# FPGA BASED RELIABILITY SYSTEM USING MODULAR REDUNDANCY TECHNIQUE

Yamini Agrawal, Shavi Goel, Priyanka Das, Prof. Prasheel N. Thakre

*Department of Electronics and Communication Engineering, Shri Ramdeobaba College of Engineering and Management, Nagpur, India.*

## Abstract

*This article validates the reduction in hardware utilization to establish a reliable system at the cost of time. Redundancy is a technique to improve reliability and availability of a system. Triple modular redundancy uses three functionally equivalent units to provide a redundant backup and it has a majority voter circuit which produces a single output. An alternative way to have a fault less system and the idea of boosting the dependency of a system is fault tolerant system. The most fundamental method used for the fault tolerant system is triple modular redundancy in which the majority voter circuit obtains fault free response. Traditionally, for building a fifth modular redundant system, five different modules are required which increases the hardware substantially. Hence, to reduce the hardware, we have established a fifth modular redundant system using a combination of triple modular redundancy and time redundancy***.** *In this study, we have used 2bit multiplier using different modular redundancy modules to verify our purpose.*

**Keywords:** *Fault tolerant system, fifth modular redundancy, time redundancy, TMR, Reliability system, Embedded systems.*

## 1 INTRODUCTION

As the utilization growth is increasing in FPGA based designs in various applications currently, for example in aerospace, traffic signaling and biomedical, there is a need for a robust and fault tolerant design in order to safeguard systems from failure. For that purpose, various redundant techniques are used to implement the logic on FPGA. One of the most persistent used and the state of art technique for FPGA systems is Modular Redundancy Technique. Modular redundancy is the most popular to achieve the timing constraints and reliability. It enlist multiple replicas of the same module and the approach to provide fault tolerance is through redundancy in real time systems.One of the widely used and commonly known used fault tolerant modular redundancy technique is triple modular redundancy. In this paper, we provide a new approach to provide fault tolerance in the system using fifth level modular redundancy by introducing a delay.

### 1.1 Fault tolerant system

Fault tolerance is a high-performance system and the ability of system to continue error free operation in presence of unexpected fault. The proven ability to continue operating irrespective of the failure of a limited subset in the hardware or software is fault tolerance. The ultimate aim of designing fault tolerant system is to enhance reliability by providing conditions for a system to continue its operation despite of existing of some faults. The main goal of a fault tolerant system is that no single fault can fail the system. Digital systems have more critical tasks therefore they need higher reliability. Usually using design techniques and components with high quality do not decrease failure probability sufficiently.

**PURPOSE OF SYSTEMS TO BE FAULT TOLERANT**:

Due to operations in harsh environment electronic systems are prone to faults. As per rapid increase in the complexity of applications, it tends to put even stronger demand on the performance and reliability of used system. To meet such requirement of reliability of used system, embedded system must be equipped with suitable error detection and meeting the timing constraint limitation.

### 1.2 Redundancy

Redundancy is the duplication of critical components or function of the system with purpose of enhancing reliability of the system as in a form of backup or fail-safe. Faults which occur in the safety critical systems can lead to the failure of the entire system and turns into higher economical losses or endanger human health. For instance, space, aerospace or medical systems which are working in the environment with rapid increase in faults can serve.

Hardware redundancy- such as dual modular redundancy and triple modular redundancy.

Information redundancy- such as error detection and correction methods.

Time redundancy- performing the same operation multiple times such as multiple executions of a program.

Software redundancy- such as N-version programming.

### 1.3 Modular Redundancy

Modular redundancy is a method of masking the fault instead of detecting them. In modular redundancy we hide the fault based on polling. It is a fault tolerant architecture based on n-identical modules which accomplish the same goal. The input of this modules receive the same data. Their outputs are supplied to a majority polling circuit.

#### 1.3.1 TRIPLE MODULAR REDUNDANCY:

Triple modular redundancy is a fault tolerant form of n-modular redundancy.

