International Journal of Emerging Engineering Research and Technology
Volume 3, Issue 7, July 2015, PP 41-47
ISSN 2349-4395 (Print) & ISSN 2349-4409 (Online)

Fault Tolerant Architecture Design for Digital Adder

M.Santhosha, M.E.

Dept. of Electronics and Communication Engg., M.V.S.R Engineering College, Hyderabad.

B. Sarala

Associate Professor, Dept. of Electronics and Communication Engg., M.V.S.R Engineering College, Hyderabad.

ABSTRACT

Fault tolerant architecture is used to increase the reliability and decrease the fault rate. Fault tolerant in arithmetic operations mainly deal with addition, subtraction, multiplication division. In this paper mainly focused with adders. These adders are half adder, full adder, ripple carry adder, carry look ahead adder, conditional sum adder. Kogge stone adder is a high speed adder. In this paper implemented a single fault tolerant circuit for the Kogge stone adder. Fault tolerant architectures are mainly used in mission critical applications like satellites, defense systems and safety measures etc.

Keywords: fault tolerant, Kogge stone adder.

INTRODUCTION

Fault-tolerant computing is the art and science of building computing systems that continue to operate satisfactorily in the presence of faults. A fault-tolerant system may be able to tolerate one or more fault-types including transient, intermittent or permanent hardware faults, software and hardware design errors, operator errors, or externally induced upsets or physical damage. An extensive methodology has been developed in this field over the past thirty years, and a number of fault-tolerant machines have been developed. It deals with random hardware faults, while a smaller number deal with software, design and operator faults [1]. A large amount of supporting research has been reported for fault tolerance and dependable systems. In this research work covers a wide spectrum of applications ranging across embedded real-time, commercial transaction, transportation, and military and space systems. The supporting research includes system architecture, design techniques, coding theory, testing, validation, and proof of correctness, modeling, software reliability, operating systems, parallel processing, and real-time processing. These areas often involve widely diverse core expertise ranging from formal logic, mathematics of stochastic modeling; graph theory, hardware design and software engineering. The majority of fault-tolerant designs have been directed toward building computers that automatically recover from random faults occurring in hardware components. The techniques employed to do this generally involve partitioning a computing system into modules that act as fault-containment regions. Each module is backed up with protective redundancy so that, if the module fails, others can assume its function. Special mechanisms are added to detect errors and implement fault recovery. In general, two approaches to hardware fault recovery have been used:

1) Fault masking 2) Dynamic recovery.

Fault masking is a structural redundancy technique that completely masks faults within a set of redundant modules. A number of identical modules execute the same functions, and their outputs are voted to remove errors created by a faulty module. Triple modular redundancy (TMR) is a commonly used form of fault masking in which the circuitry is triplicated and voted. TMR involves creating three redundant copies of a circuit and adding majority voters to select the correct circuit output from the three copies. With this mitigation methodology, a single module failure will not cause an error in the circuit output, since the other two modules continue to operate correctly and will overrule the faulty module [2]. A TMR system fails whenever two modules in a redundant triplet create errors so that the

Address for correspondence:

msaug12@gmail.com

vote is no longer valid. TMR process area overhead is very high because in this the full system is triplicated and an extra voter is included.

Dynamic recovery involves automated self-repair where the system consists of some identical spare modules along with the active modules and a fault detection and reconfiguration unit, that is assumed to be capable of detecting any erroneous output produced by the active module, disconnecting the faulty active module, and connecting instead a fault-free spare. The approach is generally more hardware-efficient than voted systems. Instead of finding and replacing the whole faulty module, make each module self- reconfigurable itself. This decreases hardware cost as well as increases the reliability of the system.

