Implementation of High Reliable Fine Grain Fault Tolerance Redundant Technique for FPGA

K. Sreelaxmi, B. Srinivas, M. J. C. Prasad

Abstract- SRAM based FPGAs are attractive to use in space applications because of more flexibility and reprogram ability. As technology size decreases below nanometer SRAM based FPGAs are more susceptible to radiation. These effects can cause transient or permanent bit flipping on SRAM cells and respectively change the function of logic elements within FPGAs. Fault-masking methodologies are essential, because it is vital for the system to work always properly irrespective of various faults that occurs in Complex digital circuitry. Due to this fact, redundancy techniques, which target fault masking and fault tolerance are in our scope.

In this project we are proposing Quadruple Force Decide Redundancy (QFDR) a new approach in fault tolerance for mitigation problems in digital circuits, as simply replicating complete systems in Triple Modular Redundancy (TMR) technique may not be sufficient anymore when especially applies to the space applications, failure rate increases because of second instance occurs before the first one recovers. It QFDR makes SRAM-based FPGAs effectively immune from SEU (Single Event Up-set) mitigation challenges.

The proposed QFDR is operated at an abstraction level of CLBs of FPGA. The Quadruple Force Decide Redundancy (QFDR) is a redundant logical structure which quadruplicates logical functions and defines two different Force and Decide rules for different quadruple logic functions based on their level in design and then connects them together using special connection patterns. The complete logic of QFDR is implemented in VHDL.

Modelsim Xilinx edition (MXE) will be used for simulation and functional verification. Xilinx ISE will be used for synthesis. Xilinx FPGA board will be used for testing and demonstration of the implemented system.

I. INTRODUCTION

Fault-tolerant systems can capable of executing their tasks correctly regardless of either hardware failures or software errors. In practice - we can never guarantee the faultless execution of tasks under any circumstances. Critical applications require extreme fault tolerance (e.g., aircrafts, nuclear reactors etc). A malfunction of a computer in such applications can lead to accident. Their probability of failure must be low as possible as one in a billion per hour of operation. Systems operating in a space with high failure possibilities are due to the Electromagnetic disturbances

Fault-tolerant systems are typically based on the concept of redundancy. a coarse-grained description of a system regards large subcomponents while a fine-grained description regards smaller components.

Manuscript Received October 28, 2013.

K. Sreelaxmi, Electronics and Communication Engineering, Malla Reddy engineering College, secunderabad, India.

B. Srinivas, Electronics and Communication Engineering, Malla Reddy engineering College, secunderabad, India.

M.J.C.prasad, Electronics and Communication Engineering, Malla Reddy engineering College, secunderabad, India.

The terms granularity, coarse, and fine are relative, used when comparing systems or descriptions of systems. The scope of the project is that when fault occurring in the IC'S use the QFDR (QUADRUPLE FORCE DECIDE REDUNDANCY) concept for working correctly if error occurred in the FPGA.

Reconfigurable architectures such as Field Programmable Gate Arrays (FPGA) have gained more and more importance in recent years. State of the Art SRAM-based FPGAs embed megabits of RAMs and plenty of configurable logic and routing resources, which makes it possible to implement a circuit composed of millions of gates. The possibility to integrate a complex Systems-on-Chip (SoC) on a single device, combining speedup of hardware with flexibility of software, the opportunity of practically unlimited reconfiguration in SRAM based FPGAs as well as drastically sinking prices, made FPGAs attractive for a wide variety of application domains. In special application domains such as space, FPGAs are becoming increasingly popular as they are inherently flexible to meet multiple requirements and offer significant performance and cost saving for such critical applications.

Mitigation techniques are necessary to note before we are going to the specifics of mitigation: Some different aspects we deal with are to make distinguish between error masking and error correction. When an error is masked, there is not any feedback. Even for those upset, which occurs, it will not propagate through the circuit. However, whatever has been upset, stay upset and just is masked. It can be accumulated, and the system goes down. Error correction has feedback. Error correction must determine what will be right or be correct to be stored. The best mitigation will be a combination of both masking and correction such that error can not propagate through the design; also is able to be corrected.

Programmable switches in FPGAs can be implemented, antifused or SRAM-based. Antifused-based ones are one-time programmable while SRAM based are re-programmable. Antifused is similar to fuse technology in that is one-time programmable. The programming method in anti-fuse is done by growing a link to make a connection instead of passing current through a metal connection and breaking it. Anti-fuses can be amorphous silicon or metal-to-metal connections. They use materials which normally reside in high impedance state but can be fused irreversibly into low impedance states.

