Improving F_{max} of FPGA Circuits Employing DPR to Recover from Configuration Memory Upsets

Ediz Cetin
Australian Centre for Space Engineering Research
School of Electrical Engineering and Telecommunications
University of New South Wales, Sydney, Australia
e.cetin@unsw.edu.au

Oliver Diessel, Lingkan Gong
School of Computer Science & Engineering
University of New South Wales,
Sydney, Australia
{odiessel, lingkang}@cse.unsw.edu.au

***Abstract*—*Field-Programmable Gate Arrays* (FPGAs) provide an ideal platform for meeting the performance, cost and flexibility requirements of on-board processing in spacebourne applications. However, given the reliance on SRAM-based configuration memory, off-the-shelf FPGAs are vulnerable to radiation-induced *Single Event Upsets* (SEUs). The detection and mitigation of the effects of SEUs is therefore of paramount importance. Moreover, in time critical applications, it is also desirable to detect and recover from errors rapidly. Techniques for partially reconfiguring a corrupted module of a *Triple Modular Redundant* (TMR) implementation have been described in the literature. In this paper we address the speed penalty incurred with such techniques and provide a generalized approach for alleviating it. The results indicate that the speed penalty can be greatly reduced enabling rapid recovery from SEUs in reconfigurable hardware.**

Keywords—fault tolerance; radiation induced errors; reconfigurable hardware;

I. INTRODUCTION

Growing international interest in the development of space missions based on low-cost nano-/microsatellites demands new approaches to the design of reliable, low-cost, reconfigurable digital processing platforms. The implementation devices most suited to meeting these requirements are *Commercial-Off-The-Shelf* (COTS) *Field-Programmable Gate Arrays* (FPGAs). However, the main challenge to deploying reconfigurable systems in space is mitigating the impact of radiation-induced *Single Event Upsets* (SEUs). Recovery from radiation-induced configuration errors involves error detection, reconfiguration to eliminate the error, and state restoration e.g. re-synchronization to re-establish maximum reliability.

There are a number of approaches that have been reported in the literature dealing with SEUs in FPGAs [1] – [7]. One of the most commonly used SEU mitigation approaches is *Triple Modular Redundancy* (TMR). With this approach the user design is triplicated with each module operating in parallel. The outputs of these modules are then fed into a voting circuit that arbitrates between them. When a triplicated module is acyclic i.e. contains no feedback paths, SEUs are detected as a transient error in the output. TMR ensures that faults in a single module are detected and "overruled" by majority vote and that the faulty module can be identified. Furthermore, a single transient error will clear itself after a single erroneous cycle since the data processed in the immediately following cycle won't have been affected by the SEU and there is no feedback path for the original error to be stored or circulated. However, when an SEU is trapped in a feedback path e.g. in a cyclic component, or the configuration memory of the FPGA is affected, errors can persist. These faults can also be detected using TMR by observing successive errors in a single module. However, TMR on its own does not provide a means of correcting these faults. Thus TMR is typically combined with *configuration memory scrubbing* [2], or with *Dynamic Partial Reconfiguration* (DPR) [4], [7]. Scrubbing has a relatively long average recovery time and considerable energy penalty with respect to DPR. Using DPR, persistent faults are recovered by partially reconfiguring the erroneous module at run-time while the remaining modules of a TMR component continue to operate. One of the challenges of such an approach is how to re-synchronize the newly re-configured TMR module before it could be used for checking the correctness of its siblings. This detail is less comprehensively studied in the literature.

For an acyclic module, which has just been reconfigured, waiting until the inputs propagate to an output would suffice for re-synchronization [7]. This is not the case for cyclic modules and to re-synchronize, a cyclic component must be linearized i.e. all feedback edges are cut and voted upon and re-enter the new component, which is now acyclic, as inputs [7]. This technique is illustrated in Fig. 1.

