Design and Analysis of a Fault Tolerant Microprocessor Based on Triple Modular Redundancy Using VHDL

Deepti Shinghal[1], Dinesh Chandra[2]
[1]Asst. Prof., Deptt. of Electronics Engg., M.I.T., Moradabad, U.P., INDIA
shinghaldeepti0@gmail.com
[2]Prof. & Head, Deptt. of Electronics Engg., J.S.S.A.T.E., Noida, *U.P., INDIA*
dinesshc@gmail.com

Abstract:

There are numerous real time & operation critical systems in which the failure of the system is unacceptable at any stage of processing. The examples of such systems are like ATM machines, satellites, spacecraft etc. In this paper a fault tolerant microprocessor is developed by using checker units with a fault secure ALU and to develop a fault secure ALU the parity prediction logic and two rail checker method was used. Finally triple modular redundancy is applied to develop a fault tolerant processor. Proposed method was validated using the VHDL test environment and the results showed that the reliability of the system increased with a little area overhead.

Keywords: *Triple Modular Redundancy, Fault Tolerant, Parity Prediction, VHDL.*

1. INTRODUCTION

Computer systems are developed over a period of time. They usually go through a number of phases (stages) starting from the specification phase, through the design, prototyping, and implementation phases and finally the installation phase. A fault can occur during one or more of these phases. A fault is defined as a physical defect that takes place in some part(s) of a system. A fault that occurs during one development stage can become apparent only at some later stage(s). Faults manifest themselves in the form of error(s). When an error is encountered during the operation of a system, it will lead to a failure. A system is said to have failed if it cannot deliver its intended function. Figure 1 shows a simple example that illustrates the three terms.

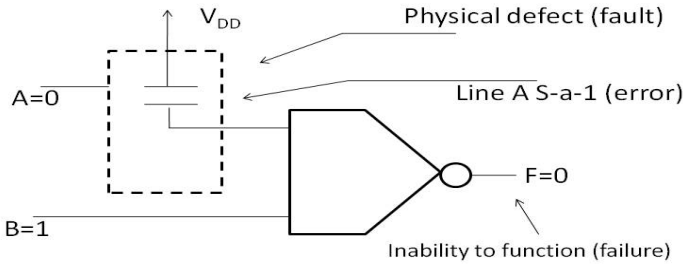

Fig 1 Relationship between failure, error & fault.

A general approach to fault-tolerant design is the use of protective redundancy to permit continued correct operation of a system after the occurrence of specified faults. This protective redundancy is extra hardware, software, information, or time to mask faults or to reconfigure a faulty system. [2] A

discussion of concepts in fault tolerance, including redundancy will be presented in this paper followed by a discussion of the proposed fault-tolerant Microprocessor.

2. CONCEPTS OF FAULT TOLERANCE

In the design of fault-tolerant systems, the designer must consider the possible occurrence of several different kinds of faults such as transient faults, intermittent faults, permanent faults, logical faults, and indeterminate faults. Transient faults, often caused by external disturbances, exist for a finite length of time and are nonrecurring. Intermittent faults occur periodically and typically result from unstable device operation [1]. Permanent faults are perpetual and can be caused by physical damage or design errors. Logical faults occur when inputs or outputs of logic gates are stuck-at-0 or stuck-at-1. Indeterminate faults occur when inputs or outputs of logic gates float between logic 0 and logic 1.

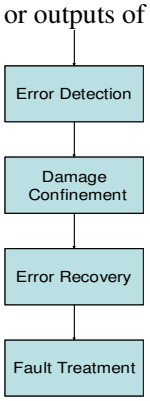

Fig 2 General fault tolerant procedure

A system can operate correctly in the presence of the aforementioned faults if the appropriate form of redundancy is incorporated into the system. Two major fault-tolerant design approaches are static and dynamic redundancy. Static redundancy is the use of redundant components so that faults may be masked. Dynamic redundancy is the reorganization of a system so that the functions of a faulty unit are transferred to other functional units [3]. Four specific types of redundancy are information redundancy, time redundancy, software redundancy, and hardware redundancy. Information redundancy is the use of error detecting or error correcting codes for information representation. Time redundancy is the repetition of system operations so that transient faults can be masked. Software redundancy is the inclusion of several alternative programs for system operations so that software faults (design mistakes) can be tolerated. Hardware redundancy is the inclusion of multiple copies of critical components so that intermittent and permanent faults can be tolerated

Hardware redundancy is the concept used in a very popular architecture for fault-tolerant processors. This architecture is referred to as multiprocessors [8]. A multiprocessor system is a computer system that is made up of several CPUs or, more generally, processing elements which share computational tasks. Multiprocessors are different from multicomputer systems which have several processing elements working independently on separate tasks. The processing elements of multiprocessors typically share communication facilities, 110 devices, program libraries, and databases. Additionally, all of these processing elements are controlled by the same operating system.