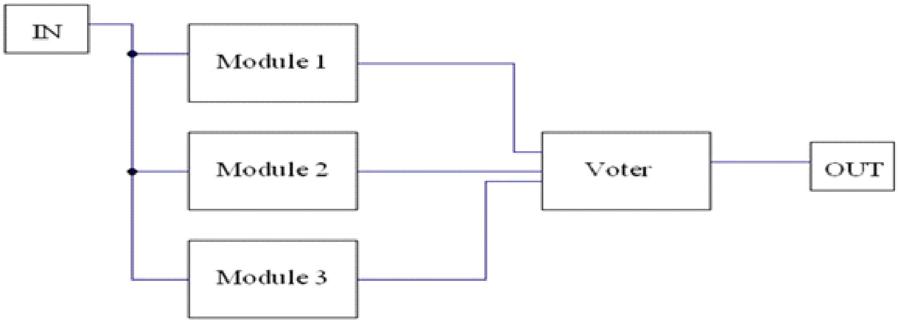
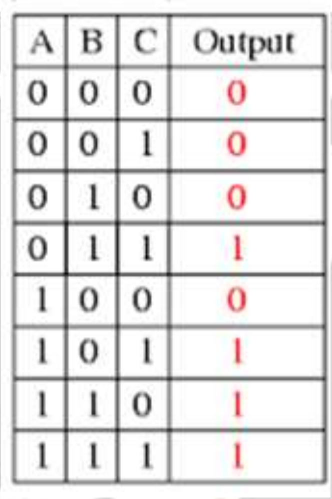


Fig 1: Triple modular redundancy



| A | B | C | Output |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Fig 2: Truth table voter block (3 bits)

Triple modular redundancy is composed of three modules (module 1, module 2 and module 3) and a voter. Here module 2 and module 3 are the replication of module 1 and they all accept the same input values the output of these modules should be consistent. The voter is the majority voter which takes the majority of input to be the output value.

Due to a fault in the system, one of these 3 modules have an error inside and will generate different output. This inconsistency will be caught and corrected by the voter. Thus, the voter output is always a correct value under the assumption of single error.

## 2. FIFTH MODULAR REDUNDANCY:

We increase the number of modules in order to enhance the reliability of the system. In case, in TMR if any one of the systems fail then system will work but if 2 or 3 modules gets failed then the entire system stops working. So, in order to make system work persistently though up to 2 modules gets failed, we can apply for 5th modular redundancy.

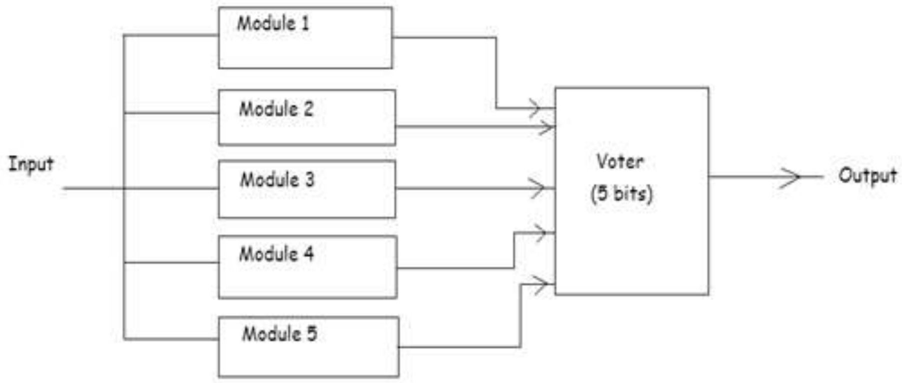
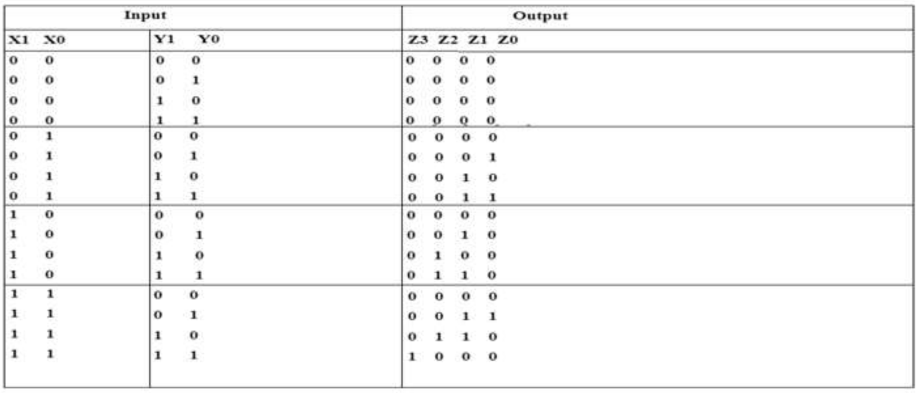


