ISSN NO. 2320-5407

Journal homepage: http://www.journalijar.com

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH

RESEARCH ARTICLE

VHDL IMPLEMENTATION OF FAULT TOLLERENT CONTROL SYSTEM

Prof. R.N.Nawkhare[1], Mr. Rohit N. Khodke[2]

1. Associate Professor, WCEM, Nagpur (MH), India

2. Student, Department of Electronics WCEM, Nagpur (MH), India

Manuscript Info

Manuscript History:

Received: 12 October 2015
Final Accepted: 25 November 2015
Published Online: December 2015

Key words:

Corresponding Author

Prof. R.N.Nawkhare

Abstract

A Kogge-Stone configuration can be used to apply a fault tolerant parallel-prefix adder due to inherited redundancy in the carry tree. Though thisdesign can be used to precise a fault in the carry-tree, noproviding was prepared for fault detectability. This paperintendsusing a sparse Kogge-Stone adder that isproficient of both fault correction and fault correctiondetectability.A number of smaller ripple carry adders are required to completethe sparse carry tree. Two extra ripple carryadders permit fault tolerance to be accomplished. TMR-RCA i.e. a triplemoderedundant ripple carry adder is usedas a point of reference. An FPGA platform carries out the Synthesis and simulation process. It is known that theTMR-RCA is the finestmethod for an FPGA faulttolerantimplementation because of the ease of theapproach and the usage of the fast-carry chains. However,the higher performance of the carry-tree above a ripplecarry adder in a Very Large Scale integrated circuit (VLSI) implementation creates thisproposed approach attractive for ASIC designs.

INTRODUCTION

Fault tolerant circuit designs are important for a lot of mission-critical applications, like as medical electronic and avionic, and for improved reliability for systems such as satellites that must operate in harsh and remote environments. Furthermore, a fault tolerant system is also important for circuits implemented in Nano scale technologies where the small device dimensions make the circuit susceptible to external interference, such as cosmic radiation. Thus, the ability of a circuit to detect and correct an error will be a main concern for future technologies. As the adder circuit often determines the critical path through several digital circuit systems, such as digital signal processors and processor data paths, the optimization of the adder design is the attention of ongoing research. For this reason, the emphasis of this paper will be on the application of fault tolerant techniques to high-speed adder designs. A fault tolerant circuit should be able to perform two tasks: (1) the detection of an error and (2) the correction of the error. An efficient fault tolerant design can perform this two stage method with minimal effect on the performance of the adder in terms of speed, power, and area. This paper will examine the implementation and execution of fault tolerance in adders that make use of a parallel-prefix scheme for quickcalculation of the carry signals. In particular, two designs based upon the Kogge Stone adder will be evaluatedfor implementation in Field Programmable Gate Arrays(FPGAs) and Application specific integrated circuits (ASICs). Thetypical method of Triple Mode Redundancy (TMR) applied to the Ripple Carry Adder (RCA) will be used as the reference design. Here, the paper the section is outlined

as follows. The background information and a review of the related research in this area is delivered by the second section. The next section discusses the design and implementation of the fault tolerant designs based on the Sparse Kogge-Stone adder(SKS). Some discussion on the specific strengths and weaknesses of each design will be

reviewed and evaluated. The final section delivers some conclusions and discussion of ongoing and future work in this area.

I I. BACKGROUND

Triple Mode Redundancy (TMR) is a common technique used to build fault tolerant designs using both ASICs and FPGAs. As the name denotes, the hardware is fundamentally replicated in triplicate by a voter circuit used to pass the majority rule signals to the output. An example of TMR using the Ripple Carry Adder (RCA), which forms our reference design, is shown in Fig. 1. The advantage of a Triple Mode Redundancy Ripple Carry (TMR-RC) is its easiness and effectiveness in both detecting and correcting errors and faults. The disadvantage is the increased area overhead with arelated tripling of the power dissipation.

