International Journal of Emerging Engineering Research and Technology
Volume 3, Issue 5, May 2015, PP 62-67
ISSN 2349-4395 (Print) & ISSN 2349-4409 (Online)

Triple-Fault Tolerant Architecture Design for Ripple Carry Adder

[1]Shaik Mabjani, [2]K. Subramanyam

[1]Electronics and Communication Engineering, Audishankara College of Engineering and Technology, Gudur, Nellore, India
[2]Asst Professor Electronics and Communication Engineering, Audishankara College of Engineering and Technology, Gudur, Nellore, India

ABSTRACT

A system must be fault tolerant to decrease the failure rate and increase the reliability of it. Multiple faults can affect a system simultaneously and there is a trade-off between area overhead and number of faults tolerated. This paper presents fault tolerant architecture design for a ripple carry adder and a conditional sum adder as fast adder assuming single double and triple faults. The philosophy can be generalized for any other system which has structural regularity within it.

Keywords: Fault tolerant, Ripple carry adder, Test pattern generation

INTRODUCTION

Fault tolerance is the property that enables a system to continue operating properly in the event of the failure of (or one or more faults within) some of its components. If its operating quality decreases at all, the decrease is proportional to the severity of the failure, as compared to a naively designed system in which even a small failure can cause total breakdown. Fault tolerance is particularly sought after in high-availability or systems. A fault-tolerant design enables a system to continue its intended operation, possibly at a reduced level, rather than failing completely, when some part of the system fails. The term is most commonly used to describe computer systems designed to continue more or less fully operational with, perhaps, a reduction in throughput or an increase in response in the event of some partial failure. That is, the system as a whole is not stopped due to problems either in the hardware or the software. An example in another field is a motor vehicle designed so it will continue to be drivable if one of the tires is punctured. A structure is able to retain its integrity in the presence of damage due to causes such as fatigue, corrosion, manufacturing flaws, or impact. Within the scope of an individual system, fault tolerance can be achieved by anticipating exceptional conditions and building the system to cope with them, and, in general, aiming for self-stabilization so that the system converges towards an error-free state. However, if the consequences of a system failure are catastrophic, or the cost of making it sufficiently reliable is very high, a better solution may be to use some form of duplication. In any case, if the consequence of a system failure is so catastrophic, the system must be able to use reversion to fall back to a safe mode. This is similar to roll-back recovery but can be a human action if humans are present in the loop.

Fault tolerance is defined as the ability to continue operating after the failure of a given system component. To be fault tolerant, a system must have one or more redundant components that can take over the function when the primary component fails. In addition, the system must have both a means of Detecting failures in the components and a means of transferring to working components after a failure has been detected. Fault-tolerant system configurations are used extensively in processes where the system must remain on-line in the event of component failure. Although applications are widespread, industrial processes, aerospace vehicles, and ground transportation are especially noteworthy. The need for optimization in the design of these systems is apparent when one thinks of the complexity of modern spacecraft. The high costs associated with their fabrication and launch dictate that any design proposal be assured a very high probability of success at the lowest possible system cost.

