Radiation-Hardened DSP configurations for implementing arithmetic functions on FPGA

Marcos Sanchez-Elez, Inmaculada Pardines, Felipe Serrano, Hortensia Mecha
Dept. Arquitectura de Computadores y Automatica
Universidad Complutense de Madrid UCM
Madrid, Spain
marcos@ucm.es

Abstract— **This paper presents a study of different implementations of arithmetic operations on FPGAs. Radiation vulnerability has been analyzed for each implementation using the fault injection platform NESSY. Results in terms of area, delay and reliability are presented. We propose to build a library of HDL templates taking into account the performed tests. This library is used during the design process with a synthesis tool that implements digital circuits as reliable as possible. Experimental results show that implementations using DSP slices are the ones which achieve better results.**

Keywords—fault injection; DSP48 slices; FPGA; reliability; HDL templates; single event upset

I. Introduction

Performance and flexibility features of FPGAs make them very attractive for aerospace applications. However, the high amount of radiation of these environments can modify their functionality [1]. Radiation effect appears as an ionized particle that changes the content of a memory cell. This is the so-called Single Event Upset (SEU). Changes in the configuration memory [2], [3] may lead to serious errors in the normal operation of the system. It is important to use techniques to neutralize or mitigate the effect of SEUs.

During the last ten years researches have worked hard in this topic. A summary of the most relevant methods proposed is made by Siegle et al. in [4]. Triple Modular Redundancy (TMR) [5], [6] is a widespread masking technique used to protect circuits against SEUs. It consists in replicating the circuit three times and processing the three obtained outputs by voting them in order to generate a single output.

Other interesting alternative is the failure recovery technique called *scrubbing*, which consists in refreshing the memory to reconfigure the whole circuit [7], [8]. This solution generates a performance overhead that can be unaffordable in some cases. A lot of works have been made in order to solve this problem. An example of this is the SEU detection and recovery mechanism proposed by Legat et al. [9]. They propose a SEU recovery mechanism through the ICAP, with a small hardware overhead and hardened with the TMR technique.

A different solution to increase the robustness of digital circuits on FPGAs is implementing them by using the embedded DSPs existing in the target FPGA. The goal of this work is to develop a tool to generate hardened circuits against SEUs, based on the use of DSP48E slices. Serrano et. al [10] show that the implementation of a filter using DSP slices improves circuit reliability against SEUs. Based on this result and with the aim of mitigating the effects of SEUs on the configuration memory, this paper performs a detailed study of the failures produced by a bit-flip on circuits that implement the most used arithmetic operations. Moreover, a solution to obtain a more hardened circuit is to use a TMR scheme. This scheme will protect the content of the registers. Therefore, several TMR implementations are also studied. However, their use could increase up to three times the configuration bits, and then the possibility of failure. As a result of this study we create a library with the most reliable implementations.

A SEU injection platform named NESSY [11] has been used in order to perform this study. NESSY injects a SEU by modifying a bit in the region of the configuration memory that implements the circuit under test, including regions that contain configuration information about DSPs. This fault injection platform stands out by its high performance and non-intrusiveness features.

The remainder of the paper is organized as follows: Section II proposes a DSP based methodology to implement reliable digital circuits. Section III discusses different hardened against SEUs DSP configurations to join the H-CIS library. Section IV presents the obtained experimental results. Finally, Section V shows the conclusions of this article.

II. DSP48E Methodology

This work is part of a DSP methodology that, given a synthesizable VHDL code as an input, generates automatically another VHDL code as an output [12]. This new code implements some functions of the entry code over DSP48E blocks preserving the original functionality. Also, these DSP48E blocks have been hardened against SEUs.

Several concepts have to be defined in order to give theoretical support to the methodology. First, we define an Operation Class (OC) as an arithmetic operation type, such as addition/subtraction, multiplication or multiply accumulate. Each Operation Class has associated different templates to be implemented in a DSP. These templates implement the operation by using different DSP configurations.

A specific implementation of an OC in the DSP is known as Class Instance (CI). A CI has an associated VHDL template that uses DSP48E blocks with a specific port-map

This work was supported by the Spanish Ministry of Education, Culture and Sports under grant TIN2013-40968-P

(input and output signals), clock and reset signals. There could be as many CIs as different OCs. In the same way, a specific OC could have associated several CIs, that is, different implementations in a DSP48E of the same arithmetic operation. The methodology selects the CIs to implement the operations of the original VHDL code.