Fig 3: Fifth Modular Redundancy



| Input | | | | Output | | | |
|---|---|---|---|---|---|---|---|
| X1 | X0 | Y1 | Y0 | Z3 | Z2 | Z1 | Z0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

Fig 4: Truth-table of Fifth modular redundancy

## 3. Modular redundancy with delay

The use of time redundancy along with modular redundancy gives us an efficient utilization of the reliable system, that is, the hardware is reduced. In real time operating systems, we assume that error occurs at a paticular time ‘t’ only. In such systems the error is encountered at time ‘t’ and the system works error free otherwise. So, by using the delay element, we are taking the output at an instant when there is no error, that is the 3$^{rd}$ output is being taken after a delay of ‘dt’ time. This makes as a error-free input for the voter in the modular redundant system.

### 3.1. TMR with delay

We prove our statement by initially analysing TMR where we have built a triple modular redundant system using only 2 modules which consist of the central system and the backup module having an additional delay element. The output after the delay element makes the required 3rd input for the 3 bit voter in TMR system.

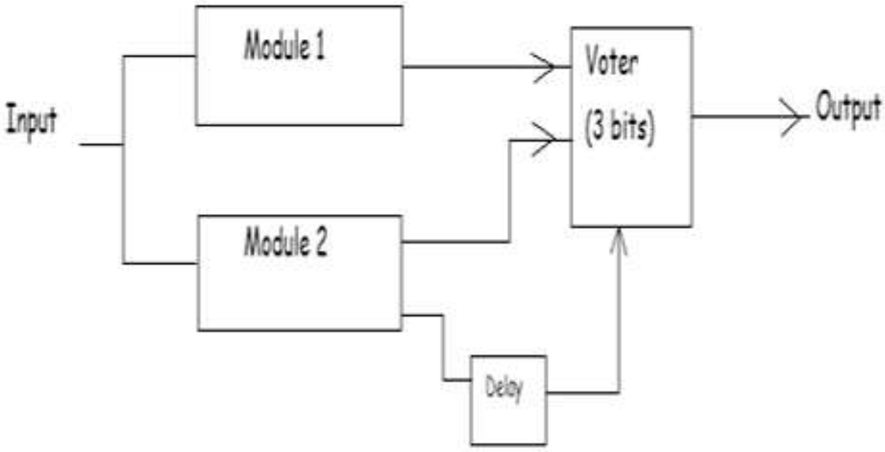


Fig 5. TMR with 2 modules and delay element.

### 3.2. 5th level modular redundancy with delay

Let's assume the occurrence of fault at time 't' and creating a small delay of dt, we will get fault free output. Therefore, we can obtain a 5th modular redundancy by 3 modules just by adding a delay element in the 2nd and 3rd module.