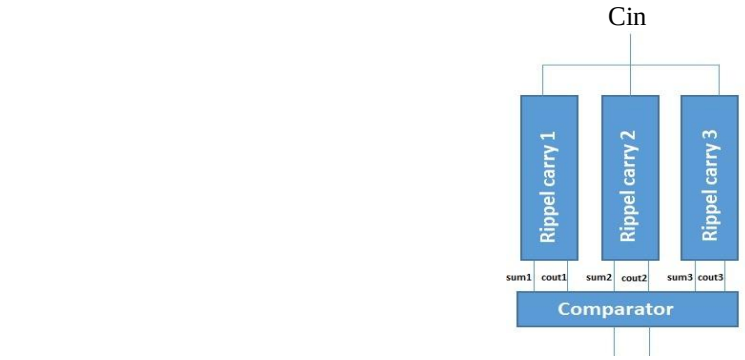

Fig.1. TMR adder circuit using Ripple carry adder as a reference design .i.e. (TMR-RC)

More sophisticated solutions can be reached by considering aParallel-prefix design, for instance a Kogge-Stone adder [3].There are two main benefits for this design. First, theCarry tree decreases the logic depth of the adder. These adders are recognised as parallel-prefix adders since the carries are basically generated in parallel. While the RCA hascritical path delay which growths linearly with the number of stages (i.e., it is of order *0(n)*, where *n* is the bit width of the adder), a parallel-prefix adder has a critical path which is O(log2«), for a radix-2 design. The differences in terms of logic depth and number of logic blocks can be realized by comparing Fig. 2, which explains a 16 bit Kogge StoneAdder. The carry block in Fig.2 consists of generate and propagate (GP) blocks, which are characterized as Black Cells (BC) and Gray Cells (GC) as defined in [4]. Basically, the parallel-prefix adder trades-off area and complexity for speed.

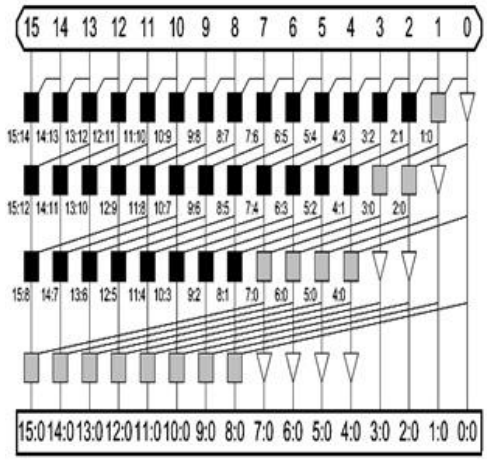

Fig.2. 16 bit Regular Kogge Stone Adder

The second benefit of the parallel-prefix design is dueto the inherent spatial redundancy in the carry-tree structure of the Kogge-Stone adder. As saw in [5], the even and odd carries generated in the adder are mutually exclusive as illustrated in Fig. 3.

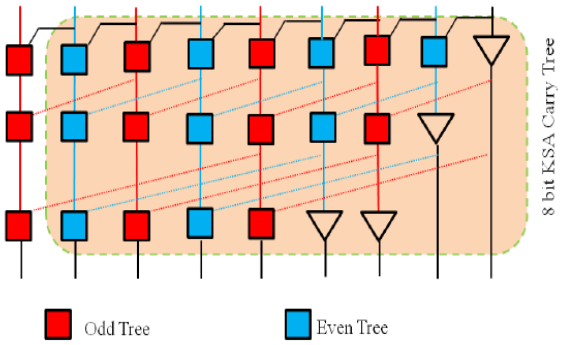

Fig. 3. An 8 bit Kogge-Stone carry tree showing the Mutually exclusive even and odd carry trees

Due to this mutual exclusivity, if a defect renders one half of the carry tree defective, the other half can be used to compute the carries for both the even and odd carries. This means that the adder will produce an output in two clock cycles instead of one. This trade-off in time may be satisfactory in some circumstances using appropriatearrangement and micro-architectural changes, as detailed in [5]. However, while this approach can correct for errors, no provision is given for detecting the errors.