The goal of this work is to build a CIs library (H_CIS) to mitigate the effects of SEUs over digital circuits implemented in a FPGA. Therefore, we analyze the sensibility against SEUs of different CIs. The most hardened ones are added to the library. Then, the designer could decide to use a CI of this library to implement an OC.

III. Implementation of TMR on DSP48E slices

The robustness of digital circuits against SEUs improves when a large part of the circuit is implemented using DSPs instead of CLBs [10]. In this section we analyze all the possible implementations of a circuit with DSPs in order to obtain hardened designs against SEUs. We use different templates for the DPS48E primitive [13] to achieve higher versatility and to specify the design in a very detailed level.

DSP48E can perform different operations. Its functionality is controlled by a combination of dynamic control signals and static parameter attributes. Dynamic control signals allow the DSP48E to run in different configuration modes in each clock cycle. Moreover, DSP48E blocks can be used as pattern detectors. We use this configuration to implement a voter, as shown in Fig. 1. FPGAs have a limited number of DSPs, so it is important to reuse them implementing different operations concurrently. Fig. 2 shows a DSP configuration with a multiplier and a voter. So, we take advantage of the versatility of DSPs by using them to implement arithmetic operations (DSP(O)) or the comparator of the voter (DSP(=)).

TMR technique is applied to mitigate SEUs effect, but its use could not decrease the error probability because the SEUs can affect to the configuration bits of the voter. A solution to harden TMR is to triplicate the voter. Then, we study the robustness of different implementations of TMR using a tripled voter. For example, we can set the arithmetic operations and the comparators in different DSP blocks (see Fig. 3). We compare this solution with others that uses CLBs to implement the same TMR. One of these proposals implements the triplicated arithmetic operation in CLBs and the comparators in DSPs, and the other implements the comparators in CLBs and the operations in DSPs.

Another possible solution is to implement each arithmetic operation and each comparison in the same DSP provided that both can share the DSP to be executed concurrently (see Fig. 4). This reuse of the DSP decreases the area of the circuit. We also propose a TMR solution with a unique voter as shown in Fig. 5. In this solution, operations 1 and 2 are compared, and the final output will be the result of operation 1 or 3 depending on whether the comparison is true or false. In Fig. 5 the arithmetic operations and the comparator are implemented in DSPs. This solution can be also implemented with the arithmetic operations in CLBs and the comparator in a DSP

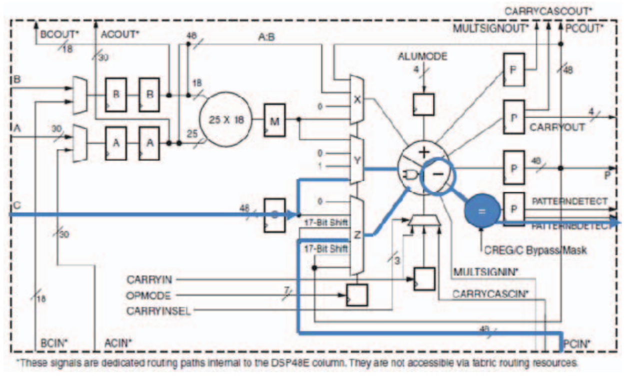

Fig. 1. Voter configuration of a DSP48E.

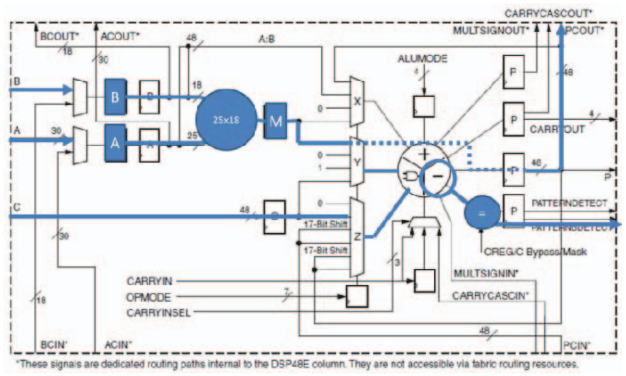

Fig. 2. Multiplier +Voter configuration of a DSP48E.

or with the arithmetic operations in DSPs and the comparator in CLBs. As was explained for the configuration of Fig 4, a DSP can be reused to perform the operation and the comparison using the minimum possible area.