BUILT IN SELF TEST (BIST)

BIST stands for Built-In Self-Test, approaches the problem from a different angle. BIST attempts to move as much of the tester functionality as possible onto the silicon. Embedding the tester equipment functionality into the semiconductor product not only reduces the burden on and the complexity of external test equipment, but the on-chip access is simplistic and faster. BIST is extra circuitry to enable complete testing without much external help. It basically consists of a test pattern generator, test response compactor, test response comparator and control circuitry to carry out the test process and give the result signal. BIST is not a replacement for scan and does not result in fewer test patterns [3]. BIST enabled designs still require the use of testers for certain portions of the design that BIST cannot be inserted. BIST is the circuitry that enables a chip to test itself, and it has been consider as a good substitute for the ATPG (Automatic Test Pattern Generation) procedure which deterministically uses some algorithms to produce the function vectors and their corresponding responses required for testers. The main advantage of using BIST are: (i) eliminating (or at least minimizing) the costs of ATPG and fault simulations, (ii) shortening the time duration of tests (by running tests at circuit speeds), (iii) simplifying the external test equipment, and (iv) easily adopting to engineering changes (in other words, low technology dependency). A simple BIST configuration, as shown in fig.1, includes an LFSR (Linear Feedback Shift Register), and MISR (Multiple Input Signature Register), and some control circuitry. The LFSR basically produce pseudorandom input vectors, so it's also called a PRPG (Pseudo Random Pattern Generator); the MISR (another type of LFSR) com-presses the response of the CUT (Circuit under Test) into a signature to be compared to a good circuit's response [4-5]. The BIST results, "pass" or "fail", depend on whether the signature captured into the MISR matches the one coming out of the good circuit.

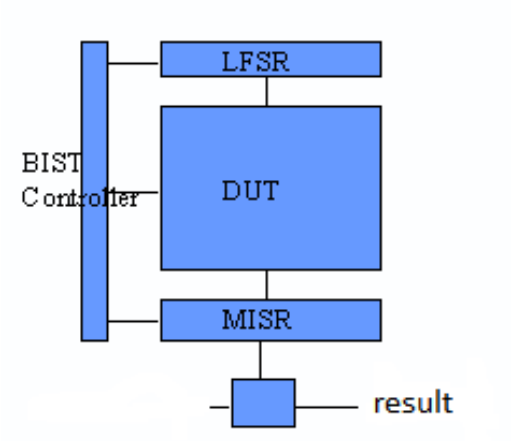

Fig1. *Simple BIST configuration*

PARALLEL-PREFIX ADDER

Parallel-prefix structures are found to be common in high performance adders because of the delay is logarithmically proportional to the adder width [6].

PPA's basically consists of 3 stages.

- Pre computation
- Prefix stage
- Final computation

Pre Computation-

In this stage, propagates and generates are computed for the given inputs.

Prefix Stage-

In this stage, group generate/propagate signals are computed at each bit. The black cell (BC) generates the ordered pair, the gray cell (GC) generates only left signal.

Final Computation-

In this stage, the sum and carryout and are the final output.

The basic structure of the parallel prefix adder is shown fig.2.

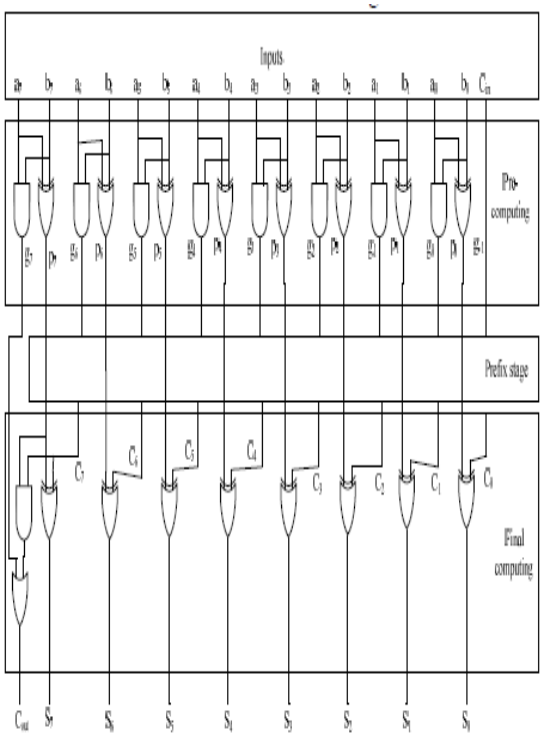

Fig2. *Paralle prefix structure*

The block diagram and logic definition of the black cell is shown in the following fig.3.