The two main reasons for including multiple processing elements in a single computer system are to improve performance and to increase reliability. Performance improvement is obtained either by allowing many processing elements to share the computation load associated with a single large task, or by allowing many smaller tasks to be performed in parallel in separate processing elements. A multiprocessor consisting of n identical processors is an example of an n-unit processor that can, in principle, provide n times the performance of a comparable single-unit system or uniprocessor. The fact that the failure of one CPU does not cause the entire system to fail improves the system reliability. The functions of the faulty processor can be taken over by the other processors which mean that the system is fault-tolerant.

3. FAULT TOLERANT MICROPROCESSOR

The primary function of a processor, is to execute sequences of instructions stored in memory (main memory), which is external to the CPU. The CPU also supervises the other system components, usually via special control lines. For example, the CPU directly or indirectly controls I/O operations such as data transfers between I/O devices and main memory.
The CPU contains several registers which are used for temporary storage of instructions and operands, and an arithmetic-logic unit (ALU) which executes data processing instruction. The processor in Figure 3 is the PARWAN processor that will be transformed into a fault-tolerant system [5]. It is proposed that redundancy to the critical components of this processor may provide a feasible alternative to the multiprocessor architecture. The technique of fault tolerance that will be used is triple modular redundancy (TMR).

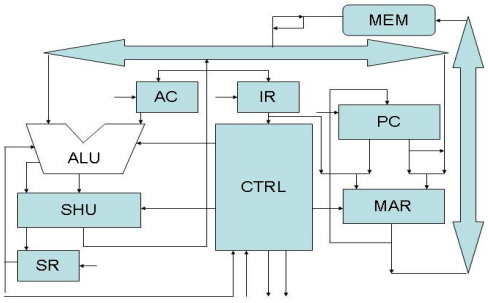

Fig 3 Fault tolerant microprocessor

TMR is the replication of a component into three identical copies where all copies contain the same information and perform the operation at the same time. The output of all three components is then voted upon by a voter module, and the majority output is selected. Therefore, if two of the three copies are functionally correct, the voter will produce that correct output. TMR is arguably the most reliable technique in fault tolerance, but it tends to be bulky when used with large system components.

4. PROPOSED TESTING AND DIAGNOSE APPROACH

4.1. Testing via TMR

A Fault masking technique, for instance TMR, was embedded within the most important Microprocessor's component like control unit and ALU. The TMR technique [6] on line detects the operational faults and specially tolerates the transient faults. However, TMR produces an extra hardware overhead which can be accepted for the critical systems.
The concept of redundancy implies the addition of information, resources or time beyond what is needed for normal system operation. The redundancy can take one of several forms, including hardware, software, information and time redundancies. The physical replication of hardware is perhaps the most common form of redundancy used in systems.
As semiconductor components have become smaller and less expensive, the concept of hardware redundancy has become more common and more practical. The fault masking technique concept is to hide the occurrence of faults and prevent the faults from resulting in errors. The most common form of fault masking is the TMR. The basic concept of TMR is to triplicate the hardware and perform a majority vote to determine the output of the system. If one of the modules becomes faulty, the two remaining fault free modules mask the results of the faulty one when the majority vote is performed. The primary difficulty with TMR is obviously the voter; if the voter fails, the complete system fails.

In this article, the voters were considered as faulty free. The TMR technique is applied only on ALU and CU as a case study as shown in Figure 4.

