TRANSIENT FAULT-TOLERANT FAST ADDERS IMPLEMENTED IN FPGAS

[1] *Guilherme Corrêa,* [1] *Helen Franck,* [1] *Eduardo Mesquita,* [1] *Luciano Agostini,* [2] *José Luís Güntzel*

[1] Group of Architectures and Integrated Circuits (GACI) – Dept. of Informatics
Federal University of Pelotas (UFPel) – Pelotas, RS, Brazil

[2] System Design Automation Lab (LAPS) – Dept. of Informatics and Statistics
Federal University of Santa Catarina (UFSC) – Pelotas, RS, Brazil

{gcorrea_ifm, hsfranck.ifm, emesquita.ifm, agostini}@ufpel.edu.br, guntzel@inf.ufsc.br

ABSTRACT

Single-Event Transients (SETs) are becoming a real problem in the design of complex electronic systems that are to be fabricated with nanometer CMOS technologies. As long as a typical integrated system may require many arithmetic calculations, the performance of adders under the presence of SETs may influence the whole system performance. In this work we investigate three fast adder architectures protected against SETs using three classical fault-tolerance techniques. Protected and unprotected adders were synthesized for Altera Stratix III FPGAs and the obtained data were used to compare them in terms of resource use, critical delay and power dissipation.

1. INTRODUCTION

Radiation-induced transient faults are becoming a relevant matter in the design of high performance electronic systems that are to be fabricated with state-of-the-art "nanometer" CMOS technology [1]. Due to their low noise immunity, nanometer technologies are more vulnerable to such kind of faults. A transient fault may occur when an ionizing particle hits a sensitive region of a circuit causing a transient voltage pulse. The sensitive regions correspond to the reversed-biased p-n drain junctions of the transistors that are off [1][2]. As CMOS devices continue to shrink, radiation-induced soft errors tend to become more and more frequent in terrestrial environment.

The radiation-induced transient fault may occur in the circuit's combinational part or in a memory element. In the former case, it is referred to as Single-Event Transient (SET) while in the latter case, it is known as Single-Event Upset (SEU) [1].

Until recently, SEUs were considered more seriously than SETs because the probability of a SET to become an error was much lower than a SEU. However, it has been predicted that, due to the continuous shrink of CMOS devices, by the year 2010 the soft error rate due to SET will be as great as the soft error rate of unprotected memories [3]. Therefore, the use of techniques to mitigate SET effects will become mandatory in forthcoming designs.

Arithmetic circuits are found in most electronic systems and very often are responsible for their performance limitation. Adders play a very important role among all arithmetic circuits, mainly because they serve as the basis for practically all other arithmetic operations such as subtraction, multiplication and division. This way, with the advent of new problems associated to the design of state-of-the-art integrated systems, as low immunity to noise and transient faults, it is necessary to investigate architectural choices to implement robust adders and to evaluate each solution in terms of resource use, speed, power and degree of protection.

This paper presents an evaluation of four different adder architectures implemented using three different tolerance techniques. We have described the Ripple-Carry [4], Carry-Select [5], Carry-Lookahead [6] and the Re-computing the Inverse Carry-in [7] adders in VHDL, using the Triple Modular Redundancy [8], Time Redundancy and Duplication With Comparison associated with Time Redundancy [9] techniques.

This article is divided as follows. Section 2 describes briefly each adder architecture and section 3 explains the fault-tolerance techniques that were used. Section 4 presents the experimental results. Section 6 concludes this work and enumerates possible future works.

2. FAST ADDERS

Several fast adder architectures can be found in the literature. Some adders explore architectural modifications that usually require extra hardware in order to accelerate the carry propagation chain. This is the case for the Carry Lookahead [6] and Carry-Select [5] adders.

The philosophy behind the Carry Lookahead Adder (CLA) is to compute simultaneously several carries [6]. The CLA, just like other basic adders, follows the structure of the algebraic addition, performing the addition of the bit S_i as a function of the corresponding bits in the input operands (A_i and B_i) and the resulting carry of the previous stage (referred to as C_{i-1}).