I I I .PROPOSED DESIGN OF FAULT TOLERANT ADDER

Aninnovative design that utilizes a Sparse Kogge-Stone (SKS) adder which is both fault detecting and fault correcting is described in this section. As the name implies, an SKS adder features a reduction in the number of generate-propagate blocks in the carry-tree. Ripple carry adders are required at the output, whose length is dependent on the amount of sparseness in the carry tree. So as For example, Fig. 5 illustrates an SKS adder with a factor of four reduction in the carry-tree. In this case, four 4-bit ripple-carry adders are required to complete the summation process. We have previously demonstrated that the sparse and regular Kogge-Stone adders have essentially the identical delay when implemented on an FPGA while the former uses much less resources [6].
The additions needed to make the SKS adder fault tolerant are described in two steps. First, a fault tolerant design for the lower half (i.e., the ripple carry adder chains) is considered. Then, the design is extended to make the upper half (i.e., the carry tree) fault tolerant.

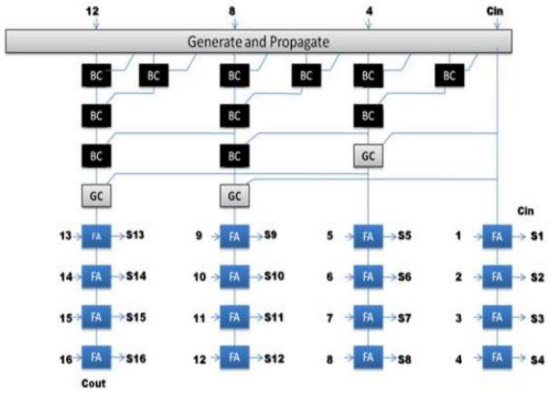

Fig. 4. A 16-bit Sparse Kogge-Stone (SKS) Adder

A. Lower Half partSparse Kogge Stone Fault Tolerant Design

Due to the adding of the ripple carry adders in the SKSDesign, analike testing methodology can be used as with TMR-RC to detect and correct errors found in any of the ripple carry adders. Some multiplexers and a bit counter are as well added to this design. An illustration of this design can be seen in Fig. 5 below, where the error correction and detection logic is highlighted in red.

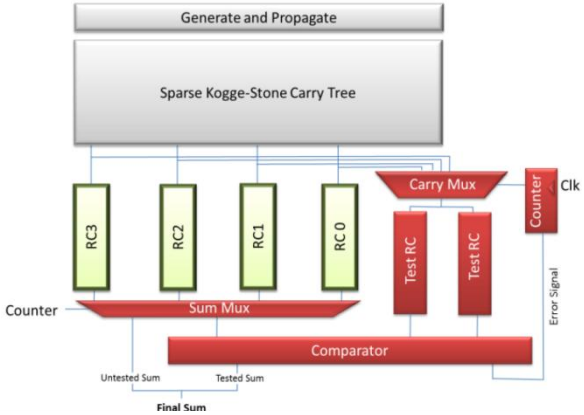

Fig. 5. Block diagram of Fault Tolerant Sparse Kogge-Stone Adder.

Similar to the TMR-RC, two additional ripple carry adders need to be added to the design for testing i.e. TestRC. During every clock cycle one of the four ripple carry adders (highlighted in green in Fig. 5) is selected for testing. The equivalent carry-in and A and B operands are routed through the Carry Mux to the TestRCs. The selection of these inputs is controlled by the counter, which is driven by a clock. Simultaneously, the output of the tested RCA branch (one of RCO to RC3) is switched to the comparator voter circuit for evaluation. On the following falling clock edge, after the valid output from the comparator has been passed through, the tested sum is latched and concatenated back with the untested sum resultant in the final sum. A timing diagram illustrates this process in Fig. 6.

When an error is detected, the clock is stopped which is operating the bit counter, at the RCA branch where the error was detected and will carry on tocorrecting the faulty RCA branches till the error is no longer detected. The final design was coded in VHDL and simulated using Xilinx ISIM. The simulation results shown in Fig. 8 showeffective detection and correction of an error in one of from the RCA chains.