IV. Experimental results

This section presents the SEU-emulation experimental results obtained for different implementations of arithmetic operations. Some of these implementations take advantage of using DSP48E slices. Our goal is to analyze the effect of a bit-flip in the configuration memory in order to build a library of hardened templates. So, the designer can decide, at the beginning of the design process, to use any template of this library.

For this work a Virtex 5 FPGA has been used. However, the results presented in this paper are extensible to any other Xilinx Virtex device.

The fault injection platform used to analyze the reliability of the proposed implementations against SEUs is NESSY [10], [11]. The SEU is injected without introducing any modification in the placement and routing of the circuit under test except the bit-flip.

We have implemented three different arithmetic functions as benchmarks (add, multiply and multiply-accumulate) to study the sensitivity against bit-flips in the circuits described in section III.

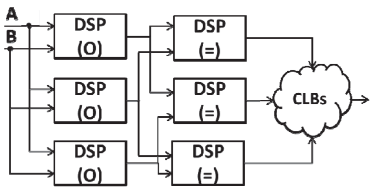

Fig. 3. TMR with tripled voter implemented using DSPs.

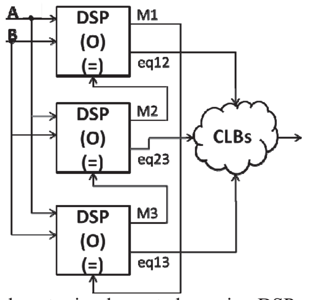

Fig. 4. TMR with triple voter implemented re-using DSPs.

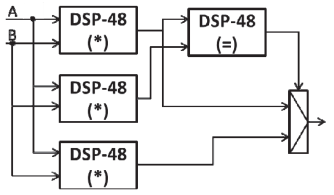

Fig. 5. TMR with a unique voter implemented using DSPs.

We have done an exhaustive study using NESSY to emulate a large number of bit-flips. As an example, NESSY has performed 1,546,240 bit-flips for the largest implemented circuit.

Table I describes the implemented experiments. We have used the following nomenclature:

- Operation name followed by the size of the operands when the circuit is implemented in CLBs (for example, *mult16*).
- Operation name followed by the size of the operands and the acronym DSP if the circuit is implemented in a DSP48E slice (*mult16dsp*, see Table I).
- The same nomenclature described above followed by "_v" when the circuit implements the arithmetic operation with TMR. We can distinguish four different versions with all their possible combinations depending on which part of the circuit is implemented in CLBs or in DSP48E slices. Versions 1 and 2 represent TMR implementation with triple voter, corresponding to Fig. 3 and Fig. 4 schemes respectively. Versions 3 and 4 implement TMR with a unique voter. In case of v4 a DSP48E slice is reused in order to implement the operation and the voter whereas in v3 there is not DSP reuse (see Fig. 5).

Results of failures are depicted in Table II. Column *CLB Failures* shows the failures affecting the functionality of the CLBs and their routing resources. Column *DSP Failures* presents errors affecting the functionality and the routing resources of DSP48E slices. We distinguish between them to know which part of the new circuit increments the number of failures. Column *ICLB* shows the improvement percentage

TABLE I
EXPERIMENTS DESCRIPTION. SIZE IS MEASURED IN TERMS OF NUMBER OF DSPS + CLB SLICES

Experiment	Operation	Voter		Size	Period (ns)
		HW	Description		
add16	CLBs	No	--	48	1.89
add16_v3	CLBs	DSP	1 voter	1 + 34	4.60
add16dsp	DSP	No	--	1+0	2.91
add16dsp_v1	DSP	DSP	3 voter	6+16	6.47
add16dsp_v3.0	DSP	CLBs	1 voter	3+22	5.27
add16dsp_v3.1	DSP	DSP	1 voter	4+16	6.17
mul16	CLBs	No	--	0+430	6.83
mul16_v1.0	CLBs	CLBs	3 voters	0+1091	6.85
mul16_v1.1	CLBs	DSP	3 voters	3+1058	6.85
mul16_v3.0	CLBs	CLBs	1 voter	0+1069	6.85
mul16_v3.1	CLBs	DSP	1 voter	1+1058	6.85
mul16dsp	DSP	No	--	1+0	2.91
mul16dsp_v1.0	DSP	CLBs	3 voters	3 +80	5.85
mul16dsp_v1.1	DSP	DSP	3 voters	6 +32	6.51
mul16dsp_v2	DSP	DSP	3 voters + reuse	3 +32	10.1
mul16dsp_v3.0	DSP	CLBs	1 voter	3 +48	5.54
mul16dsp_v3.1	DSP	DSP	1 voter	4 +32	6.21
mul16dsp_v4	DSP	DSP	1 voter + reuse	3 +32	3.93
macc16	CLB	No	--	0+462	6.83
macc16dsp	DSP	No	--	1+0	2.91
macc16dsp_v1	DSP	DSP	3 voters	6+32	6.51
macc16dsp_v2	DSP	DSP	3 voters + reuse	3+32	10.1
macc16dsp_v3.0	DSP	CLBs	1 voter	3+43	5.43
macc16dsp_v3.1	DSP	DSP	1 voter	4+32	6.21