The CLA differs from the other adders because the C_i, which is the carry bit used to compute the addition bit of the next stage (S_{i+1}), is not generated based on the addition bit S_i. Instead, two auxiliary signals are generated: G_i and P_i. The first one, called Generate, indicates the generation of a carry bit in the stage i (i.e., $C_i = 1$) and can be calculated by the following equation: $G_i = A_i \cdot B_i$. The signal P_i, called Propagate, indicates that

the carry coming from stage i-1 (C_{i-1}) is propagated to the stage i (i.e., $C_i = C_{i-1}$). This situation happens when one of the operands is equal to '1' and the other is equal to '0'. This way, C_i does not depend on C_{i-1}. The signal Propagate is calculated by the following equation: $P_i = A_i \oplus B_i$.

In the Carry-Select Adder (CSA) the addition is divided into a given number of sections that are computed in parallel [5]. This speeds up significantly the carry propagation, resulting in high operation speed. Each section performs two additions at the same time: one considering a '0' as carry-in and another considering a '1' as carry-in. This way, the carry chain propagation is broken to be accelerated. After all the addition sections finish their operation, for each section the correct addition value is selected based on the resulting carry-out of the previous section.

The drawback of the CSA relies on its high cost in terms of hardware resources. Resource overhead comes mainly from the need for using two adders in each section, in order to allow the parallel computation of the additions. The high cost of CSA may prevent its use in several applications where hardware resources are limited. However, its redundancy may be explored in order to derive fault-tolerant adder versions.

We developed in a previous work a new adder architecture based on the CSA which we called Re-computing the Inverse Carry-in Adder (RIC) [7]. This architecture reduces the hardware overhead created by the computation of two sections in parallel, replacing one of the two adders by a block that is cheaper than an adder. This way, in each section just one adder generates the result, assuming that the carry-in is equal to '0'. The new block receives this result R and converts it to R+1, eliminating the necessity of computing the addition when the carry-in is equal to '1'.

The RIC adder (Figure 1) uses a specific block called Re-computing Block (RB in the figure) to substitute one of the parallel adders presented in each section of the CSA. The remaining Ripple-Carry Adder (RCA) receives a carry-in equal to '0' and the RB re-computes the result of this RCA adder to generate a result with the carry-in equal to '1'. This re-computation is based on the exploration of the binary addition proprieties [10].

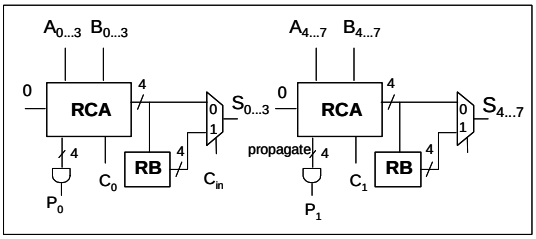

Fig. 1 – The 8-bit RIC adder is composed by two sections with one RCA and one Re-Computing Block.

3. SOFT ERROR PROTECTION TECHNIQUES

Among the existing soft error protection techniques, the Triple Modular Redundancy (TMR) is the simplest and most used one [9]. It also serves as reference, since its overhead in terms of hardware resources is known to be near 200% (for masked IC implementations). The high resource overhead comes from the fact that TMR requires the use of three exemplars of the block to be protected plus a voter circuitry. The outputs of these three blocks are connected to a voter that decides, by means of a majority election, which is the correct result. The voter is the solely block susceptible to error. If an error occurs in the voter, it may take a wrong decision.

In the Temporal Redundancy (TR) technique the output of the block to be protected is sampled in three different moments (clk, clk+d and clk+2d). The time difference between two consecutive resulting samples must guarantee the disappearance of the fault, if it occurs. To implement this technique it is necessary to triplicate the storage elements in order to store the three samples that are used as inputs to the majority voter. In this technique the hardware overhead is smaller than in the TMR case. However, the TR version is more complex because it requires different clocks for the three storage elements. Thus, this technique presents disadvantages concerning the time necessary to perform the whole computation, which is approximately equal to clk+2d+tp, where tp is the voter delay and d is the time required to perform each computation [9].