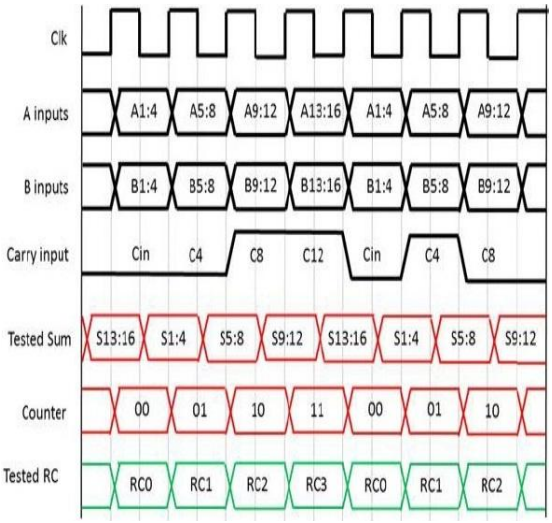

Fig. 6. Timing Diagram of the Fault Tolerant SKS Adder (lower half part).

The simulation results in Fig. 8 depict a possible worst case for the detection of the fault in the adder. This happens when the fault is introduced into the ripple carry branch immediately after the branch's fault testing took place. In the simulation, the input signal "fault" is used to insert a fault into the RCA i.e.one of the RCAs and the signal "error" rises to highonce a fault is detected. As can be seen from the graph, the adder takes three clock cycles, depicted as delay T, to recognize this error, allowing three incorrect results to pass through ("fffa" twice and "5550"). When the error is get known on the third clock cycle, then the signal driving the counter (labeled"controlout") is stopped and the correct solutions are formed on the falling edge of the clock ("ffff).

Synthesis results for TMR-RC, Fault Tolerant RKS, And our proposed Fault Tolerant Lower Half part SKS were obtain for SPARTEN 3E FPGA.Theseresults are depicted in fig.no.9 gives an estimation of number of look tables (LUT) Used by the designs. Fig. 9, give an estimation of the number of look up tables (LUT) used by each of the designs.

As can be seen from the Fig. 9, the Fault Tolerant RKS requires a lot more resources than the proposed FT-SKS design. This trade off comes at the expense of a higher logic depth resulting in a longer critical path in the FT-SKS than the Kogge Stone in the given design. The TMR is clearly the most efficient fault tolerant design for an FPGA implementation, in terms of resources, due to its simplicity and capability to take benefit of the fast-carry chain on an FPGA.

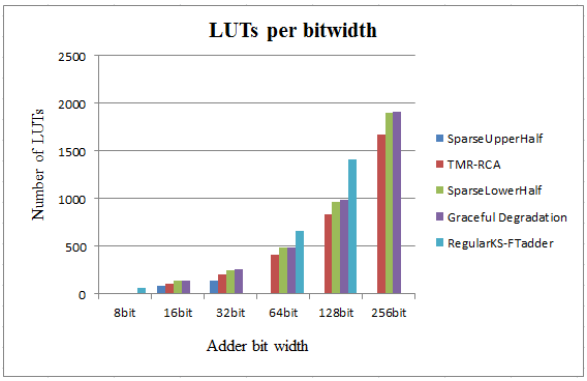

Fig. 7. Estimation of resources used from FPGA Synthesis

B. Upper Half part Fault Tolerant Design

The previous section only addresses the fault tolerance for the ripple carry portion of the adder (bottom half). For a design to be truly fault tolerant the carry tree section (upper half) needs to be considered. To achieve this, a testing methodology was developed that makes use of the redundant carries generated in the design by the carry tree (C/-C) and the ripple carry (C/-R). For example, for a 16-bit SKS with four 4-bit RCAs, there are two sets of carries generated for C4, C8, and C12. These two pairs must be equivalent for there not to be a fault within the circuit. When a mismatch is detected, the fault can be isolated to three sections of the adder as depicted by the green (1), purple (2) and blue (3) areas in Fig. 10 below.