TABLE II
EXPERIMENTAL RESULTS

Experiment	#Failures	#CLB Failures	#DSP Failures	ICLB	IDSP
add16	1660	1660	0	--	-38.0%
add16_v3	2425	2249	176	-46%	-135%
add16dsp	1029	0	1029	38.0%	--
add16dsp_v1	1112	960	152	33.0%	-8.06%
add16dsp_v3.0	1031	794	237	37.89%	-0.19%
add16dsp_v3.1	1192	1015	177	28.2%	-15.8%
mul16	12323	12323	0	--	-88.0%
mul16_v1.0	5543	5543	0	55.0 %	-275%
mul16_v1.1	5812	5567	245	52.8%	-74,5%
mul16_v3.0	4507	4507	0	63,4%	-205%
mul16_v3.1	3466	3190	276	71.8%	-57.3%
mul16dsp	1480	0	1480	88.0 %	--
mul16dsp_v1.0	3497	3072	425	71.6%	-136%
mul16dsp_v1.1	1422	1191	231	88.5%	3.92%
mul16dsp_v2	2046	1684	362	83.4%	-38.2%
mul16dsp_v3.0	1618	1331	287	86.9%	-9.32%
mul16dsp_v3.1	1668	1328	340	86.5%	-12.7%
mul16dsp_v4	4487	3012	1475	63.6%	-203.1%
macc16	10811	10811	0	--	87.5%
macc16dsp	1348	0	1348	87.5%	--
macc16dsp_v1	1347	794	553	87.5%	0.00%
macc16dsp_v2	1437	1103	244	86.7%	-6.60%
macc16dsp_v3.0	1624	1301	323	84.98%	-20.47%
macc16dsp_v3.1	1589	1351	238	85.3%	-17.9%

(decrease of failures) of using any of the implementations described in Section III versus an implementation that only uses CLB slices. Finally, column *IDSP* presents the improvement of using the proposed implementations versus a circuit implemented on DSP48E slices. None of the reference circuits uses a TMR scheme.

The analysis of the results presented in Table II shows that the CLB implementation solution always has the highest number of failures. The differences, in terms of failures, among the implemented operations depend on the size of the circuit. So, *add16* (48 CLBs) presents less failures than *macc16* (462 CLBs). On the other hand, the circuit implemented in DSP slices (without TMR) always achieves the best results. However, our experiments only measure the SEU effect on the configuration bits, so a TMR technique is required to protect the content of the registers.

The circuit sensibility changes for the operations implemented on CLBs when TMR technique is applied. The hardware that implements these operations is hardened as is clearly shown in Table II (i. e. *mul16* and any *mul16_vi*). For the experiments in which operations are implemented on DSP48E the most notable change appears in the sensibility of the DSP48 blocks. In this case, a decrease in the number of failures of 80% on average is observed (see i.e. DSP Failure column of *mul16dsp* versus *mul16dsp_v1.1*). However, there is an increase in the number of failures in CLBs due to the implementation of the voter.

TMR implementations obtain quite good results against bit-flips affecting the configuration memory as we can see in Table II. But for some cases, i. e. *mul16dsp_v1.0* and *mul16dsp_v2*, the number of these failures increases. This happens because TMR implies the synthesis of a circuit with more configuration bits, so these circuits could be more sensitive to SEU effects.

After analyzing the obtained results we have decided to include the circuit proposed in Fig. 3 in the H-CIS Library as recommended solution. This means, the use of a triple TMR with the operation and the voter implemented in 6 DSP48E slices (i. e. *mult16dspv1_1*).