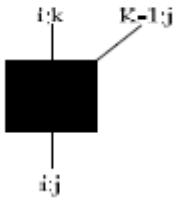

Fig3(a). *Black cell*

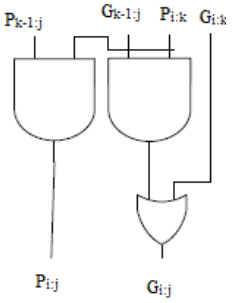

Fig3(b). *Logic definition of Black cell*

The block diagram and logic definition of gray cell is shown in the following fig.4.

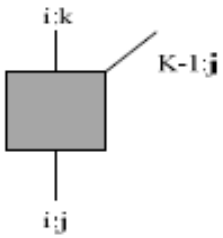

Fig4(a). *Gray cell*

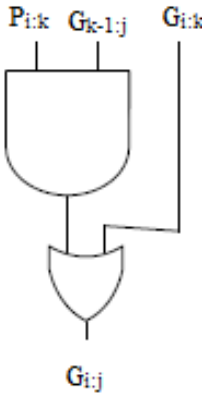

Fig4(b). *Logic definition of Gray cell*

KOGGE STONE ADDER

Kogge Stone Adder (KSA) is a parallel prefix form of carry look ahead adder. Kogge stone adder can be represented as a parallel prefix graph consisting of carry operator nodes. The required to generate carry signals in this prefix adder is O (log n). It is the fastest adder with focus on design time and is the common choice for high performance adders in industry. The better performance of Kogge stone adder is because of its minimum logic depth and bounded fan-out. It is the common design for high performance in industry. Kogge Stone Adder is that use the fewest logic levels. The 16 bit Kogge stone adder uses black cells (BC) and gray cells (GC) and it won't use full adders. The 16 bit KSA uses 36 BC's and 15 GC's. And this adder totally operates on generate and propagate blocks. So the delay is less when compared to the Sparse Kogge stone adder (SKA) and spanning tree adder (STA). The 16 bit KSA is shown in fig.5. In this Kogge stone adder, there are no full adder blocks like sparse Kogge stone adder and spanning tree adder [7].

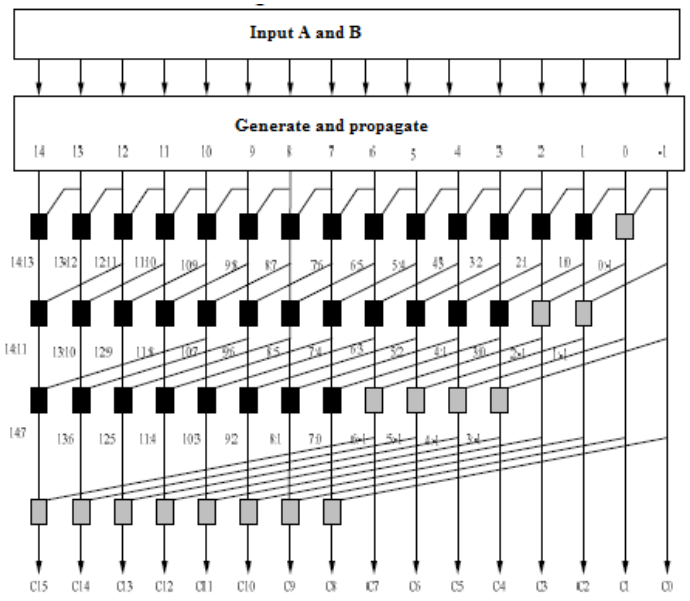

Fig5. *Simple 16- bit Kogge stone adder*

TPG:

First objective is to make the design C- testable so that size of the test vector becomes independent of the size of Kogge stone adder. For that need to identify the identical sub-modules within KSA block. Group the 8-bits and make the KS0 module. Similarly make the two more modules such as KS1 and KS2. Third module KS2 is the redundant module is used as the spare. For testing of Kogge stone adder, each module has to be tested [8]. A 4–bit counter and few a combinational gates required to generate the test vectors shows the test pattern generation (TPG) circuits for KS0 module. The circuit for the test pattern generation for KS0 module is shown in fig.6 (a). Similarly test pattern generation circuits for remaining modules shown in fig.6 (b), (c).