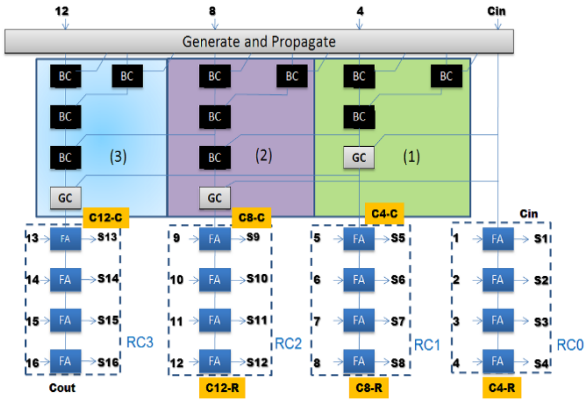

Fig. 8. Upper half error detection scheme for Sparse Kogge-Stone

For example, if the C4-C does not match C4-R, the error must definitely exist in the green (1) section assuming the first ripple carry adder (RCO) is fault free. This is likely the case due to the error detection and correction developed for the bottom half in the previous section. If this correction is evaluated as fault free, a fault free C4 will enter the second ripple carry adder (RC1). Next, if a mismatch is found in the C8-R and C8-C pair, the fault must exist in the purple (2) section. Finally if C8 and C4 are determined to be fault free, the error can be isolated to the blue section i f a mismatch is found in the C12 pair. A truth table describing these scenarios can be seen below in Fig. 9.

C12	C8	C4	Fault selection
X	X	1	Green (1)
X	1	0	Purple (2)
1	0	0	Blue (3)
0	0	0	No Error

X = don’t care
1 = Carry Mismatch
0 = Carry Match

Fig. 9. Upper half detection truth table.

With the error isolated to a section of the carry tree, the sections can be replaced with backups of the sections made available using multiplexers to reroute the carry tree from the faulty branches.

A method for increasing the number of sections detectable is also possible with the adaptation of a time redundant ripple carry adder made already available by the bottom half test circuit. By feeding the carry out produced by the first ripple carry (C4-RC0) into the test ripple carry adders (Test RCs), the adder can produce completely fault free carryvalues for comparison over the course of threeclock cycles.

These fault free carry's can then be comparedto the carry tree's output for a possibility of fault detection in five sections as depicted in fig. 10 below.

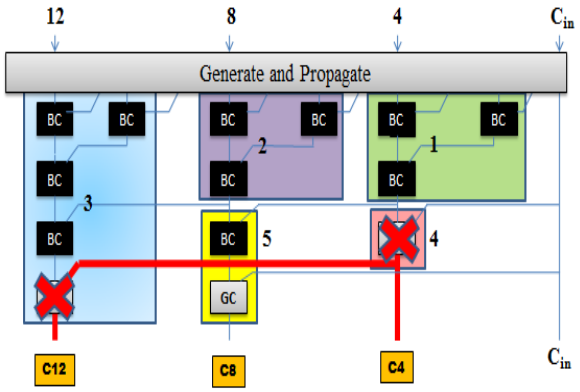

Fig. 10. Upper half error detection scheme with fault free Carry comparisons

An example of the increased resolution is also depicted in Fig. 10. The fault originates in the red section and passes through to C12 and C4. Because C8 showed no signs of a fault, the only possible case for this error combination can be traced back to a fault in the red section. Similarly, with this scheme, a list of possible combinations was developed into a truth table as shown in Fig. 11 with an example included of the time redundant ripple carry circuit capable of producing the fault free carries necessary for the comparison.

C12 Pair	C8 Pair	C4 Pair	Fault Section
match	match	match	No Error
match	match	mismatch	Not Possible
match	mismatch	match	YELLOW (5)
match	mismatch	mismatch	Not Possible
mismatch	match	match	BLUE (3)
mismatch	match	mismatch	RED (4)
mismatch	mismatch	match	PURPLE (2)
mismatch	mismatch	mismatch	GREEN (1)

1 = Error 0 = No Error

Fig. 11. Truth table for upper half error detection with fault free comparisons

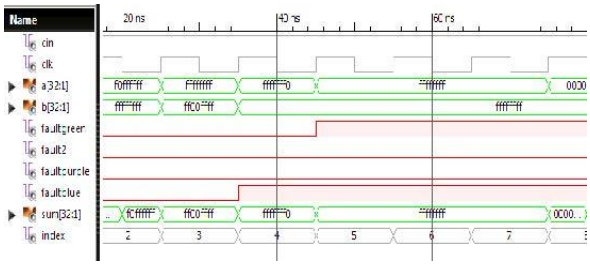

Fig. 12. Simulation of upper half fault detection and correction

Once the error has been successfully isolated to a specificsection of the adder the faulty section can be replaced using a decoder and multiplexers, which reroutes the signals to replacement sections built into the design. As a proof of concept, a 16 bit version of this fault tolerant adder was coded in VHDL and simulated with ISIM. As depicted in Fig. 12, the fault can only be corrected after a faulty sum value has been allowed to pass. Once the

rerouting of the replacement section has completed, the adder will perform normally as can be seen from the simulation results.