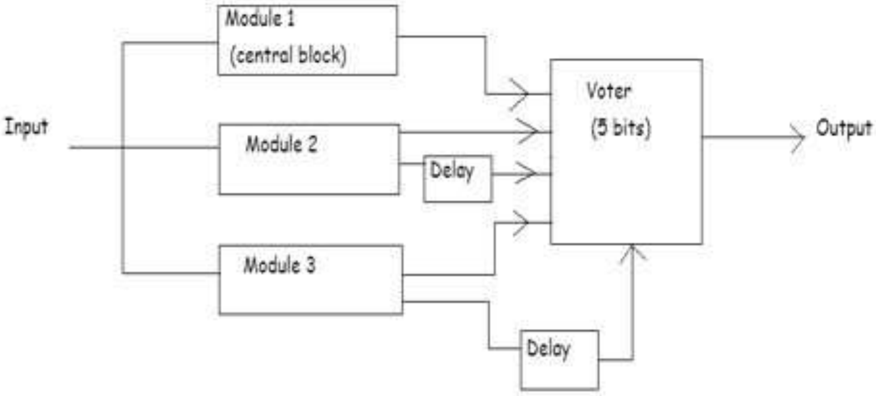
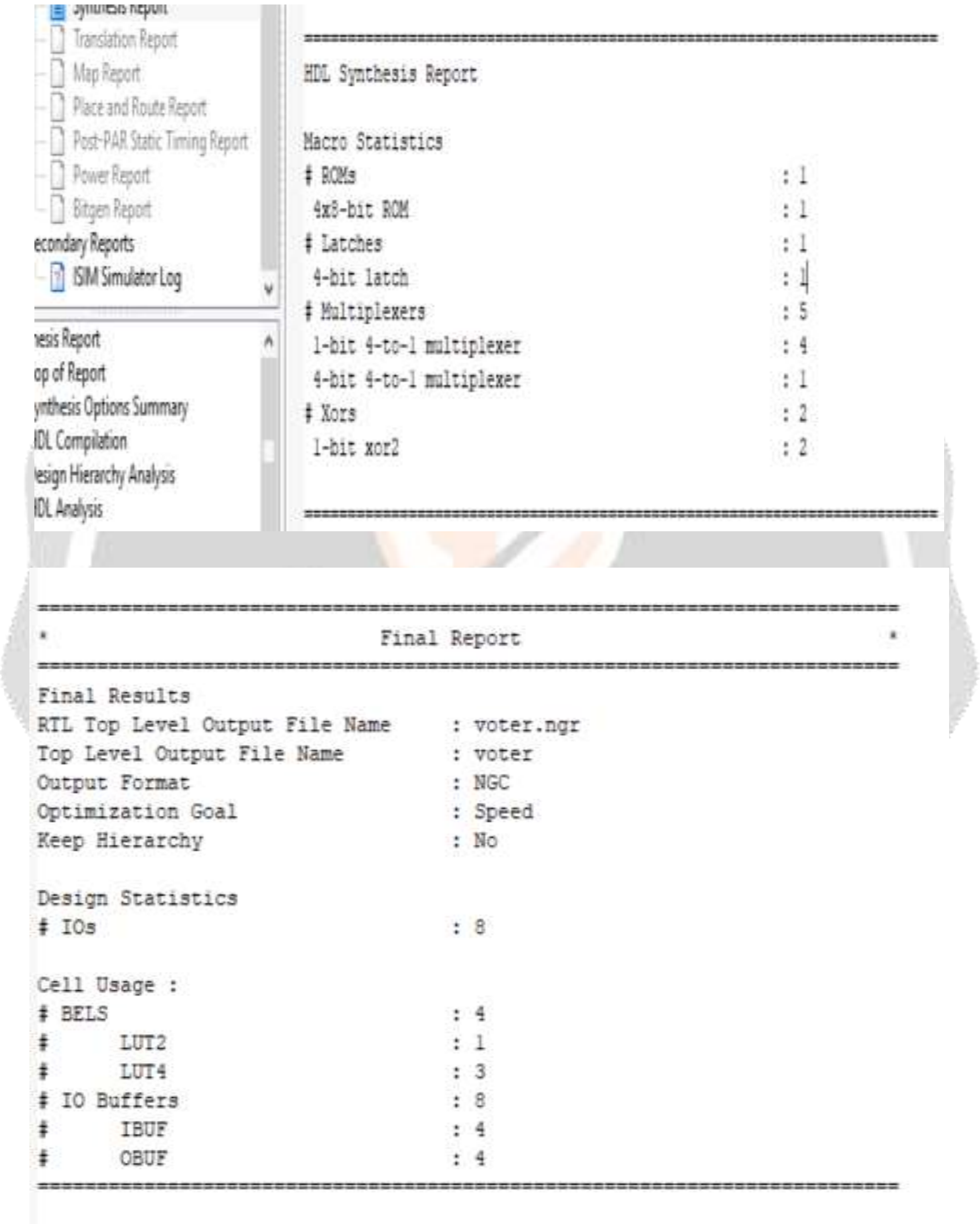