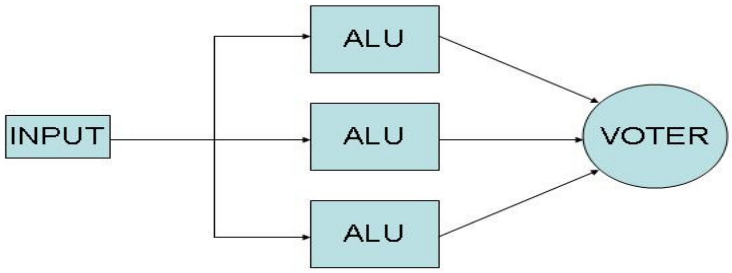

Fig 4 Triple Modular Redundancy (TMR) for ALU

4.2. Characteristics of VHDL

Another concern in the design of a fault-tolerant CPU is the design tool that will be used for layout and verification. Regardless of the fault-tolerant approach that is implemented, VHDL would serve as an excellent language to use in the modeling of the processor.
Higher level languages, in general, are really the new system-design approach. Currently, designers are being asked to manage the design of very large ASICs and very large systems. These tasks are far too complex for one person, therefore hardware description languages are being used for managing design complexity.
VHDL is the very high speed integrated circuit hardware description language. In VHDL a given logic circuit is represented as a design entity [4]. The logic circuit represented can be as complicated as a microprocessor or as simple as an AND gate. The design entity consists of two different types of descriptions which are the interface description and one or more architectural bodies. The interface description names the entity and describes its inputs and outputs. The architectural body specifies the behavior of the entity directly or through the structural decomposition of the body in terms of simpler component. Due to these basic characteristics and more complex characteristics not mentioned here, VHDL would serve as an excellent language for the modeling of a processor.

5. IMPLEMENTATION

Figure5 is the logical diagram of fault tolerant ALU that was implemented in active HDL 8.1. To make the processor fault tolerant, a fault secure adder/ALU is taken as a base unit. To develop a fault secure adder/ALU two techniques are applied.

- Parity prediction logic
- Two rail checker

The parity prediction logic helps in predicting correct parity of input and output. The two rail checker is helpful for checking the carries. The drawback of these two techniques is in the form of area overhead, but the area can be optimized using the techniques such as removing the parity generator, avoiding duplication of complex carry generation blocks, use of partial carry duplication, and logic gate optimization.

A simulation study has been done by executing some instructions stored in ROM for the three ALU modules to check the correctness of Microprocessor design and implementation. A fault has been injected at one time to one of the three ALU modules and the voter selects one from the majority of ALU. This can be clearly shown by the result verification waveform of voter unit.