Address for correspondence

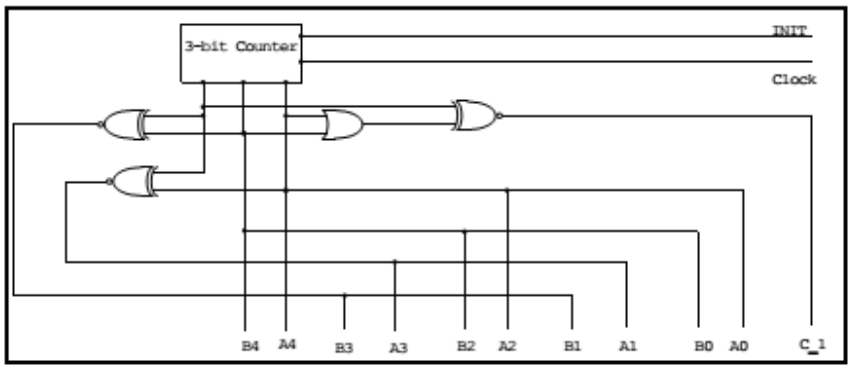
mabjanijani45@gmail.com

Reliability and availability have become increasingly important in today's computer dependent world. In many applications where computers are used, outages or malfunction can be expensive, or even disastrous. Just imagine the computer system in a nuclear plant malfunctioning or the computer systems in a space shuttle booting just when the shuttle is about to land... These are the more exotic examples. More close to everyday life are the telecommunications switching systems and the bank transaction systems. To achieve the needed reliability and availability, we need fault-tolerant computers. They have the ability to tolerate faults by detecting failures, and isolate defect modules so that the rest of the system can operate correctly. Reliability techniques have also become of increasing interest to general-purpose computer systems. Four trends contribute to this:

The first is that computers now have to operate in harsher environments. Earlier, computers operated in clean computer rooms, with stable climate and clean air. Now the computers have moved out to industrial environments, with temperatures over a wide range, dust, humidity and unstable power supply. All these factors alone could make a computer fail. Second, the users have changed. Earlier, computer operators were trained personnel. Now, with an increasing number of users, the typical user knows less about proper operation of the system. The consequence is that computers have to be able to tolerate more. Third, the service costs increases relative to hardware costs. Earlier the average machine was a very expensive, big monster. At that time, it was common with one or several dedicated operators to keep the system up and running. Today, a computer is cheap, and the user has the job of being the "operator". The user can not afford frequent calls for field service. The fourth and last trend is larger systems. As systems become larger, there are more components that can fail. This means, to keep the reliability at an acceptable level, designs have to tolerate faults resulting from component failures. So, what can cause outages of equipment, making fault-tolerance techniques necessary? We can split them into outages caused by:

- *Environment:* This is facilities failures, e.g. dust, fire in the machine room, problems with the cooling, Earthquakes or sabotage.
- *Operations:* Procedures and activities of normal system administration, system configuration and system operation. This can be installation of a new operating system (requires booting of the machine), or installation of new application programs (which requires exit and restart of programs in use).
- *Maintenance:* This does not include software maintenance, but could be hardware upgrading.
- *Hardware:* Hardware device faults.
- *Software:* Faults in the software.
- *Process:* Outages due to something else, e.g. a strike.

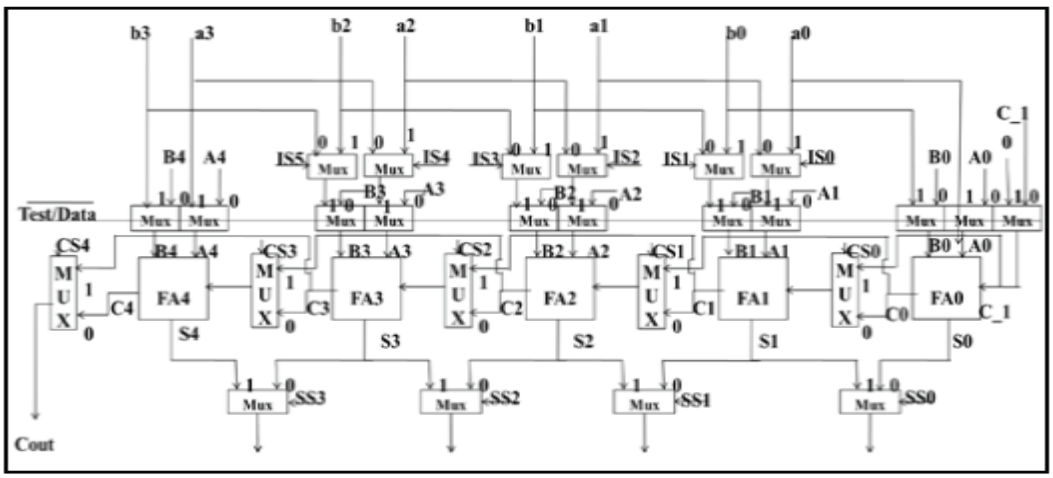
Interesting to note is that, contrary to common assumptions, few outages are caused by hardware faults. In a modern system, fault-tolerance masks most hardware faults, and the percentage of outages caused by hardware faults are decreasing. On the other side, outages caused by software faults are increasing. According to a study on Tandem systems [4], the percentage of outages caused by hardware faults was30% in 1985, but had decreased to 10% in 1989. Outages caused by software faults increased in the same period, from 43% to over 60%!