Fig 6. 5th level modular redundancy with only 3 modules and delay in module 2 and module 3.1

## 4. SIMULATIONS AND RESULT

### 4.1 Compilation and Synthesis report

All components generated for 5th level modular redundancy design was simulated. Compilation of code generated for function is successful. Total logical element used is 14. Compilation is done for error correction and it was successful.



```
=========================================================================
HDL Synthesis Report

Macro Statistics
# ROMs                                                 : 1
 4x8-bit ROM                                           : 1
# Latches                                              : 1
 4-bit latch                                           : 1
# Multiplexers                                         : 5
 1-bit 4-to-1 multiplexer                              : 4
 4-bit 4-to-1 multiplexer                              : 1
# Xors                                                 : 2
 1-bit xor2                                            : 2

=========================================================================
```

```
=========================================================================
*                            Final Report                               *
=========================================================================
Final Results
RTL Top Level Output File Name     : voter.ngr
Top Level Output File Name         : voter
Output Format                      : NGC
Optimization Goal                  : Speed
Keep Hierarchy                     : No

Design Statistics
# IOs                              : 8

Cell Usage :
# BELS                             : 4
#      LUT2                        : 1
#      LUT4                        : 3
# IO Buffers                       : 8
#      IBUF                        : 4
#      OBUF                        : 4
=========================================================================
```

Fig 7. Synthesis Report

The overall delay in the system when we use 5 different modules is of 7.858ns whereas, the delay in the system when we use 3 modules with delay in two backup modules is of 15.665ns. As each module is provided with a delay of 5ns.

### 4.2 Simulation and Result

#### 4.2.1 2-Bit Multiplier using TMR and TMR with delay in only one module

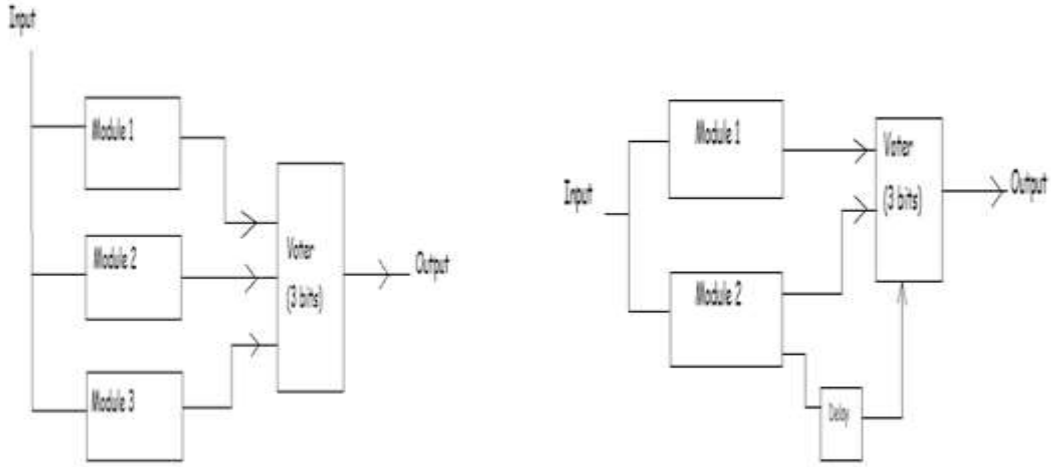


Fig 8. a) 2-Bit Multiplier using TMR, b)2-Bit Multiplier using TMR with delay in module 2.