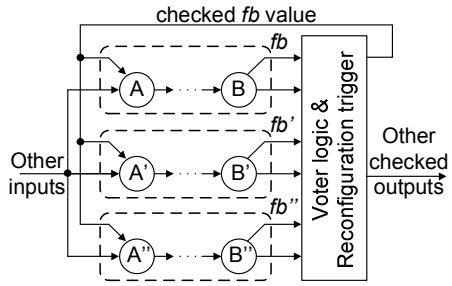

Fig. 1. TMR implementation of cycle AB.

After a module is reconfigured, its state is resynchronized with that of its siblings when the inputs to the module (including any feedback edges that have been voted upon) have emerged as outputs. The latency of the component therefore determines the resynchronization delay. In addition, some

means of coordinating the requests for reconfiguration between many voters and the reconfiguration controller also needs to be implemented. In [8], we have outlined and assessed a token-ring architecture we use to implement a *Reconfiguration Control Network* (RCN) for this purpose.

Unfortunately, the use of TMR results in considerable hardware overhead as well as increased power consumption and the insertion of the voter circuitry results in reduced operating frequency. This is further aggravated by voting on all feedback edges and feeding back the voted results to enable rapid re-synchronization, as depicted in Fig. 1. Internal feedback paths that were short prior to triplicating and voting can be significantly lengthened by inserting voters and routing feedback signals via the static circuit region back into the (dynamically reconfigurable) modules.

In this paper we propose a generalized approach that can be applied to any type of circuit to alleviate the speed penalty in TMR-based SEU mitigation and recovery approaches. In this paper we do not address the area overhead as this has been extensively studied in the literature [6], [9] – [13]. The paper is organized as follows Section II assess the overheads associated with TMR and DPR approaches while our proposed approach for alleviating them is given in Section III with evaluation results in Section IV and concluding remarks in Section V.

II. TMR AND DPR OVERHEADS

To assess operating frequency overheads associated with TMR and TMR combined with DPR techniques, we have used representative satellite-based signal processing circuits namely a single multiply accumulator (MAC) based 21-tap *Finite Impulse Response* (FIR) filter with 16-bit signal and 32-bit accumulator values, and a *Block Adaptive Quantizer* (BAQ).

For Earth observation applications, BAQ is the most commonly used technique for on-board *Synthetic Aperture Radar* (SAR) data compression [14], [15]. With BAQ, SAR data (both the *In-phase* (I) and *Quadrature* (Q) channels) are divided into blocks of size N and the standard deviation of each block is used to adapt the quantizer threshold values for that block. In this way, the raw SAR data, which is originally m-bits wide, is optimally quantized block-by-block using $n < m$-bits [16]. Fig. 2 depicts the BAQ architecture we studied.

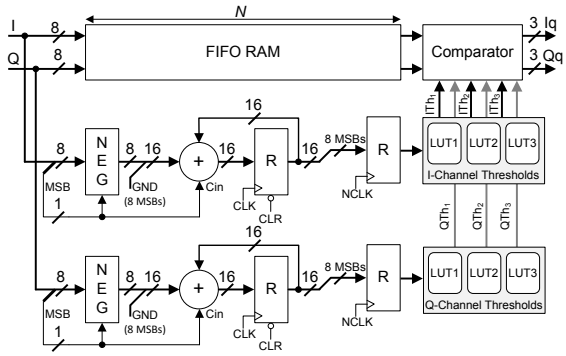

Fig. 2. BAQ Architecture

In our design, N was chosen to be 256 samples, $m = 8$ and n was set to 3-bits. The standard deviation calculation was replaced with an average magnitude calculation, |I|, |Q|, to eliminate the need for a square root operation [14], [15]. The quantization threshold values, ITh/QTh, were determined by calculating the average magnitude for a block of N raw SAR data samples and looking up the corresponding thresholds in *Look-Up Tables* (LUTs). The design thus takes 256 cycles to calculate |I| and |Q| for a particular block and a further 256 cycles to realize the quantization. These two operations can be overlapped for successive blocks of N samples to obtain a continuous output stream.