Test Pattern Generation Circuit

SINGLE FAULT MODEL

Adder is absolutely essential block in any digital architecture. Among different types of adders ripple carry adder (RCA) is most popular in different type of computing machines because it is simple in structure and easy to implement. It has also high throughput for bit level pipelining. We have taken a 4-bit RCA to make itself reconfigurable. We will also discuss how the same approach can be applied to any system, which can be divided into some identical modules, to make the system fault tolerant. The fault tolerant designs are also cascadable to increase the number of input bits Making a system module wise self-reconfigurable is more cost effective and hardware efficient rather than trying to make the whole system fault tolerant at a time. We will also elaborate how the approach to design a self reconfigurable 4-bit RCA can be applied to design any fault tolerant module. He assumed that all possible cell inputs must be applied to each cell in order to test it completely and that a fault in a cell may affect the cell outputs in any arbitrary manner. These are known as general fault assumptions. In general, the number of tests required to test a one-dimensional ILA of p cells is dependent on p and is at most pmn (n 1), where m is the number of columns and n is the number of rows of the cell flow table. Required test patterns are independent of the size of the ILA. Cheng and Patel have shown that a RCA composed of several full adder (FA) cells can be completely tested by a minimum test set of size eight independent of the number of cells in the RCA under single faulty cell assumption. But there is no provision to detect and eliminate the error. Their work cannot take care of bridging faults. We have modified the circuitry to make each intermediate FA controllable and observable to improve the testability. Our design can tolerate bridging faults also until it affects two FAs at a time for the single fault tolerant design. The size of eight is minimum since eight test vectors are necessary to exhaustively test just one cell. This concept can also be generalized to design any x-bit self reconfigurable Fault Tolerant Adder (x = any positive integer). A simple four bit Adder module is shown in figure 1 with its all data inputs and outputs. For checking stuck-at-faults at any of the input or output of figure 1, only six test patterns suffice. A Simple 4-bit Adder But instead of error at any of the input or output lines, one of the FAs may be faulty itself; i.e., there may be some stuck open or bridging faults or so inside the FA or some transistors in that FA may be faulty. The test Patterns for stuck-at-faults only do not consider these cases. To check for such cases, each FA should be tested exhaustively. Table I shows that only eight test patterns suffice this requirement, i.e. all 4 FAs are tested exhaustively to check for any fault in it. Even the same pattern can be replicated to test any x-bit adder (x = any positive integer) and only eight test patterns will suffice for all such cases. So, to make the Adder structure fault tolerant, we shall test each FA with all possible input patterns and check whether we get the desired output or not. If not, the Adder block must be faulty! Now, instead of replacing the faulty 4 bit adder completely, we will use the Dynamic Recovery concept to make the adder module fault tolerant itself. Here we will use one spare FA and if any of the working FA is found to be faulty, then the spare one will start working bypassing the faulty one so that the system can work properly.