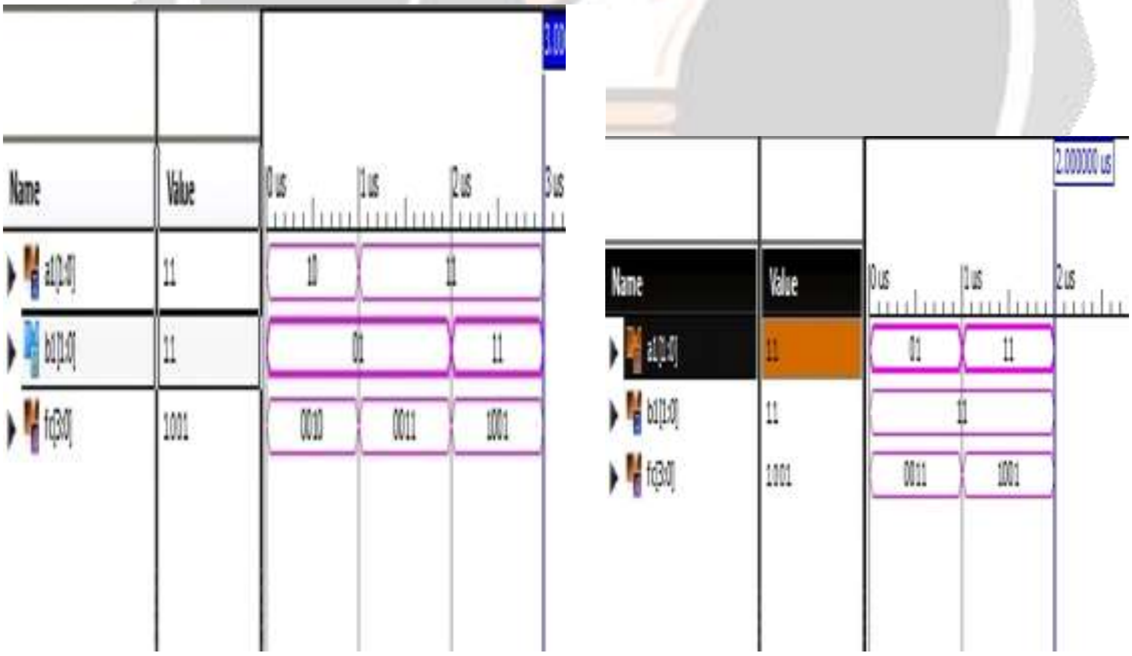


Fig 9. a) Error Free output in 2-Bit Multiplier using TMR, b) Error free 2-Bit Multiplier using TMR with delay in module 2.

### 4.2.2. 2-Bit Multiplier using 5th level modular redundancy and 5th level modular redundancy using only 3 modules and with delay in two modules.