These representative signal processing circuits were hand coded in Verilog to enable us to manually extract the feedback signals present in the data- and control-paths. Designs were implemented using ISE and PlanAhead 12.4 tools and targeted at a Xilinx Virtex-5 XC5VFX70T speed grade -2 FPGA. The results of synthesizing, floor-planning and implementing the circuits are given in Table I. Results for each design are grouped into sets of 3 rows. In each set of rows, resource utilizations, operating frequency in MHz and throughput in Msamples/s for an optimized single module base component (-1 suffix), conventional triplicated TMR component with associated voter (-3 suffix) and TMR combined with DPR (-3R suffix) incorporating circuitry needed for triggering reconfiguration and re-synchronizing cyclic components for FIR and BAQ designs are given. The table also depicts the bitstream length in 32-bit words and the latency of a module in *Clock Cycles* (CC) for the DPR implementations.

TABLE I. RESULTS OF MAPPING FIR AND BAQ DESIGNS TO XILINX XC5VFX70T-2 DEVICE

Design Name	Utilization			Freq/Thruput (MHz/Msps)	Bit. Len. (wd)	Lat. (CC)
	LUT	FF	DSP			
FIR-1	32	10	1	446/20.3		
FIR-3	167	27	3	287/13.1		
FIR-3R	789	366	3	224/10.2	4100	462
BAQ-1	362	187	-	400/400		
BAQ-3	1122	561	-	326/326		
BAQ-3R	1648	791	-	243/243	7380	512

As can be observed from Table I, the optimized (-1 suffix) base designs (rows 1 and 4) suffered considerable slowdown, 50% for FIR and 39% for BAQ, when triplicated and when feedback edges were cut and the voter logic was inserted (-3 suffix, rows 2 and 5). Significantly, to allow the reconfigurable modules (in the dynamic region) to be connected to the static voter logic and feedback paths, so-called proxy pins are inserted and these cause considerable slowdown (-3R suffix, rows 3 and 6). An examination of the critical paths revealed that the insertion of the voters and proxy pins into the feedback paths of both designs had a significant effect on performance. While we have observed as much as a doubling in the path delay, it may increase by a larger factor if the frequency of the loop or the component prior to triplication is high.

Fig. 3 illustrates the delay recorded for the feedback paths of the three FIR designs listed in rows 1–3 of Table I. The critical path length for FIR-1 was found to be about 2.24ns. For FIR-3, the total delay of the path including the voter (V) was over 3.44ns, and for FIR-3R, with proxy pins (P) inserted into the path as well, the delay was over 4.40ns.

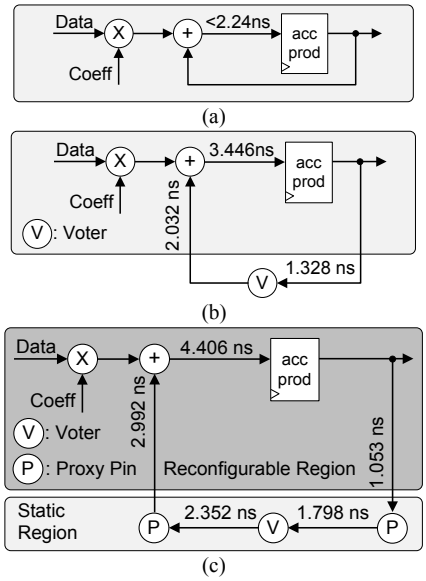

Fig. 3. FIR feedback path delays (a) FIR-1, (b) FIR-3, (c) FIR-3R

In the following section we propose a generalized approach that can be applied to any type of circuit to alleviate the speed penalty in TMR-based approaches.

