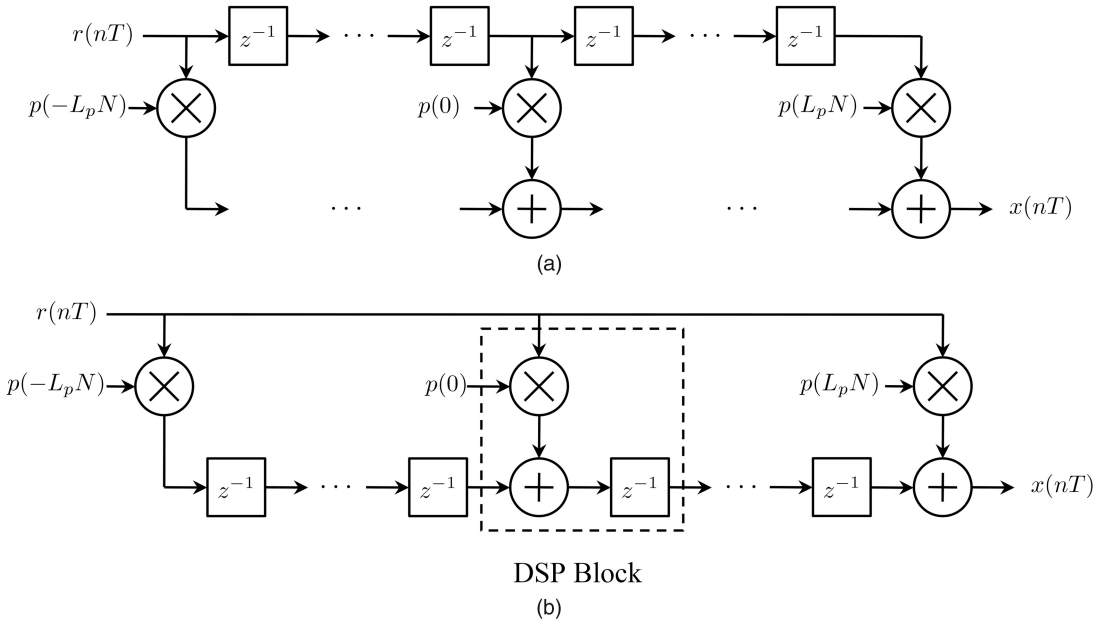

Fig. 5. FIR filter structures examined in fault injection experiments. (a) Direct form 1 FIR filter. (b) Transposed direct form 1 FIR filter.

Because the embedded DSP blocks operate with a fixed precision (18 bits), an RP filter based on the embedded DSP blocks consumes the same resources as a full-precision filter. Consequently, RPR and TMR require equivalent FPGA resources. On the other hand the resources required to implement filters using combinational logic decrease as the bit width k decreases. Using combinational logic to implement the RP filter allows the experiments to simultaneously capture impacts of arithmetic precision, SEU detection, and FPGA area. For this reason the RP filters used in our RPR experiments were implemented using combinational logic.

As discussed in Section III-B, the value of k affects the area overhead of RPR as well as the amount of protection offered. As k is increased the cost of mitigation increases and approaches that of TMR. If k is too small, however, the RP filter does not adequately represent the original filter and is not an effective estimator of its output. In the authors' experience $k = 8$ is a good compromise between these two factors. For any given system or module to be protected, there could be a range of k values that could be useful, with the trade-offs discussed previously.

A value of $T_h = 0.5E_b$ (E_b = energy per bit) was chosen as the threshold to compare between the RP and FP outputs. For these filter sizes this threshold ensured that an error would never be declared when no SEUs were present in the system.

In addition to the RPR implementations, the experiments report on the results of the TMR mitigation as well. The application of TMR was done by the BL-TMR tool [22], which is similar to the Xilinx TMRTool [23]. Both tools have been shown to be effective in protecting FPGA designs from the effects of SEUs. The majority voting circuitry was performed off-chip, as with the RPR implementations.