IV. DISCUSSION

The SKS adder can be made fully fault tolerant provided that the accompanying tradeoffs in terms of time and hardware complexity are acceptable. For an FPGA design, the TMR-RCA is still the best option in terms of speed and hardware complexity due to its straightforward implementation and its capability to take benefit of the fastcarry chain, which favours a linear adder design over a parallel-prefix adder. However, when ported to an ASIC implementation, the parallel-prefix adder becomes more favourable in terms of speed due to the 0(log2«) delay through the carry path compared to O(n) for the RCA. The RSK fault tolerant design proposed by [5] takes advantage of the inherent spatial redundancy in the carry tree to produce a fault correcting design. However, no provision is given to detect errors. This paper has shown that the SKS adder provides a suitable middle ground - a reduction in the complexity of the carry-tree via the addition of a set of smaller RCA chains while allows varying degrees of fault detectability to be included in the design. Errors in the RCA branches can be detected by adding two additional RCAs. At the cost of some additional hardware overhead and clock cycles, errors in the propagate-generate blocks in the carry tree can also be detected. Finally, it should be noted that the additional RCA's for fault checking can be used to implement a graceful degradation scheme. In this scenario, a faulty RCA branch can be permanently replaced with one of the fault checking adders. This removes the possibility of continual fault checking of the adder branches, but does allow the adder to remain fully functional until a more permanent fix can be implemented.

V. SUMMARY AND FUTURE WORK

While the standard TMR-RCA is found to be superior for FPGA designs, the short critical path through the carry chain makes the fault tolerant parallel-prefix adders more attractive in a VLSI implementation. For very large bit widths, there are signs that the Kogge-Stone adder dealshigher performance over an RCA when executed on an FPGA [6]. The sparse Kogge-Stone designs described in this paper allow fully fault tolerant performance as well as the possibility of graceful degradation. Simulation and synthesis using FPGA design tools have validated the performance of the lower-half and upper-half SKS for fault detection and correction. It should be noted however that although the upper half concept was proven in this paper, a method of easily scaling to wider bit widths needs further investigation. Partitioning the sections of the adder without a visual representation available during the implementation is quite a challenging task. Future work includes the development of automated techniques for implementing the fully fault tolerant sparse Kogge-Stone adder.

REFERENCES

[1] F. Kastensmidt, L. Cairo, and R. Reis, *Fault-Tolerant Techniques for SRAM-Based FPGAs,* Springer, 2006.

[2] L. Sterpone and M. Violante, "Analysis of the Robustness of the TMR Architecture in SRAM-Based FPGAs," *IEEETrans. Nucl. Set,* vol. 52, no. 5, pp. 1545-1549, Oct. 2005.

[3] P. Ivi. Stone and H. S. Stone, "A Parallel Algorithm for theEfficient Solution of a General Class of Recurrence Equations," *IEEE Trans, on Computers,* vol. C-22, no. 8, pp.786-793, Aug. 1973.

[4] N. H. E. Weste and D. Harris, *CMOS VLSI Design,* 4,h
Edition, Pearson-Addison-Wesley, 2011.

[5] S. Ghosh, P. Ndai, and K. Roy, "A Novel Low OverheadFault Tolerant Kogge-Stone Adder using Adaptive Clocking," *Design, Automation and Test in Europe,* pp. 366-371,2008.

[6] D. H. K. Hoe, C. Martinez, and J. Vundavalli, "Design andCharacterization of Parallel Prefix Adders using FPGAs,"*IEEE 43rd Southeastern Symposium on System Theory,* pp.170-174, March 2011.