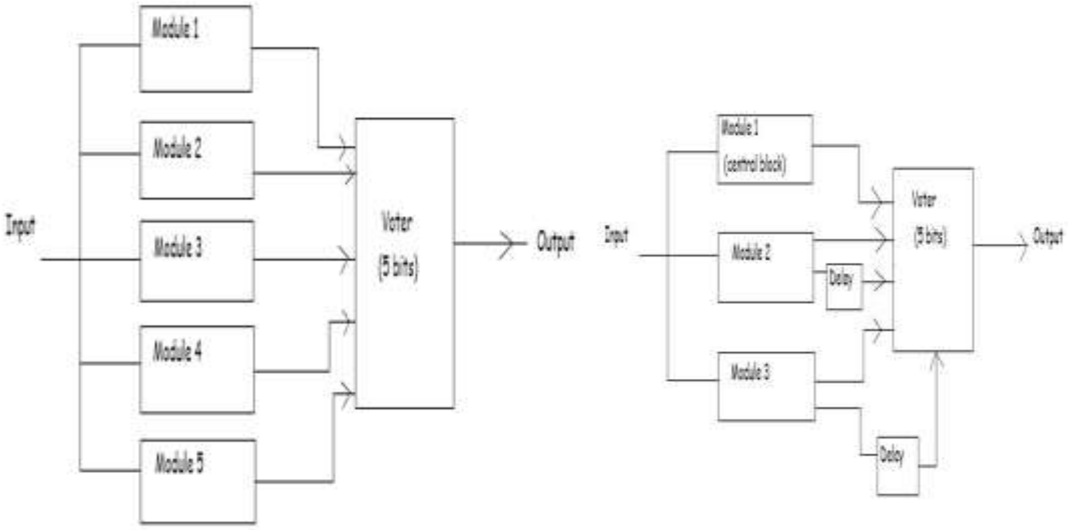
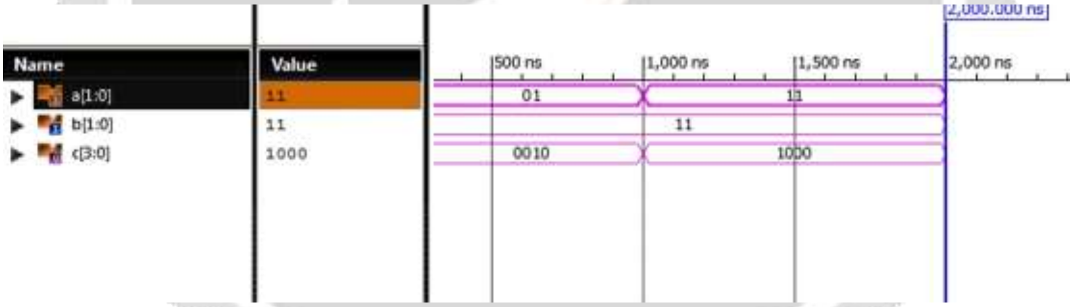
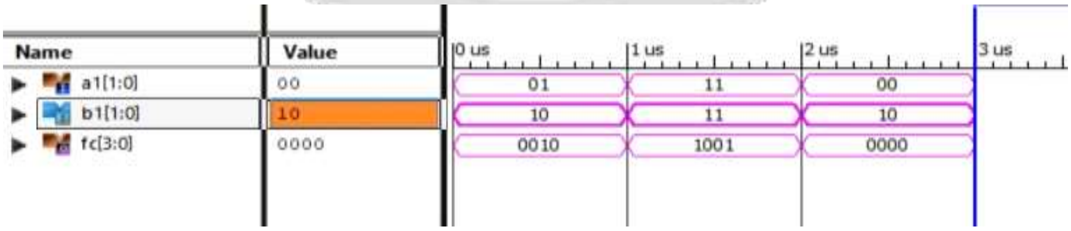


Fig 10. a) 5th level modular redundancy using 5 modules, b) 5th level modular redundancy using delay in module 2 and module 3.



| Name | Value |
|---|---|
| a[1:0] | 11 |
| b[1:0] | 11 |
| c[3:0] | 1000 |

Fig 11. a) 5th level modular redundancy without error correction.



| Name | Value |
|---|---|
| a1[1:0] | 00 |
| b1[1:0] | 10 |
| fc[3:0] | 0000 |

Fig 12. b) 5th level modular redundancy with error correction.

## 4. Accuracy

| S.no. | Module 1 | Module 2 before delay | Module 2 after Delay | Module 3 before Delay | Module 3 After Delay | System Status |
|---|---|---|---|---|---|---|
| 1 | X | working | working | working | working | working |
| 2 | working | X | working | working | working | working |
| 3 | working | working | X | working | working | working |
| 4 | working | working | working | X | working | working |
| 5 | working | working | working | working | X | working |
| 6 | X | X | working | working | working | working |
| 7 | X | working | X | working | working | working |
| 8 | X | working | working | X | working | working |
| 9 | X | working | working | working | X | working |
| 10 | working | X | X | working | working | working |
| 11 | working | X | working | X | working | working |
| 12 | working | X | working | working | X | working |
| 13 | working | working | X | X | working | working |
| 14 | working | working | X | working | X | working |
| 15 | working | working | working | X | X | working |
| 16 | X | X | X | working | working | X |
| 17 | X | X | working | X | working | X |
| 18 | X | working | X | X | working | X |
| 19 | working | X | X | X | working | X |
| 20 | working | X | X | working | X | X |
| 21 | working | X | working | X | X | X |

| | | | | | | |
|---|---|---|---|---|---|---|
| 22 | working | working | X | X | X | X |
| 23 | X | X | X | X | working | X |
| 24 | X | X | X | working | X | X |
| 25 | X | X | working | X | X | X |
| 26 | X | working | X | X | X | X |
| 27 | working | X | X | X | X | X |
| 28 | X | X | X | X | X | X |
| 29 | X | X | working | X | X | X |
| 30 | X | working | X | working | X | X |
| 31 | X | working | working | X | X | X |
| 32 | working | working | working | working | working | working |

These 32 models are for stuck at zero fault. The same is true for stuck at one fault.