SRAM-based ones in contrast, employ SRAM (Static RAM) cells to control pass transistors and transmission gates. The RAM cell in the memory device is designed for fastest possible READ/WRITE performance. SRAM-based devices are first powered on and then booted because the static memory is volatile (When the power is turned off, the contents disappear).This causes programmability and re-programmability, even in

real time. Therefore, SRAM-based FPGAs are suitable in reconfigurable computing applications.

1.2 REDUNDANCY:

Redundancy is at the heart of fault-tolerance. Incorporation of extra components in the design of a system to improve its reliability.

Redundancy consists of five types of formats:

1. Hardware redundancy (spatial redundancy)
» Static, dynamic and hybrid redundancy
2. Software redundancy
» N-version programming
3. Information redundancy
» Error detecting and correcting codes
» Usually requires extra hardware for processing
4. Time redundancy
» Re-execution
5. Physical Redundancy
» Physically replicate modules
» Effective for all sorts of faults

II. TRIPLE MODULAR REDUNDANCY

2.1 TRIPLE MODULAR REDUDANCY:

The most general hardware masking technique is Triple modular redundancy. Sometimes triple-mode redundancy (TMR) is called as a fault-tolerant form of N-modular redundancy.

Common Redundancy based Techniques – TMR:

Triple Modular Redundancy (TMR) approach is used when fault-masking is necessary. The basic concept of TMR is that a circuit can be hardened against SEUs and other type of failures by designing three functional and identical copies of the same circuit and building a majority voting on the output of three replicas. By majority voting simply the best two of three wins. In other words, it is very common and easily expressed expression.

TMR systems should use data scrubbing -- rewrite flip-flops periodically -- in order to avoid accumulation of errors.

- Uses identical logic blocks performing the same task.
- The corresponding outputs are compared through majority voters.
- Number of errors occurred are decided by error detection block.

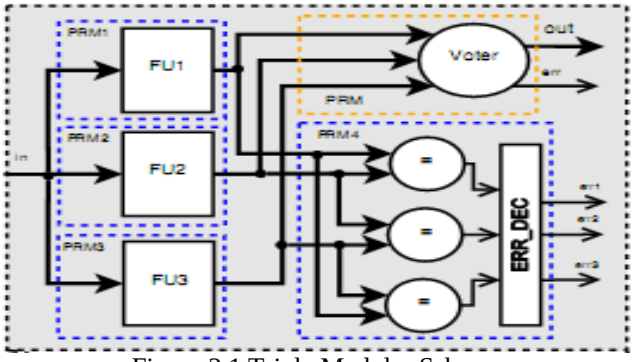


Figure 2.1 Triple Modular Scheme

2.2 Reliability of TMR Systems:

A vector picks the majority output.

Vector can fail –reliability of voter.

$$= R_{vot}(t)\sum_{i=2}^{3}\binom{3}{i}R(t)^{i}(1-R(t))^{3-i}$$

R tmr (t)

$$= \text{Rvot}(t)\,(3R^{2}(t) - 2R^{3}(t)).$$

2.3 Triplicate Processor/Memory System:

All communications (in either direction) between triplicate processors and triplicate memories go through majority voting.Higher reliability than a single majority voting of triplicate processor/memory structure.

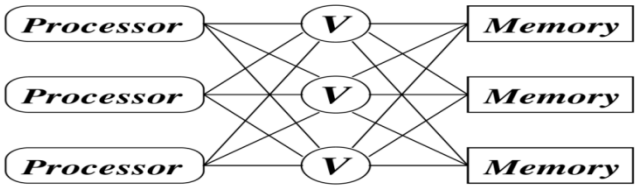


Figure2.2 Voting of triplicate processor/memory structure

Triple Modular Redundancy (TMR) is a well-known fault Tolerant technique for avoiding errors in integrated circuits.TMR is especially suitable for protecting designs synthesized in SRAM-based Field Programmable Gate Arrays (FPGAs) due to the peculiar effect of an upset in the user's combinational logic design.

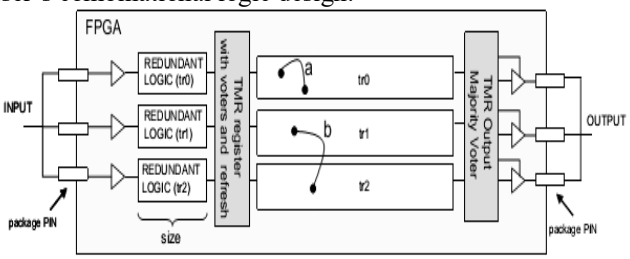


Figure 3.3.Triple Modular Redundancy (TMR) Scheme in the FPGA

2.4 Advantages Of TMR:

- The masking action of fault occurs immediately.
- The separate fault detection mechanism is not necessary before masking the fault.
- Conversion from a non-redundant system to a TMR system is straightforward.