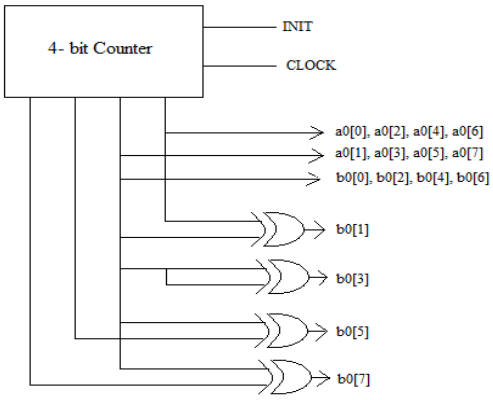

Fig6(a). *TPG for KS0*

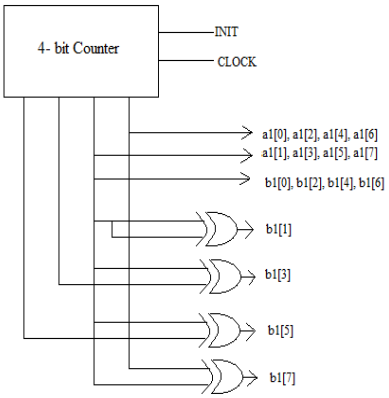

Fig6(b). *TPG for KS1*

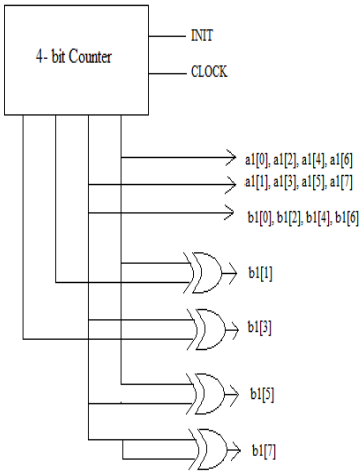

Fig6(c). *TPG for KS2*

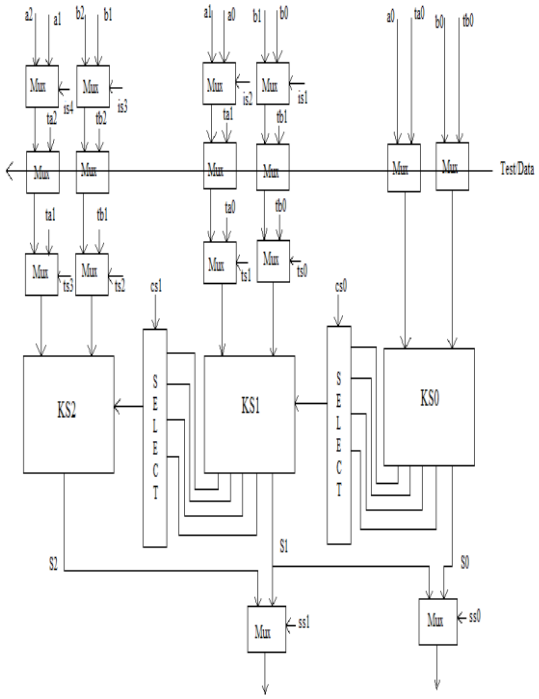

Fig7. *Single fault tolerant kogge stone adder*

The test/data lines are fed to the multiplexes (mux). If the test signal is 1 the test pattern values are fed to the identical sub modules otherwise normal input values are fed to the modules. The functionality of them is checked and compared with the desired ones generated by simple combinational circuitry. One spare identical module is used with the two operating identical modules and if any one of the modules is found to be fault in testing phase, the functionality of that is bypassed by the spare one using multiplexers (Mux) at input and output levels. The complete single fault tolerant architecture for 16-bit kogge stone adder is shown in fig.7.