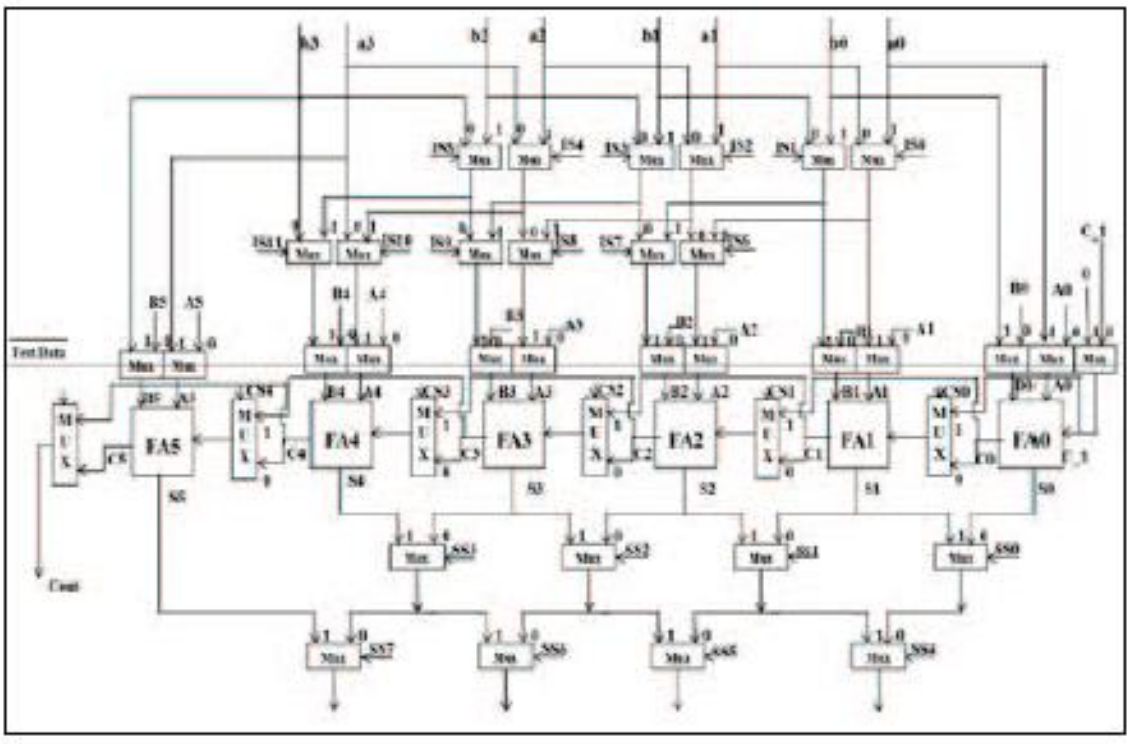
4 Bit Fault Tolerant Adder

One redundant FA is used along with the four operating FAs to incorporate dynamic recovery feature in the system. If any one of the first four FAs is found to be faulty or its input or output is at

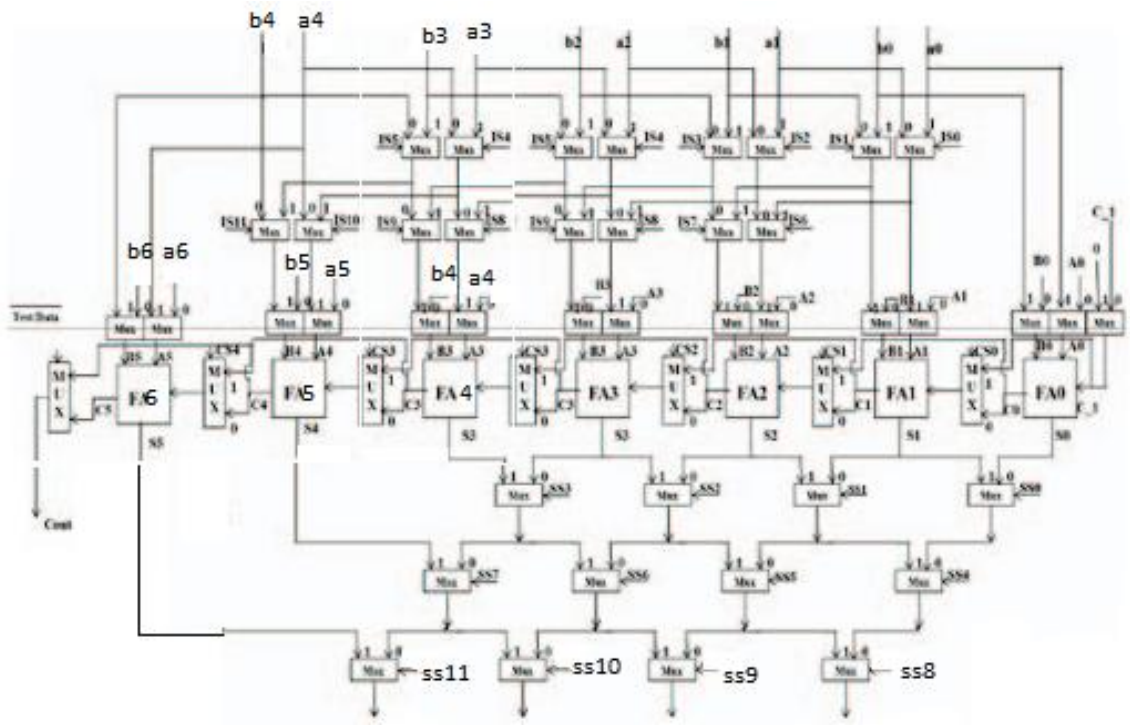
some faults, then the system will bypass the inputs and outputs of the corresponding faulty FA to the next FA and the spare one will be activated automatically. To use same carry input to all the FAs, we select the multiplexers (mux) select input CSi as '1'. But if carry input to this FA is at some stuck-at-fault, it will be propagated to the carry input of the next FA giving an overall erroneous output! It now seems that a large number of test vectors are required to test each FA exhaustively as we cannot control the carry inputs of the intermediate FAs due to their direct dependence on output carry of the immediately preceding FAs. But we designed the truth table (Table I), where it shows that only eight patterns still suffice to test each FA exhaustively. Generated carry output at any stage works as carry input to the next stage. The truth table shows test input pattern for five FAs. Even the same pattern can be replicated to test any x-bit adder (x = any positive integer) and hence only eight test patterns will suffice for testing the x-bit adder.