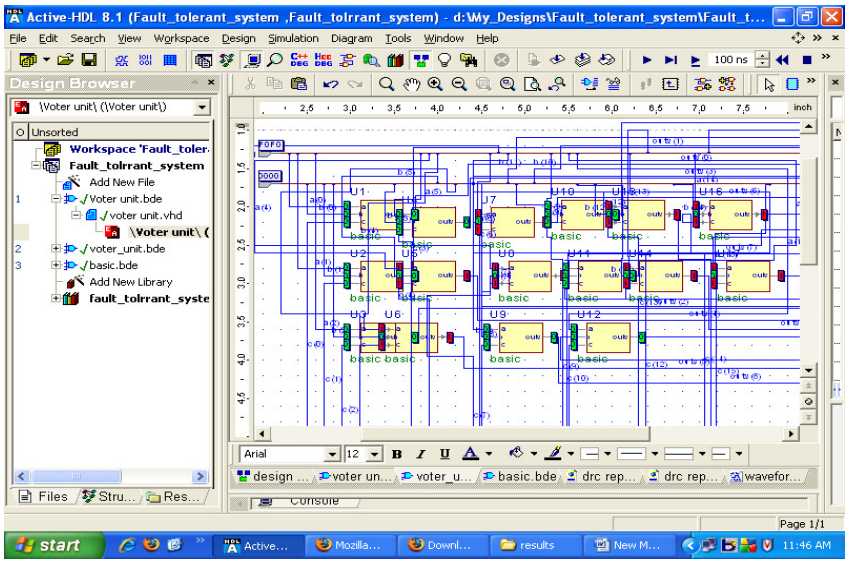

Fig 5 Snapshot view of fault tolerant ALU

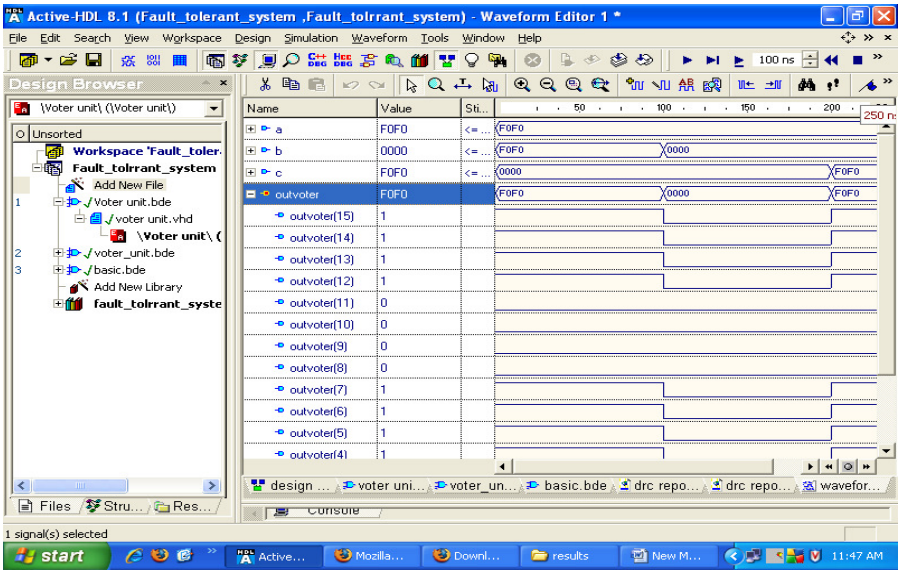

Fig 6 Result verification waveform of voter unit

6. COMPARISON OF TMR WITH ALTERNATIVE FAULT TOLERANT DESIGN TECHNIQUES.

Although TMR has been shown to significantly improve design reliability, it carries a high overhead cost. At a minimum, full TMR of a design requires three times the hardware to implement three identical copies of a given circuit. In addition, additional logic is required to implement the majority logic voters. In the worst case, TMR can require up to six times the area of the original circuit. The additional hardware resources required to triplicate the original circuit result in other secondary problems such as increased power and slower timing space should be left above and below the table. This paper evaluates two additional fault tolerance techniques and compares them to TMR. These include quadded logic and temporal redundancy, all well-known techniques in custom circuit

technologies. Each of these techniques is compared to TMR in both area cost and fault tolerance. For comparison the data for the other two techniques have been taken from the standard results.

Table 1. Comparison between various Fault tolerant techniques

Study	Design	Technique	Area (LUTs/FFs/Slices)	Area Overhead (%)	Sensitivity Reduction/Reliability (%)
Temporal Redundancy	36-bit Adder	None	109/70/73	n/a	n/a
		TSTMR	121/136/125	71	-66
		QTR	114/138/115	58	-68
		TMR	327/246/407	458	49
Quadded Logic	Circuit1 (8bit adder)	None	3/0/2	n/a	n/a
		QL	37/0/21	1133	14.8
		TMR	10/0/6	233	27
	Circuit2 (8bit comparator)	None	3/0/2	n/a	n/a
		QL	29/0/16	866	15
		TMR	10/0/6	233	27.2

7. RESULTS AND CONCLUSION

As mentioned earlier, the objective of this paper was to design and implement a fault tolerant microprocessor, and then do the reliability analysis for this processor. For this purpose simulation of the design net list was done using Active HDL 8.1.
Finally a comparison of the TMR technique used to develop Fault tolerant microprocessor was done with the two other techniques temporal redundancy and Quadded logic in terms of reliability, area and speed.

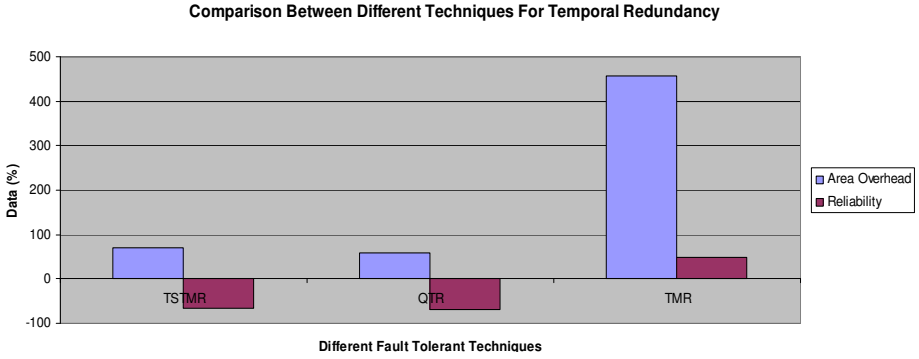

Fig 7 Comparison between different techniques for temporal redundancy

The comparison showed that none of the techniques evaluated provide greater reliability than TMR and that these techniques are often more costly than TMR. In fact, the following techniques actually decreased the design reliability: TSTMR, QTMR. Despite the reliability decrease, TSTMR and QTR did have a much lower area overhead than TMR—only 71% and 58% more than the original circuit compared to 458% for TMR.(Figure7).
Quadded logic did actually improve reliability, but not by as much as TMR. In other words, TMR had much higher gains in reliability for a slightly larger area cost. Quadded logic reduced the sensitivity of circuit1 (8 bit adder) by 14.8% compared to 27% with TMR (Figure 8). For circuit2 (8 bit comparator), quadded logic reduced the sensitivity only 15% while TMR reduced the sensitivity about 27% (Figure 9). Quadded logic also required more area than TMR with 866% to 1133% area

overhead compared to 233% for TMR. In other words, TMR yielded much higher reliability in comparison to other techniques at substantially smaller area costs.