III. Reducing The Speed Penalty

The proposed approach to alleviate the speed penalty is based upon the idea that the increase in critical path delay that is obtained when feedback signals are voted upon can be reduced by pipelining these signal paths. The consequence of pipelining is that the feedback signals are delayed with respect to the stream of inputs being processed by the component. To realign the delayed feedback values with the inputs, we perform a correspondingly large prefix operation on the inputs. This is illustrated in Fig. 4.

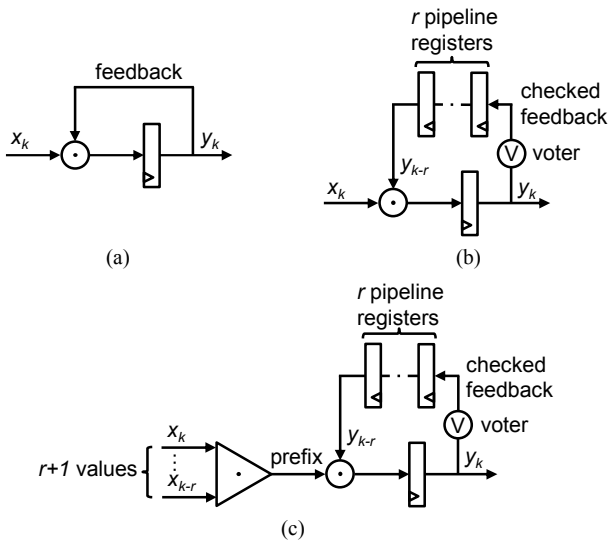

Fig. 4. Reducing feedback path delay in a run-time reconfigurable triplicated component (a) simple feedback loop, (b) pipelined feedback (c) with prefix correction

In Fig. 4(a) an example of a simple feedback loop in a component is illustrated. Combining input x_k and feedback value y_k in time step k using binary operator $\odot$ produces output y_{k+1} in time step $k+1$. Such operators may be arithmetic (+, -, ×, max, etc.) or logical (and, or, etc.). The diagram includes a register that synchronously stores the feedback value. The register output and feedback value can be expressed as $y_{k+1} = x_k \odot y_k$. When the component is triplicated and prepared for run-time reconfiguration using DPR, the feedback path is cut in order to vote on the signal value y_k. Doing so allows a newly reconfigured module to recover and store the correct feedback value once the x_k input to the $\odot$ operator is correctly received. Note that this latter condition may take some time if the illustrated feedback loop is buried deeply within the component.

Assuming the delay of the feedback path is increased by a factor of less than $r+1$ with respect to the critical delay of the remaining paths in the component, our approach to reducing the feedback path delay is to insert r pipeline registers along the path as illustrated in Fig. 4(b). Where the registers are inserted along the path is not too important as long as the resulting cycle period is minimized. As a result, the feedback signal is delayed by r clock cycles relative to the one cycle delay it experienced in the circuit of Fig. 4(a). To address the relative shift in timing between input x_k and checked but delayed feedback y_{k-r}, we perform an r-step prefix operation on the inputs x_i. That is, since $y_{k+1} = x_k \odot y_k$, we can also derive $y_{k+1} = x_k \odot x_{k-1} \odot y_{k-1} = \ldots = x_k \odot x_{k-1} \odot \ldots \odot x_{k-r} \odot y_{k-r}$. The corresponding circuit is illustrated in Fig. 4(c). In particular, if $\odot$ is the addition operation, Fig. 4(a) illustrates an accumulator with recurrence equation $y_{k+1} = x_k + y_k$ and the equation corresponding to Fig. 4(c) is $y_{k+1} = x_k + x_{k-1} + \ldots + x_{k-r} + y_{k-r}$.

The principles of the above argument apply equally to more complex feedback structures. Fig. 5(a) depicts a linear sequence (chain) of operations that a feedback value passes through before it affects the feedback value. If inserting the voter and proxy pins requires us to also insert r pipeline registers into the feedback path, then a prefix operation of length r also needs to be added to each node in the chain that depends upon external inputs as shown in Fig. 5(b). If a node in the chain is a unary operator (has no other input other than the output of the previous operator in the chain), it does not require modification.