DOUBLE FAULT MODEL

It is a very general case where a faulty module affects its neighbors also. Again different reliability issues such as electro migration, hot-carrier degradation, oxide breakdown etc. affects a region in the system rather than affecting one single transistor. Sometimes faults near the different boundary regions are also very common causing more than one fault in same system. We will modify the previous design for fault tolerant RCA so that it can incorporate more than one fault such that at most two FAs are affected. Now this design can also tolerate bridging faults affecting two FAs maximally. As this new design can handle with two faulty FAs mutinously, there must be two spare FAs along with original number of active FAs. Number of MUX selection levels also increases. So reliability of the system improves with the added hardware complexity. The minimum test size of arbitrary ripple carry adders under multiple cell fault model is 11 [5]. But this procedure can't detect the position of the fault so that the faulty part can be taken care of! So to make the system self-reconfigurable, even if multiple faults occur, we modify our previous design as shown in figure 9. This design can handle errors which affect at most two FAs in the system



A complete system tolerating double faults

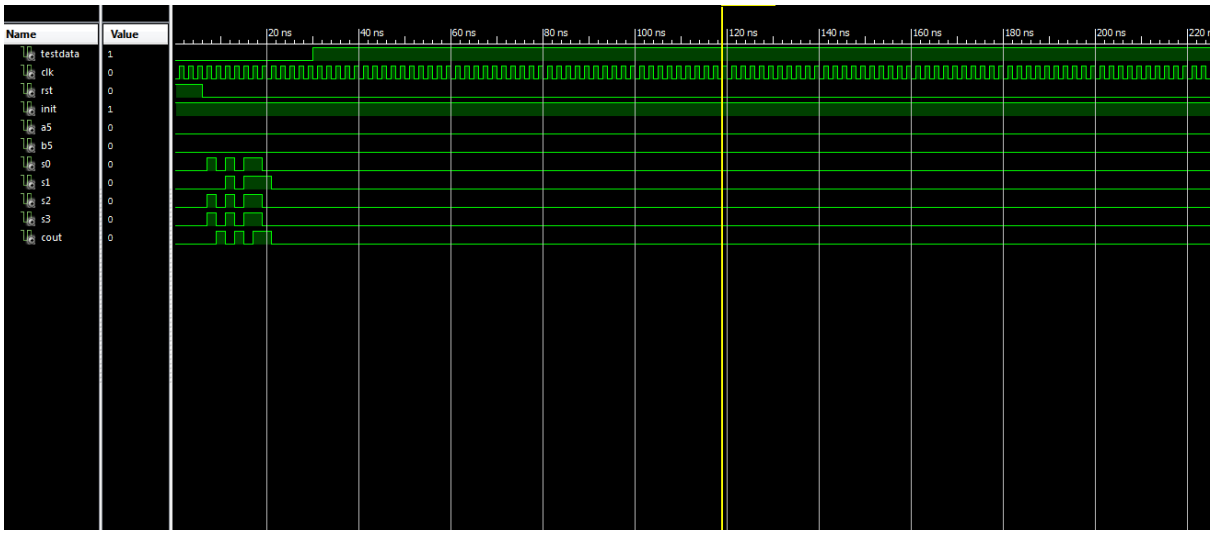


Triple fault model

RESULTS

In quadded logic or TMR/TIR processes, area overhead is very high to make architecture self-reconfigurable. In TMR, the full system is triplicate and an extra voter is needed. In case of quad logic, area requirement is more than four times of the original design. Here we have used dynamic reconfiguration technique where the system can be made self reconfigurable with minimal hardware requirement. In case of RCA, we have designed a 4-bit fault tolerant RCA using one redundant FA. If we use the same single redundant FA for designing 8-bit RCA, area cost decreases. If we cascade our two 4-bit fault tolerant RCAs by connecting the Cout of first fault tolerant RCA to the C_1 of the next fault tolerant RCA for designing an 8-bit RCA, the design now can itself tolerate two faults occurring in two RCAs separately, but area cost increases.

Simulation Waveforms



Triple fault tolerant ripple carry adder