B. Results

Table I shows the results from the fault injection experiments. The table illustrates the differences between the RPR and TMR implementations of each filter design and their improvements over the original unmitigated filter. For each design the number of SEUs in each class are tabulated along with the implementation overhead, which is measured by the number of configuration bits. The main point here is the factor by which the MTTF—measured by the mean time to the occurrence of an unmitigated catastrophic SEU—increases. These results show that both TMR and RPR increase MTTF. Theoretically, TMR eliminates all SEUs and offers an infinite increase in MTTF. However, some catastrophic SEUs remain in this TMR implementation resulting in a finite MTTF. RPR reduced the number of catastrophic SEUs by over 95% in each case, increasing the MTTF of each by over 20×. This was true for each combination of filter architecture and coefficient values.

In addition to the catastrophic SEUs, TMR eliminated all class 2 SEUs, while the number class 2 SEUs increased for RPR in two of the three cases. In the case of RPR, an increase in class 2 SEUs is reasonable due to the added logic in the system. Many more resources are added, not all of which are protected fully as in TMR. As discussed earlier the class 1 and 2 SEUs are not as critical as the catastrophic SEUs.

TABLE I
Fault Injection Results for Three FIR Filter Designs Protected with RPR and TMR, Compared Against the Original (unmitigated) Filters

Design	Class 1 SEUs	Class 2 SEUs	Class 3 SEUs	Class 4 SEUs	Total Bits	Implementation Overhead (bits)	Total Catastrophic (% Reduction)	Increase in MTTF
logic $\alpha = 1.0$	33,287	7,154	1,638	899	42,978	—	2,537 (—)	—
TMR logic $\alpha = 1.0$	129,387	0	0	2	129,389	86,411 (201%)	2 (99.9%)	1,268.5×
RPR logic $\alpha = 1.0$	54,310	9,409	96	2	63,817	20,839 (48.5%)	98 (96.1%)	25×
logic $\alpha = 0.25$	21,072	44,205	2,908	1,022	69,207	—	3,930 (—)	—
TMR logic $\alpha = 0.25$	212,102	0	0	2	212,104	142,897 (206%)	2 (99.9%)	1,965×
RPR logic $\alpha = 0.25$	86,309	23,828	192	2	110,331	41,124 (59.4%)	194 (95.1%)	20.26×
dsp48 $\alpha = 1.0$	20,514	7,031	867	1,118	29,530	—	1,985 (—)	—
TMR dsp48 $\alpha = 1.0$	62,483	0	0	2	62,485	32,955 (112%)	2 (99.9%)	992.5×
RPR dsp48 $\alpha = 1.0$	52,517	9,911	9	11	62,448	32,918 (111%)	20 (98.99%)	99.25×

For both TMR and RPR, the total number of SEUs increased as well as the number of class 1 SEUs. This is expected due to the area overhead required by both techniques. The area overhead required by RPR, however, was significantly less than TMR for the logic-based filters at about one-quarter the cost of TMR.

Interestingly, the implementation cost of TMR was relatively lower for the filter based on the embedded DSP blocks. Due to the architecture characteristics of the DSP blocks, fewer than 3 times the number of configuration bits are needed to fully triplicate this design. The RPR implementation also caused a more significant increase in area than the logic-based designs, most likely due to the use of logic-based filters for the RP modules rather than the fixed-width DSP blocks. In the case of this DSP block-based filter, then, TMR is preferable to RPR. If resource constraints limit the availability of DSP blocks, however, or possibly if a different set of filter bit-widths is selected, some form of RPR may be appropriate for this type of filter.

VI. CASE STUDY II: MITIGATION OF RECURSIVE SYSTEM

The recursive receiver system is pictured in Fig. 6(a). This is a binary PAM receiver with a symbol timing synchronization PLL. In this experiment the matched filter pulse shape was the SRRC pulse shape with excess bandwidth $\alpha = 0.5$ using $L_p = 3$. The 25-tap matched filter (again with 13 multipliers) operated at $N = 4$ samples/bit. The unmitigated filter used 16-bit registers and coefficients.