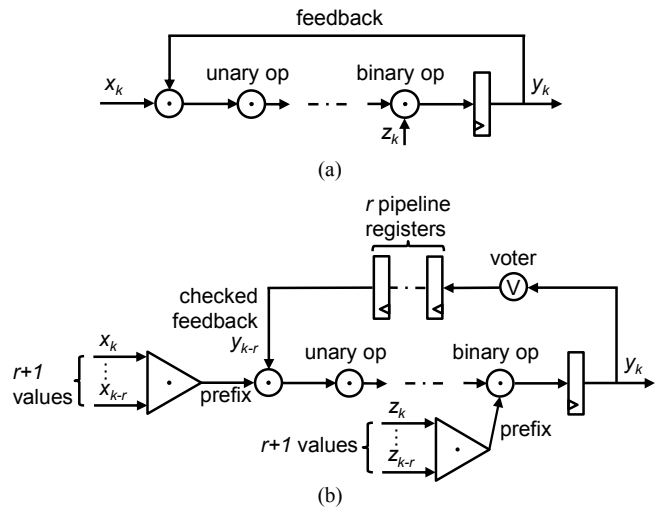

Fig. 5. Transformation of complex feedback loops

IV. PERFORMANCE EVALUATION

We have evaluated the efficacy of the proposed approach for our FIR and BAQ designs and the results are given in Table II targeting Xilinx Virtex-5 XC5VFX70T speed grade -2 FPGA. As it can be observed from Table II for the FIR design, we found that inserting 2 pipeline stages into the voted on feedback signal path was optimal. As the critical path delay of BAQ-3R was less than twice that of BAQ-1, only one pipeline stage was inserted into its feedback path. The results for these "pipelined" designs are listed with the letter P appended to the suffix of the design name in Table II together with the number of pipeline stages (FB Stages) inserted into the feedback signal.

As can be observed from Table II, the application of the proposed approach (-3RP suffix) resulted in an operating frequency reduction of 12.5% (FIR-3RP) and 15.8% (BAQ-3RP) with respect to the operating frequencies of the FIR (FIR-1) and BAQ (BAQ-1) baseline designs. These reductions compare very favorably with reductions of almost 50% (FIR-3R) and 39% (BAQ-3R) as observed for the non-pipelined designs respectively relative to their baselines. The operating frequencies of the FIR/BAQ-3RP designs, incorporating the proposed approach, achieved improvements of 37.5% and 23.2% respectively over their FIR/BAQ-3R counterparts.

TABLE II. RESULTS OF MAPPING FIR AND BAQ DESIGNS TO XILINX XC5VFX70T-2 DEVICE

Design Name	FB Stages	Utilization			Freq/ Thruput (MHz/ Msps)	Bit. Len. (wd)	Lat. (CC)
		LUT	FF	DSP			
FIR-1P	2	37	84	4	421/16.8		
FIR-3P	2	182	225	12	407/16.3		
FIR-3RP	**2**	**804**	**588**	**12**	**390/15.6**	**4100**	**525**
BAQ-1P	1	372	261	-	405/405		
BAQ-3P	1	1143	783	-	384/384		
BAQ-3RP	**1**	**1678**	**1013**	**-**	**333/333**	**7380**	**512**

V. CONCLUSIONS

In this paper we have presented and detailed a generalized technique for improving the performance of FPGA-based circuits that are required to be reliable and offer high availability and that are subject to radiation-induced SEUs. We have explained and evaluated a technique for alleviating the reduction in the operating frequency experienced when TMR is combined with DPR and when feedback edges are cut and voted upon to enable rapid re-synchronization. We are thereby able to recover from errors rapidly without excessive slowdown in the protected circuits.

The efficacy of the proposed approach has been evaluated using two representative circuits and the results indicate that it is possible to alleviate the speed penalty considerably, 37.5% for FIR and 23.2% for the BAQ designs respectively.