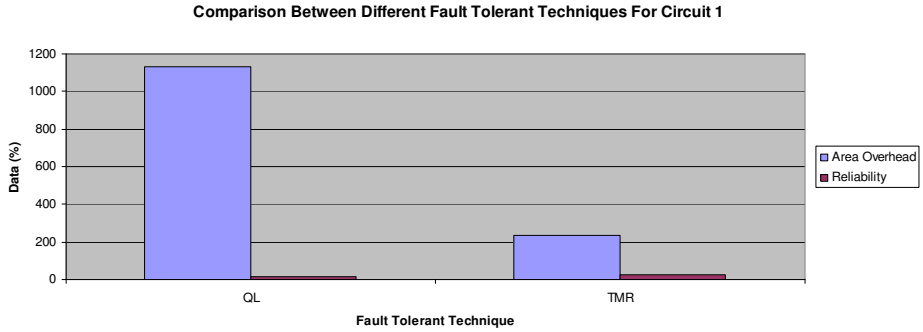

Fig 8 Comparison between different fault tolerant techniques for circuit 1

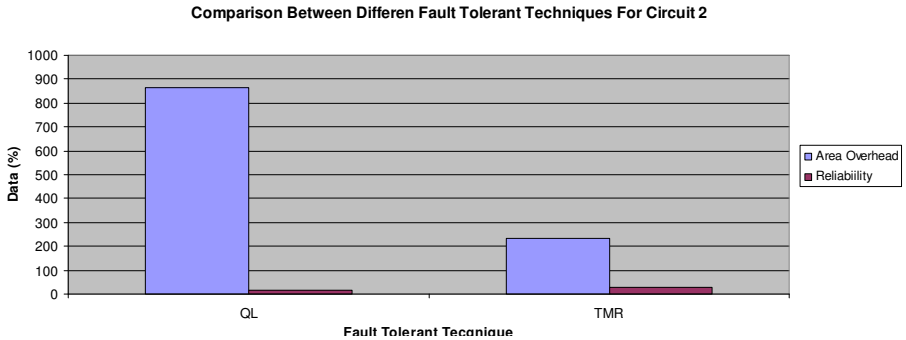

Fig 9 Comparison between fault tolerant techniques for circuit2

References

[1]. A.M. Awlendola, A. Benso, F. Coma, L. Impagliazzo, P. Marmo, P. Prinetto,, M. Re baudengo, M.Sonza Reorda(1996)'fault behaviour observation of a microprocessor system through a VHDL simulation – based fault injection experiment' IEEE, pp 536-541.

[2]. Fred A. Bower, Daniel J. Sorin, and Sule Ozev (2005) ' A Mechanism for Online Diagnosis of Hard Faults in Microprocessors,' Appears in the 38th Annual International Symposium on Microarchitecture (MICRO) Barcelona, Spain, November, 2005.

[3]. Hamid R. Zarandi,Seyed Ghassem Miremadi, Alireza Ejlali, (2003) 'Fault Injection into Verilog Models for Dependability Evaluation of Digital System,' Proceedings of the second International Symposium on Parallel and Distributed Computing (ISPDC'03), IEEE 2003.

[4]. P. J. Ashenden, The VHDL CookBook, University of Adelaide, South Australia, Technical Report, 1990.

[5]. Richard B. Brown, Senior Member, Ronald J. Lomax, Gordon Carichner, and Alan J. Drake (2000) 'A Microprocessor Design Project in an Introductory VLSI Course' IEEE Transactions on Education, VOL. 43, NO. 3, pp 353-361.

[6]. J.F. Wakerly, (1974) 'Transient Failures in Triple Modular Redundancy Systems with Sequential Modules", IEEETC, May 1974.

[7]. Carver Mead and Lynn Conway, Introduction to VLSI Systems, Addison Wesely, 1979.

[8]. D. K. Pradhan, Fault-Tolerant Computer System Design, Prentice Hall PTR, pp. 280-385, New Jersey, 1996.