The timing recovery loop operates at a rate of 2 samples/bit. The interpolator is a piecewise parabolic Farrow interpolator [21]. The timing error detector (TED) block is a zero-crossing TED. The loop filter is a first-order filter—a single constant multiplier. The numerically-controlled oscillator (NCO) generates the timing synchronization pulses and provides the fractional interpolation interval back to the interpolator.

A. Mitigation Details

Because RPR is not suitable for all types of modules, by focusing mainly on arithmetic operations, a mixture of RPR and TMR is sometimes the best approach for mitigating SEU-induced noise. Figure 6(b) shows a diagram of the recursive receiver system annotated with the type of mitigation applied to each component in the system. The locations of a TMR voter (for synchronization) and an RPR voter are also indicated. The "decision" module is not an arithmetic module and cannot be protected by RPR. The "NCO" block contains small feedback loops (not pictured), where a TMR voter or RPR decision block must be inserted. Due to the high cost of RPR decision blocks, however, it is actually more efficient to apply TMR to this and to the "TED" and "loop filter" blocks than to use RPR in this section.

The "matched filter" and "interpolator" modules, however, are ideal candidates for RPR. Each contains a significant amount of arithmetic logic. In fact these blocks make up the bulk of the design in terms of FPGA resources. The combined size of these modules offsets the cost of adding an RPR decision block.

For the matched filter and interpolator, then, RP modules with 7 bits of precision at their inputs were added. We determined that this redundancy factor ($k = 7$) would be a suitable trade-off between mitigation cost and SEU protection for this system. The value of T_h was again set to be higher than the maximum difference between the FP and RP modules.

The "RPR voter" at the output of the interpolator used three identical decision blocks and converted the three interpolator output signals (one FP and two RP) into three identical, FP outputs. The three identical outputs were needed by the triplicated TED and decision blocks, which were protected with TMR. The TMR voter at the output of the loop filter block intersects the two feedback loops pictured, thus correcting any synchronization issues between the three branches after upsets occur and are repaired.