The Duplication With Comparison + Time Redundancy technique (DWC+TR) is an alternative to TMR. It explores time and hardware redundancy in an attempt to reduce the hardware overhead. Instead of using three instances of the block to be protected, as in the TMR case, it demands only two. The output of each instance is latched into two registers, being one triggered at instant clk and the other, at instant clk+d. Hence, four samples of the block output value are available, two per block instance. These four samples are fed to the error detection block, which is responsible for verifying if an error has occurred in one of the two instances. If that is the case, the error detector delivers the right value to be used as the third input of the majority voter. The voter also receives the output of each block instance. Its output is stored in a register at instant clk+2d. It is worth to notice that the amount of time "d", which is the difference between two clock edges, must be long enough to allow a transient pulse to propagate through the logic that is placed between registers. Otherwise, the probability of sampling an error becomes higher.

4. EXPERIMENTS AND SYNTHESIS RESULTS

In order to allow high performance and great integration capabilities, the most recent FPGA families, such as Altera's Stratix III, are fabricated with nanometer CMOS technology, increasing their susceptibility to transient faults. In FPGAs, transient faults can occur either in the configuration bits or in the user's programmable logic. In the latter case, there are two possible solutions: use hardened FPGAs or adopt ordinary FPGAs and apply fault-tolerance to the design

itself. Since hardened FPGAs are prohibitively expensive, the latter solution seems to be the most reasonable.

In this work we have described in VHDL unprotected and protected versions of adders for six different data lengths, ranging from 4 to 128 bits. To serve as a reference for the comparison, we have also described in VHDL protected and unprotected versions of the Ripple-Carry Adder (RCA) with the same data lengths. For each of the four adder architectures (RCA, CLA, CSA and RIC) three protected versions were designed: TMR, TR and DWC+TR.

All adder versions were synthesized for the EP3SE50F484C2 device (from Stratix III FPGA family) and validated through functional simulation with delay using Quartus II software from Altera (version 7.2 SP2) [11]. Table 1 shows the number of logic elements (ALUTs) reported by Quartus II for each adder version.

From the data of Table 1 it is possible to conclude that the protection with TMR results in resource overheads ranging from 161.3% to 211.1%, being 175% the average overhead. In theory, the increase of TMR should be higher than 200% for full custom (masked) implementations. However, the synthesis algorithms and the granularity of FPGA programmable elements may influence this overhead.

Tab. 1 – Number of used ALUTs for each adder version synthesized and simulated for Stratix III FPGAs.

ADDER	Technique	Number of used ALUTs					
		4 bits	8 bits	16 bits	32 bits	64 bits	128 bits
RCA	unprotected	33	62	120	236	468	932
	TMR	87	173	337	665	1321	2633
	TR	73	138	260	571	1123	2227
	DWC+TR	150	280	530	1030	2031	4288
CLA	unprotected	63	120	230	468	924	1916
	TMR	196	327	637	1259	2512	5089
	TR	129	231	433	838	1646	3343
	DWC+TR	237	436	826	1616	3178	6450
CSA	unprotected	65	142	280	556	1110	2215
	TMR	184	371	733	1465	2915	5813
	TR	105	200	384	823	1619	3220
	DWC+TR	182	402	776	1532	3090	6145
RIC	unprotected	55	118	232	460	918	1831
	TMR	160	323	637	1273	2530	5044
	TR	97	184	352	758	1490	2964
	DWC+TR	166	362	696	1370	2770	5504

Also it is possible to compute the resource overhead needed to protect adders with TR. It ranges from 37.1% up to 141.9%, with an average of 81.3%. However, a careful examination of resource overheads allows us to conclude that the impact of the extra registers (and voter) to protect with TR is higher for the RCA and for the CLA, with these overheads ranging from 116.7% to 141.9% and from 74.5% to 104.8%, respectively. This is because unprotected CSA and CLA adders require less ALUTs to be implemented than unprotected CSA or RIC adders. Therefore, the impact of the extra resources is not so prominent for CSA and RIC adders.

The resource overhead needed to protect the adders with DWC+TR ranges from 175.5% to 354.5%, with an average overhead of 245.1%. These results are in conflict with the original purpose of DWC+TR that is to reduce resource overhead, as claimed by some authors (see [9], for example). However, it is necessary to remark that our implementation of DWC+TR are quite conservative, in the sense that we use four registers, while other authors use only two.