REFERENCES

[1] C. Carmichael,M. Caffrey, and A. Salazar, *Correcting single event upsets through Virtex partial configuration*, Jun 2000, XAPP216 (vl.0).

[2] C. Carmichael, "Triple module redundancy design techniques for Virtex FPGAs," *Xilinx App. Note XAPP197*, vol. 1, 2001.

[3] A. Sari and M. Psarakis, "Scrubbing-based SEU mitigation approach for systems-on-programmable-chips," in *Field- Programmable Technology (FPT), 2011 International Conference on*. IEEE, 2011, pp. 1–8.

[4] C. Bolchini, A. Miele, and D. M. Santambrogio, "TMR and partial dynamic reconfiguration to mitigate SEU faults in FPGAs," in *IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT'07)*, Sep 2007, pp. 87–95.

[5] C. Pilotto, J. R. Azambuja, and L. F. Kastensmidt, "Synchronizing triple modular redundant designs in dynamic partial reconfiguration applications," in *ACM 21st annual symposium on Integrated circuits and system design (SBCCI'08)*, 2008, pp. 199–204.

[6] E. Johnson and M. Wirthlin, "Voter insertion algorithms for FPGA designs using triple modular redundancy," in *18th annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA'10)*, 2010, pp. 249–258.

[7] E. Cetin, O. Diessel, L. Gong and V. Lai, "Towards bounded error recovery time in FPGA-Based TMR circuits using dynamic partial reconfiguration", *International Conference on Field-Programmable Logic* (FPL), September 2013.

[8] E. Cetin, O. Diessel, L. Gong and V. Lai, "Reconfiguration network design for SEU recovery in FPGAs," *IEEE International Symposium on Circuits and Systems*, pp. 1524 – 1527, June 2014

[9] B. Pratt, M. Caffrey, F. J. Carroll, P. Graham, K. Morgan, and M. Wirthlin, "Fine-Grain SEU mitigation for FPGAs using partial TMR," *IEEE Transactions on Nuclear Science*, vol. 55, no. 4, pp. 2274–2280, Aug 2008.

[10] K. P. Samudrala, J. Ramos, and S. Katkoori, "Selective Triple Modular Redundancy (STMR) based Single-Event Upset (SEU) tolerant synthesis for FPGAs," *IEEE Transactions on Nuclear Science*, vol. 51, no. 5, pp. 2957–2969, Oct 2004.

[11] V. Chandrasekhar, N. S. Mahammad, V. Muralidaran, and V. Kamakoti, "Reduced Triple Modular Redundancy for tolerating SEUs in SRAM-based FPGAs," *Military and Aerospace Programmable Logic Devices Conference* (MAPLD), 2005.

[12] K. Morgan, M. Caffrey, P. Graham, E. Johnson, B. Pratt, and M. Wirthlin, "SEU-induced persistent error propagation in FPGAs," *IEEE Transactions on Nuclear Science*, vol. 52, no. 6, pp. 2438 – 2445, Dec 2005.

[13] L. F. Kastensmidt, L. Sterpone, L. Carro, and R. M. S, "On the optimal design of triple modular redundancy logic for SRAM-based FPGAs," *Design, Automation and Test in Europe*, vol. 2, pp. 1290 – 1295, Mar 2005.

[14] R. Kwok and W. Johnson, "Block adaptive quantization of Magellan SAR data," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 27, no. 4, pp. 375 –383, Jul 1989.

[15] G. Kuduvalli, M. Dutkiewicz, and I. Cumming, "Synthetic aperture radar signal data compression using block adaptive quantization," *Goddard Space Flight Center Science Information Management and Data Compression Workshop*, pp. 43 – 57, 1994

[16] J. Max, "Quantizing for minimum distortion," *IRE Transactions on Information Theory*, vol. 6, no. 1, pp. 7 – 12, March 1960.