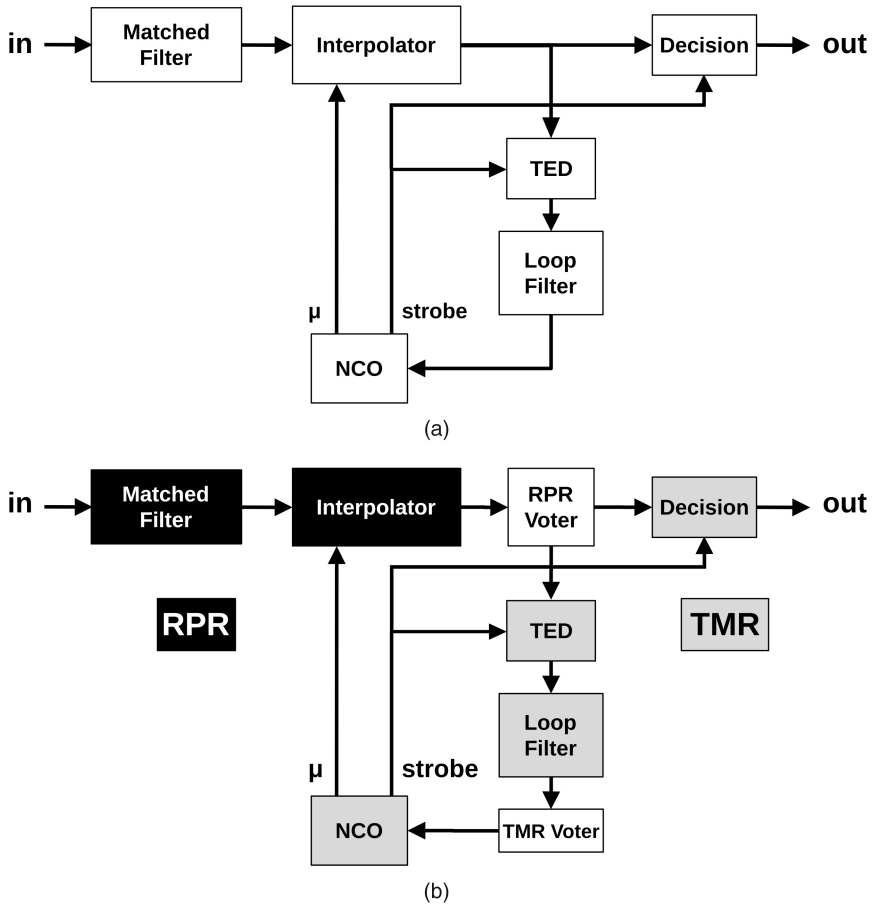

Fig. 6. Block diagram of binary PAM demodulator with timing synchronization. (a) Unmitigated. (b) Annotated for RPR+TMR mitigation.

TABLE II
Number of SEUs Causing Each Class of Effect for the Binary PAM Demodulator Protected with Full TMR and RPR+TMR, Compared Against the Original (unmitigated) Demodulator

Design	Class 1	Class 2	Class 3	Class 4	Total	Implementation Overhead	Total Catastrophic (% Reduction)	Increase in MTTF
Unmitigated	35,548	54,570	4,450	1,548	96,116	—	5,998 (—)	—
TMR	277,714	0	0	4	277,718	181,598 (189%)	4 (99.93%)	1,499.5×
RPR+TMR	96,027	82,516	136	7	178,686	82,570 (85.91%)	143 (97.62%)	41.944×

B. Results

Table II shows the results for the binary PAM receiver system. The results are similar to those observed for the feed-forward system. Specifically, TMR again eliminated virtually all of the catastrophic SEUs. The system with the combination of RPR and TMR (RPR+TMR) successfully reduced the number of catastrophic bits by over 97%.

These results confirm that RPR is a viable option for mitigating catastrophic SEUs in a recursive communications system as well. Though TMR nearly perfectly protected the system, the overhead cost was predictably near 200%. The RPR+TMR design was effective at significantly reducing catastrophic SEUs and increasing the MTTF of the system by 42× at a cost less than half that of TMR.

VII. CONCLUSIONS

The fault injection experiments presented in this paper focused on relatively simple matched filter demodulators for binary PAM. However, the results and conclusions easily extend to demodulators for high-order modulations. The basic conclusion here is that targeted SEU mitigation is capable of producing excellent performance at a fraction of the cost of full-blown TMR. This conclusion is based on extensive fault injection tests performed to the simple demodulators. We demonstrate that Shim's RPR is a very good solution to the problem.

In the case of logic-based FIR filters, the RPR systems demonstrated used about half as many configuration bits as the TMR system (corresponding to an overhead cost that is one-quarter that of TMR), which is a significant savings in circuit area and power. For filters using the embedded DSP blocks available in some FPGA devices, RPR may not be sufficient as a lower cost alternative to TMR. In this experiment RPR had about the same cost as TMR in terms of area. In each case, however, the MTTF of the system was increased by over 20 times.

For the binary PAM system with the recursive timing synchronization loop, a mixture of RPR and TMR was demonstrated. This configuration increased the MTTF of the system by over 40× at a cost of less than half that of TMR.

While this paper only demonstrates RPR on a small set of test cases, these are the first experiments using fault injection that fully characterize the effect of each configuration SEU in the FPGA. This comprehensive study strongly suggests that the techniques will work on other test cases. Future work could include examinations of RPR for protecting a wider array of computation modules as well as evaluations of the cost and benefits of RPR in larger systems. Future analysis could also focus on the trade-offs of changing the RPR redundancy factor k, effectively applying more or less protection to the system.