In addition, H-CIS Library provides other options when there are not enough DSP blocks available to implement the design. The solution in this case is to synthesize the circuit with one voter, implementing the operation on DSP48E slices (Fig. 5). The voter can be implemented on DSP48E or CLBs, depending on the number of DSP48E available. Evaluating the experimental results for these cases we observe an incremental rate in the number of failures of 9.32% (voter implemented on CLBs) or 12.7% (voter implemented on DSP48E) versus a DSP48E implementation without TMR. We consider this solution suitable to be included in H-CIS Library since TMR scheme solves the failures in the user registers and these failures are not measured in Table II.

Moreover, we have studied similar implementation schemes for add and macc operations obtaining similar results as those described for the multiplication. Results obtained for add operation can be extensible for those operations that only use the ALU (sub, or, xnor …). In the same way results of macc are comparable to mult-add and mult-sub.

V. Conclusions

This paper has presented an experimental study of the robustness of different implementations of arithmetic operations in Xilinx FPGAs. For this purpose, the SEU emulation platform NESSY has been used. The presented experiments demonstrate that the implementation of arithmetic operations on DSP48E slices reduces the number of failures. Moreover, the use of TMR inferred on DSP48E slices is necessary to eliminate the failures affecting the logic registers.

We have built the H-CIS Library from the analysis of the experimental results. We add the templates for the circuit proposed in Fig. 3 to this library. This scheme of triple TMR with the operation and the voter inferred in different DSP48E slices is shown as the best solution to get hardened circuits. The implementation of TMR with a unique voter on DSP48E slices has also been included since it is fairly good to be used in case the number of available DSPs is limited.

References

[1] M. J. Wirthlin, “FPGAs operating in a radiation environment: lessons learned from FPGAs in space,” Journal of Instrumentation, vol. 8, no. 2, p. C02020, 2013.

[2] Xilinx, “Device Reliability Report,” Xilinx User Guides UG116 (v5.12), 2011.

[3] M. Bellato et als., “Evaluating the effects of SEUs affecting the configuration memory of an SRAM-based FPGA,” in Proceedings of the Design, Automation and Test in Europe (DATE’04), vol.1, p. 10584, 2004.

[4] F. Siegle, T. Vladimirova, J. Ilstad, and O. Emam, “Mitigation of Radiation Effects in SRAM-Based FPGAs for Space Applications,” ACM Computing Surveys (CSUR), vol. 47, no. 2, p. 37, 2015.

[5] E. A. Stott, N. P. Sedcole, and P. Y. K. Cheung, “Fault tolerance and reliability in field-programmable gate arrays,” IET Computers & Digital Techniques, vol. 4, no. 3, pp. 196–210, 2010.

[6] L. Sterpone and M. Violante, “Analysis of the Robustness of the TMR Architecture in SRAM-Based FPGAs,” IEEE Transactions on Nuclear Science, vol. 59, no. 5, pp. 1545–1549, 2005.

[7] J. Heiner, B. Sellers, M. Wirthlin, and J. Kalb, “FPGA partial reconfiguration via configuration scrubbing,” in Proceedings of the International Conference on Field Programmable Logic and Applications (FPL), 2009, pp. 99–104.

[8] I. Herrera-Alzu, and M. Lopez-Vallejo, “Design techniques for Xilinx Virtex FPGA configuration memory scrubbers,” IEEE Transactions on Nuclear Science, vol. 60, no. 1, pp. 376–385, 2013.

[9] U. Legat, A. Biasizzo, and F. Novak, “SEU Recovery Mechanism for SRAM-Based FPGAs,” IEEE Transactions on Nuclear Science, vol. 52, no. 5, pp. 2562–2571, 2012.

[10] F. Serrano, J. A. Clemente, and H. Mecha, “A study of the robustness against SEUs of digital circuits implemented with FPGA DSPs,” in Proceedings of Radiation and Its Effects on Components and Systems (RADECS), pp. 1-4, 2013.

[11] F. Serrano, J. A. Clemente, and H. Mecha, “A Methodology to Emulate Single Event Upsets in Flip-Flops Using FPGAs through Partial Reconfiguration and Instrumentation,” IEEE Transactions on Nuclear Science, vol. 62, no. 4, pp. 1617-1624, 2015.

[12] E. De Lucas, “Methodology to synthesize HDL code to make use of DSP blocks presented in FPGAs,”

[13] Xilinx, “Virtex-5 FPGA XtremeDSP Design Considerations,” 2010.