CONCLUSION

In this paper, we have presented fault tolerant adder design assuming single double and triple fault models. In case of RCA, we have designed a 4-bit fault tolerant RCA using one redundant FA, two redundant FA and three redundant FA are designed.

REFERENCES

[1]. Y Sato, S. Kajihara, Y. Miura, T.Yoneda, S. Ohtake, I Inoue, H.Fujiwara, "circuit failure prediction mechanism (DART) for high field reliability," IEEE 8th International Conference on ASICON '09, pp. 581-584, Oct. 2009.

[2]. B. A. Prasad, and F. G. Gray, "Multiple fault detection in arrays of combinational cells," IEEE Transactions on Computers, vol. 100, no. 8, pp. 794-802, 1975.

[3]. Francisco J. O. Dias, "Truth-table verification of an iterative logic array." IEEE Transactions on Computers, vol. 100, no. 6, pp. 605-613,1976.

[4]. S.-K. Lu, J.-C. Wang and C.-W. Wu, "C-testable design techniques for iterative logic arrays," IEEE Transaction on VLSI Systems, vol. 3, no. 1,pp. 146–152, Mar. 1995,.

[5]. W.-T. Cheng and J. H. Patel, "A minimum test set for multiple fault detection in ripple carry adders," IEEE Transaction on Computers, vol. 36, no.7, pp. 891-895, July 1987,.

[6]. A. Mukherjee; A. S. Dhar, "Design of a Self-Reconfigurable Adder for Fault-Tolerant VLSI Architecture," 2012 International Symposium on Electronic System Design (ISED), pp. 92-96, Dec. 2012,.

[7]. J. Sklansky, "Conditional-sum addition logic," IRE Transactions on Electronic Computers, vol. 2, pp. 226-231, 1960.

[8]. A. Mukherjee, A. S. Dhar, "Design of a Fault-Tolerant Conditional Sum Adder", H. Rahaman et al. (Eds.): VDAT 2012, Springer LNCS 7373, pp. 217–222, 2012

AUTHORS' BIOGRAPHY

Shaik Mabjani, She received B.TECH degree from Priyadarshini College of Engineering and Technology affiliated by JNTU Anatapur in 2013, now pursuing M.TECH in Audisankara College of Engineering and Technology (autonomous).

K. Subramanyam, Working as a Assistant Professor in Dept. of ECE, ASCET, Gudur, Nellore Dist, A.P. He received the B.Tech degree from GKEC (Gokula Krishna College of Engineering), Sullurpet and M.Tech degree from SITAMS, Chittoor. His interested areas are Embedded systems and Communication Systems, VLSI. He thought several subjects for under graduate and post graduate students.