SIMULATION WAVEFORMS

The simulation of single fault tolerant architecture for 16-bit kogge stone adder is done using Xilinx 13.2 ISE design suite. The output wave form is show in fig.8.

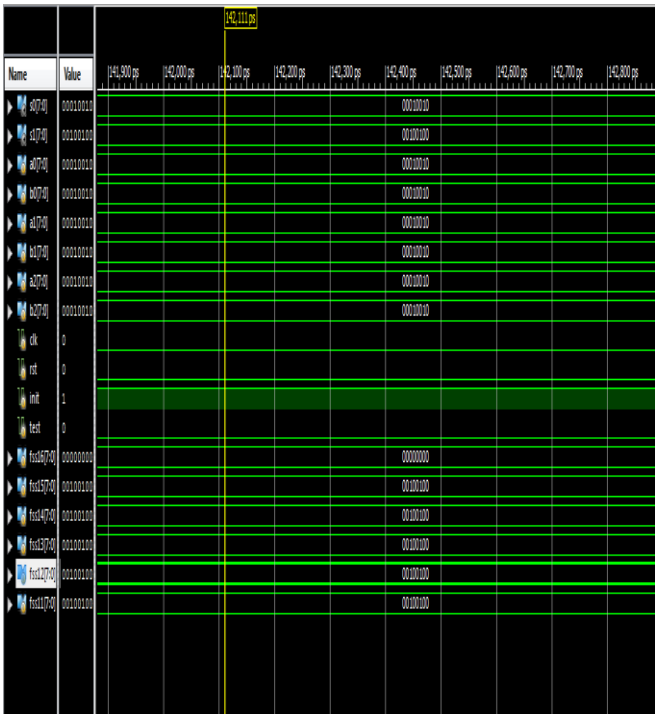

Fig8. *Simulation of single fault tolerant kogge stone adder*

CONCLUSION

The fault tolerant architecture design is presented for the 16-bit Kogge stone adder for single fault. The proposed fault tolerant adder design is implemented using Verilog and synthesized by using Xilinx. The same approach can be applied to design fault tolerant for many types of fast adders and digital blocks and these designs can be used in different digital architectures to increase the reliability of the whole system at the cost of hardware complexity.

REFERENCES

[1] J. A. Rivers and P. Kudva, "Reliability Challenges and System Performance at the Architecture Level," IEEE Design and Test of Computers, vol. 6, p.no. 62-73, November, 2009.

[2] Jaimini Patel and Prof.Deepali H.Shah, " A Realiasation of fault tolerant ALU with TMR method," IJEDR, vol.2, p.no.161-164, August 2011.

[3] B. Koenemam and S. Pateras, "Built in Self-Test (BIST) in the era of deep submicron technology," IEEE, p.no. 312-315, November, 1996.

[4] K. Gunavathi, K. Paramasivam and P. Subashini Lavanya "A novel BIST TPG for testing of VLSI circuits," IEEE, p.no. 109-114, August 2006.

[5] M.Hamad, "Modelling and analysis of delay faults in VLSI circuits" IEEE, vol.2, p.no.587-590, December, 2003.

[6] Vidya Vijayan and M. Mohana Priya "Area, delay and power," International Journal of VLSI design and communication systems (VLSICS), vol.3, p.no.153-157, February 2012.

[7] Y. Sudheer kumar and B. Rajendra Naik "Design and Estimation of delay power and area for parallel prefix adders," IEEE Transaction vol.1, p.no. 978-986, August, 2014.

[8] A. Mukherjee and A. S. Dhar, "Design of a Self-Reconfigurable Adder for Fault-Tolerant VLSI Architecture," 2012 International Symposium on Electronic System Design (ISED), p.no. 92-96, December 2012.