Table 2 contains critical delay estimates for the architecture, obtained from Altera TimeQuest Timing Analyzer [11]. The data obtained show that the adders protected with DWC+TR present an intermediate performance. The increase in critical delay resulting from the use of DWC+TR ranges from 225.4% to 603.7%, with an average of 320.8%.

Tab. 2 – Critical Delay (ns) for each adder version synthesized and simulated for Stratix III FPGAs

ADDER	Technique	Critical Delay (ns)					
		4 bits	8 bits	16 bits	32 bits	64 bits	128 bits
RCA	unprotected	1.61	2.68	5.12	9.66	18.63	36.97
	TMR	2.40	4.01	6.89	12.38	23.29	46.17
	TR	10.25	17.10	25.67	47.12	95.86	181.62
	DWC+TR	11.34	13.69	21.02	37.56	74.61	140.03
CLA	unprotected	2.75	3.97	4.14	4.94	5.84	7.05
	TMR	4.4	5.38	5.73	6.52	7.65	8.97
	TR	13.22	20.42	22.05	26.11	29.03	32.08
	DWC+TR	12.61	16.01	17.25	20.19	24.02	27.29
CSA	unprotected	2.56	2.94	3.49	4.57	4.86	7.36
	TMR	3.36	3.88	4.72	5.37	6.23	8.94
	TR	11.47	17.21	20.77	21.71	26.55	31.8
	DWC+TR	10.31	12.39	15.22	14.86	20.07	26.03
RIC	unprotected	2.39	2.93	3.63	4.05	4.17	6.76
	TMR	3.62	5.5	4.53	4.93	6.36	8.03
	TR	11.28	14.56	16.01	19.69	26.06	29.48
	DWC+TR	11.54	11.83	12.79	14.97	18.88	27.28

The use of TMR resulted in the fastest protected adders, as one could expect. On the other hand, it is important to note that, considering adders of 32, 64 and 128 bits, the CLA TMR, CSA TMR and RIC TMR versions are faster than any other RCA version (including the non-protected ones). This is an important conclusion, since it means that adders protected with TMR can be faster than any RCA, including non-protected versions. The critical delay of TMR versions has exhibited increases ranging from 17.6% up to 59.9%. In the average, TMR results in an increase of 35.8% of the critical delay.

The use of TR presented an increase that ranges from 332.2% to 538%, with an average increase of 413.4%. These results were expected, since the implemented TR needs four clock cycles: three to sample the block output and one to vote the correct result.

It is important to note that among all 4 and 8-bit fast adders protected with TMR, the CSA presents the lowest critical delay. However, among all 16, 32 and 128-bit adders the RIC presents the lowest critical delay. It is also important to note that RIC adders protected with TMR require less ALUTs than CSAs protected with TMR and less ALUTs than some CLAs. Particularly, the 128-bit RIC adder protected with TMR is almost six times faster

than the 128-bit RCA protected with TMR. Such gain is obtained using less than twice the number of ALUTs that the equivalent RCA uses. This is a reasonable cost when requirements of performance and protection must be simultaneously met.

Table 3 presents power dissipation estimates obtained through the Altera PowerPlay Power Analyzer tool [11]. Although these results do not allow a comparison of the architectures themselves, one can note that the adders protected with TMR presented the lowest power increase (36.1% with respect to the non-protected versions). The adders protected with TR presented intermediate results and the average increase with respect to the non-protected adders was of 69%. Adders protected with DWC+TR have suffered an average increase of 130.1%, hence corresponding to the worst results.

Tab. 3 – Dynamic power dissipation (mW) for each adder version synthesized and simulated for Stratix III FPGAs.