Therefore,

Accuracy= 32/64 = 50%

## 5. Conclusion

We have designed a fault tolerant system with increased reliability of the system using redundancy techniques. With the help of TMR we have obtained a fault free system. Thus, we obtained a 5th level modular redundancy by using 3 level with an introduction of delay element. Hence, hardware of our system is reduced at the cost of time.

## 6. Future Scope

In order to have a fault tolerant system, future work will be more emphasized on reduction of hardware by introducing a delay and thereby improved efficiency of the system is achieved.

## References

[1].Mostafa Sadeghi, Hossein Soltani, Mohamadreza Khayyambashi, "The study of hardware redundancy techniques to provide a fault tolerant system", in Cumhuriyet University Faculty of Science Science Journal (CSJ), Vol. 36, No: 4 Special Issue (2015), ISSN: 1300-1949.

[2]. Kashif Sagheer Siddiqui, Mirza Altamash Baig. FRAM based TMR (Triple Modular Redundancy) for Fault Tolerance implementation. Proceedings of The Sixth IEEE International Conference on Computer and Information Technology (CIT'06), 2005

[3]. Wei Chen, Rui Gong, Fang Liu, Kui Dai, Zhiying Wang. Improving the Fault Tolerance of a Computer System with Space-Time Triple Modular Redundancy. Proceedings International Conference on Dependable Systems and Networks, pp. 389-98, 23-26 June 2006 3. Mark Hunger and Sybille Hellebrand. The Impact of Manufacturing Defects on the Fault Tolerance of TMR-Systems. 25th International Symposium on Defect and Fault Tolerance in VLSI Systems, (2010)

[4]. Chris Winstead, Yi Luo, Eduardo Monzon, and Abiezer Tejeda. An error correction method for binary and multiple-valued logic. 41st IEEE International Symposium on Multiple-Valued Logic, 2011.

[5]. Jun Yao, Ryoji Watanabe, Kazuhiro Yoshimura, Takashi Nakada, Hajime Shimada, and Yasuhiko Nakashima. An Efficient and Reliable 1.5- way Processor by Fusion of Space and Time Redundancies, IEEE TRANSACTION, 2011.

[6]. C. Y. Lee, W. Y. Lee, and P. K. Meher, “Fault-tolerant bitparallel multiplier for polynomial basis of GF(2m),” in Proc. IEEE Int. Conf. Circuits Syst. Testing and Diagnosis, 2009, pp. 1–4.

[7]. Shubham C. Anjankar, Dr. Mahesh T. Kolte, “Fault Tolerant and Correction System Using Triple Modular Redundancy”, in International Journal of Emerging Engineering Research and Technology Volume 2, Issue 2, May 2014, PP 187-191.

[8]. Shubham C. Anjankar, Dr. Mahesh T. Kolte, Ajinkya Pund, Pratiksha Kolte, Ankita Kumar, Pranav Mankar, Kunal Ambhore, “FPGA Based Multiple Fault Tolerant and Recoverable Technique Using Triple Modular Redundancy (FRTMR)”, in 7th International Conference on Communication, Computing and Virtualization 2016.

[9]. R. Forsati, K. Faez, F. Moradi, and A. Rahbar, “A fault tolerant method for residue arithmetic circuits,” in Proc. IEEE Int. Conf. Information Management.

[10]. H. Fu, M. Cai, L. Fang, P. Liu and J. Dongl, “Research on RTOS-Integrated TMR for Fault Tolerant Systems”,8 *th* ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, IEEE, 2007.

[11]. M. H. Mottaghi, H. R. Zarandi, "DFTS: A Dynamic Fault-Tolerant Scheduling for RealTime Tasks in Multicore Processors," Elsevier Journal of Microprocessors & Microsystems, vol. 38, no. 1, PP. 88-97, 2014.