REFERENCES

[1] Cummings, M. and Haruyama, S.
FPGA in the software radio.
IEEE Communications Magazine, **37**, 2 (Feb. 1999), 108–112.

[2] Caffrey, M.
A space-based reconfigurable radio.
In T. P. Plaks and P. M. Athanas (Eds.), *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms* (ERSA '02), Las Vegas, NV, June 2002, pp. 49–53.

[3] Banu, R. and Vladimirova, T.
Fault-tolerant encryption for space applications.
IEEE Transactions on Aerospace and Electronic Systems, **45**, 1 (Jan. 2009), 266–279.

[4] Adell, P., et al.
Digital control for radiation-hardened switching converters in space.
IEEE Transactions on Aerospace and Electronic Systems, **46**, 2 (Apr. 2010), 761–770.

[5] Dodd, P. and Massengill, L.
Basic mechanisms and modeling of single-event upset in digital microelectronics.
IEEE Transactions on Nuclear Science, **50**, 3 (June 2003), 583–602.

[6] Johnson, E., Wirthlin, M. J., and Caffrey, M.
Single-event upset simulation on an FPGA.
In T. P. Plaks and P. M. Athanas (Eds.), *Proceedings of the International Conference on Engineering of Reconfigurable Systems and Algorithms* (ERSA), Las Vegas, NV, June 2002, pp. 68–73.

[7] Ostler, P., et al.
SRAM FPGA reliability analysis for harsh radiation environments.
IEEE Transactions on Nuclear Science, **56**, 6 (2009), 3519–3526.

[8] Carmichael, C.
Triple module redundancy design techniques for Virtex FPGAs.
Xilinx Corporation, Technical Report, Nov. 1, 2001, xAPP197 (v1.0).

[9] Morgan, K., et al.
A comparison of TMR with alternative fault-tolerant design techniques for FPGAs.
IEEE Transactions on Nuclear Science, **54**, 6 (2007), 2065–2072.

[10] Huang, K-H. and Abraham, J.
Algorithm-based fault tolerance for matrix operations.
IEEE Transactions on Computers, **C-33**, 6 (June 1984), 518–528.

[11] Reddy, A. and Banerjee, P.
Algorithm-based fault detection for signal processing applications.
IEEE Transactions on Computers, **39**, 10 (Oct 1990), 1304–1308.

[12] Ruano, O., et al.
An experimental analysis of SEU sensitiveness on system knowledge-based hardening techniques.
Proceedings of the 10th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS '07), Kraków, Poland, Apr. 11–13, 2007, pp. 1–6.

[13] Pratt, B., et al.
Reliable communications using FPGAs in high-radiation environments—Part I: Characterization.
Proceedings of the 2010 IEEE International Conference on Communications (ICC), Cape Town, South Africa, May 2010, pp. 1–5.

[14] Shim, B. and Shanbhag, N.
Reduced precision redundancy for low-power digital filtering.
Conference Record of the Thirty-Fifth Asilomar Conference on Signals, Systems and Computers, vol. 1, Pacific Grove, CA, 2001, pp. 148–152.

[15] Shim, B., Sridhara, S., and Shanbhag, N.
Reliable low-power digital signal processing via reduced precision redundancy.
IEEE Transactions on Very Large Scale Integration (VLSI) Systems, **12**, 5 (2004), 497–510.

[16] Shim, B. and Shanbhag, N.
Energy-efficient soft error-tolerant digital signal processing.
IEEE Transactions on Very Large Scale Integration (VLSI) Systems, **14**, 4 (2006), 336–348.

[17] Reviriego, P., Maestro, J., and Liu, S-F.
Efficient soft error-tolerant adaptive equalizers.
IEEE Transactions on Circuits and Systems I: Regular Papers, **57**, 8 (Aug. 2010), 2032–2040.