ADDER	Technique	Dynamic Power Dissipation (mW)					
		4 bits	8 bits	16 bits	32 bits	64 bits	128 bits
RCA	unprotected	0.47	0.54	0.72	1.00	1.24	1.88
	TMR	0.52	0.72	0.65	1.08	1.90	2.77
	TR	0.78	1.15	1.75	2.01	2.26	3.02
	DWC+TR	1.03	1.58	0.19	2.44	4.15	4.66
CLA	unprotected	0.49	0.54	0.65	0.89	1.40	1.85
	TMR	0.55	0.67	0.91	1.45	1.99	3.27
	TR	0.57	0.83	0.95	1.17	2.24	2.96
	DWC+TR	1.00	1.20	1.41	1.94	3.06	4.26
CSA	unprotected	0.50	0.53	0.62	1.12	1.51	2.31
	TMR	0.61	0.68	0.86	1.21	2.21	4.06
	TR	0.66	1.02	1.51	2.05	2.47	3.45
	DWC+TR	0.73	0.19	2.08	2.72	3.87	5.45
RIC	unprotected	0.49	0.54	0.63	0.99	1.22	1.97
	TMR	0.55	0.74	0.81	1.16	2.09	3.55
	TR	0.58	0.19	1.64	2.00	2.13	3.33
	DWC+TR	0.65	1.44	2.14	2.85	3.82	5.02

The CLA architectures showed an average decrease of 5.87% in power dissipation when compared to the RCA versions. The RIC adder figured as an intermediate architecture, with an average increase of 5.25% and the CSA adder showed the worst results, with an average increase of 10.15%.

5. CONCLUSIONS AND FUTURE WORK

This work presented an analysis of three fast adder architectures protected against soft errors by using TMR, TR and DWC+TR techniques. Data obtained from Altera's Quartus II syntheses for Stratix III devices allowed the comparison between the techniques and also between the fast adder architectures.

The obtained results pointed out that for Stratix III the TR technique requires less extra hardware resource to implement protected adders, while the DWC+TR technique requires more resources than the other techniques. The critical delay results showed that the adders protected with TR present the worst performance and an average increase in critical delay of 413.4%. On the other hand, the adders protected with TMR showed an average increase of 35.8% in terms of critical delay, which is the lowest increase in critical delay among the three techniques.

Among all 4 and 8-bit fast adders protected with TMR, the CSA presents the shortest critical delay, while among 16, 32 and 128-bit adders, the RIC is the one which presents the shortest critical delays. Another important conclusion of this work is that RIC adders protected with TMR require less ALUTs than the CSA protected with TMR and even less ALUTs than some CLA adders protected with TMR. The power estimates showed that the high performance of RIC adders was not paid in terms of power dissipation, since these adders figured as intermediate solutions with an average increase of just 5.25% in relation to the RCA architectures.

As future work we intend to compare masked realizations of the same fast adder architectures, considering nanometer CMOS technologies.

REFERENCES

[1] R. Baumann, "Radiation-Induced Soft Errors in Advanced Semiconductor Technologies", IEEE Trans. on Devices and Materials Reliability, v.5, n.3, pp.305-316, Sep. 2005.

[2] C. Messenger, "Collection of Charge on Junction Nodes from Ion Tracks", IEEE Transactions on Nuclear Science, v. NS-29, pp. 2024-2031, Dec. 1982.

[3] P. Shivakumar et al., "Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic", International Conference On Dependable Systems And Networks, pp. 389 – 398, 2002.

[4] K. Hwang, "Computer Arithmetic: Principles, Architecture, and Design", New York: Wiley, 1979.

[5] O. J. Bedrij, "Carry-Select Adder", IRE Transactions on Electronic Computers, p. 340, 1962.

[6] A. Weinberger, and J. L. Smith, "A Logic for High-Speed Addition", National Bureau of Standards, Circulation 591, p. 3-12, 1958.

[7] E. Mesquita et al., "RIC Fast Adder and its SET Tolerant Implementation in FPGAs", 17th International Conference on Field Programmable Logic and Applications, Amsterdam, Netherlands (2007).

[8] C. Carmichael, "Triple Module Redundancy Design Techniques for Virtex FPGA", Xilinx Application Notes 197, San Jose, USA, 2001.

[9] F. Lima, et al, "Designing Fault Tolerant Systems into SRAM-based FPGAs", International Design Automation Conference, Proc. New York: ACM, pp. 650 – 655, 2003.

[10] K. Kumar, P. Lala, "On-line Detection of Faults in Carry-Select Adders", International Test Conference, pp. 912, 2003.

[11] Altera Corporation, <http://www.altera.com/>, accessed in March 2008.