[18] Snodgrass, J.
Low-power fault tolerance for spacecraft FPGA-based numerical computing.
Ph.D. dissertation, Dept. of Electrical and Computer Engineering, Naval Postgraduate School, Monterey, CA, Sept. 2006.

[19] Sullivan, M. A.
Reduced precision redundancy applied to arithmetic operations in field programmable gate arrays for satellite control and sensor systems.
Master's thesis, Dept. of Mechanical and Astronautical Engineering and Dept. of Electrical and Computer Engineering, Naval Postgraduate School, Monterey, CA, Dec. 2008.

[20] Rollins, N., et al.
Evaluating TMR techniques in the presence of single event upsets.
Proceedings of the 6th Annual International Conference on Military and Aerospace Programmable Logic Devices (MAPLD '03), Washington, D.C., Sept. 9–11, 2003, paper P63.

[21] Rice, M.
Digital Communications: A Discrete-Time Approach (1st ed.).
Upper Saddle River, NJ: Prentice-Hall, 2009.

[22] Pratt, B., et al.
Fine-grain SEU mitigation for FPGAs using partial TMR.
IEEE Transactions on Nuclear Science, **55**, 4 (2008), 2274–2280.

[22] TMRTool User Guide, UG156
Xilinx, Inc., 2004.

Brian Pratt received his bachelor's degree in computer engineering in 2005 and his doctoral degree in electrical and computer engineering in 2011, both from Brigham Young University in Provo, UT.

He currently works as a digital design engineer at L-3 Communications in Salt Lake City, UT. His professional interests include reconfigurable computing, FPGA reliability, digital communications, and embedded systems.

Megan Fuller graduated from the Department of Electrical and Computer Engineering at BYU in April of 2012.

As an undergraduate she participated in a number of research activities, including the reliable FPGA design in the BYU Reconfigurable Computing Laboratory and the design of beamforming circuits for the BYU Radio Astronomy Laboratory. Her interests include digital design and signal processing. She is currently pursuing her Ph.D. in electrical engineering at MIT.

Michael Rice (M'82—SM'98) received a B.S.E.E. from Louisiana Tech University in 1987 and his Ph.D. from Georgia Tech in 1991.

He was with Digital Transmission Systems, Inc. in Atlanta and joined the faculty at Brigham Young University in 1991, where he is currently the Jim Abrams Professor in the Department of Electrical and Computer Engineering. He was a NASA/ASEE Summer Faculty Fellow at the Jet Propulsion Laboratory during 1994 and 1995, where he worked on land mobile satellite systems. During the 1999–2000 academic year, he was a visiting scholar at the Communication Systems and Signal Processing Institute at San Diego State University. His research interests are in the area of digital communication theory and error control coding, with a special interest in applications to telemetering and software radio design. He has been a consultant to both government and industry on telemetry-related issues.

Dr. Rice is a member of the IEEE Communications Society and served as the Chair of the Communication Theory Technical Committee from 2009–2011. He is currently serving as the technical editor for Command, Control and Communication Systems for *IEEE Transactions on Aerospace and Electronic Systems*.

Mike Wirthlin (M'92—SM'04) received his B.S. and Ph.D. degrees from Brigham Young University (BYU) in 1992 and 1997, respectively.

After completing his Ph.D., he worked as a staff research engineer in the Systems Architecture Laboratory at the National Semiconductor Corporation in Santa Clara, CA. He is currently a professor in the Department of Electrical and Computer Engineering at BYU in Provo, UT. He has been actively involved in FPGA design and research for over 20 years. His research interests include reliable FPGA design, fault tolerant computing, and configurable computing systems.

Dr. Wirthlin is the author of a variety of technical papers in these research areas. He has led the development of a number of tools and techniques for improving the reliability of FPGA designs. He is a senior member of the IEEE and a member of the ACM. He is active in the organization and is planning of a number of FPGA-related conferences.