2.5 Disadvantages Of TMR:

- Triple Modular Redundancy (TMR) technique may not be sufficient anymore when especially applies to the environments like space, as higher failure rates may disrupt a second instance before the first one recovers.
- To overcome this problem and to implement fine grain fault tolerance we are using QFDR. QFDR makes SRAM-based FPGAs effectively immune from SEU (Single Event Up-set) mitigation challenges.

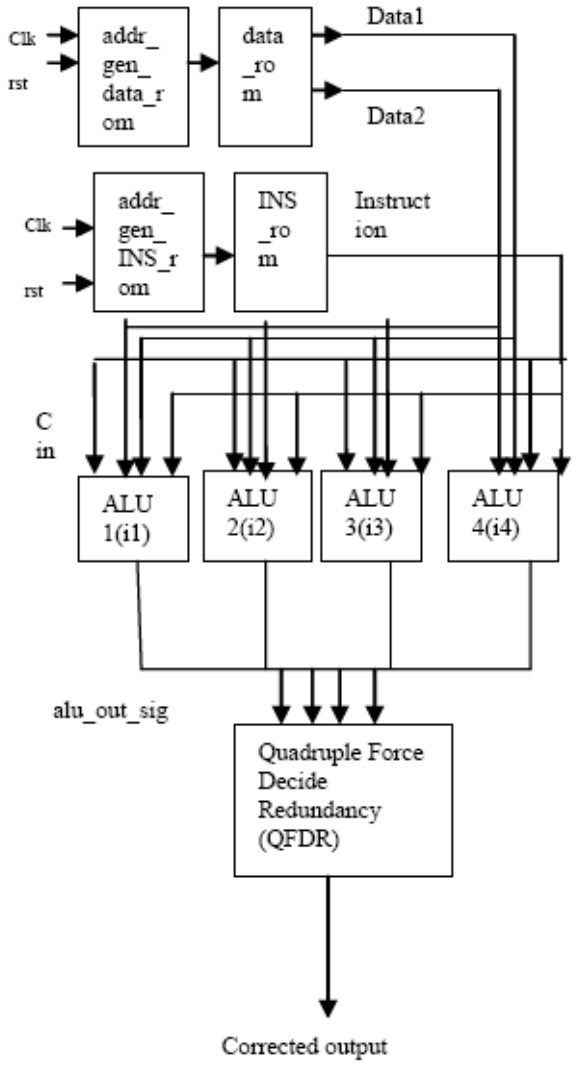
III. IMPLEMENTATION OF QFDR

The Quadruple Force Decide Redundancy (QFDR) is a redundant logical structure which quadruplicates logical functions and defines two different Force and Decide rules for different quadruple logic functions based on their level in design and then connects them together using special connection patterns.

The idea comes from Quadded Logics which are an old method of cleaning the errors up in logic gates. The major difference to Quadded Logic is the underlying functions. While QL only

allows simple gates like AND and OR that have to be interconnected in a certain manner..

3.1 BLOCK DIAGRAM OF QUADRUPLE FORCE DECIDE REDUNDANCY:



The Block diagram of the fine grain fault tolerance which consists of the quadruplicates logical functions(four equal logic modules) and defines two different Force and Decide rules for different quadruple logic functions based on their level in design and then connects them together using special connection patterns.

A quadruple network is constructed using the following underlying assumptions: First, the logical functions must appear in quadruplicated form and all inputs must be duplicated. Every two out of four quadruplicated copies are the duplicated inputs of the next level. Second, errors are corrected in the logic just downstream of the fault that caused it and third, the connection is accomplished by correct signals from the neighbors (in duplicated inputs) of the faulty logic function.

- ***The Decide Force approach works as follow:***

Any difference in duplicated inputs means that one of them has incorrect value. In the first level the output is forced to a zero in case of difference in two duplicated inputs. This means whenever the output is one, note that no error occurred. If it is zero it might be correct or wrong. This way the bit value contains additional information which is used by the next level. The next level knows about the forced output value as the result of the previous level, and in case of zero the output value will not be assumed as the correct value and, respectively, will not be selected as input value of the second level. Functions f and g are modified to satisfy the corresponding force and decide rules.

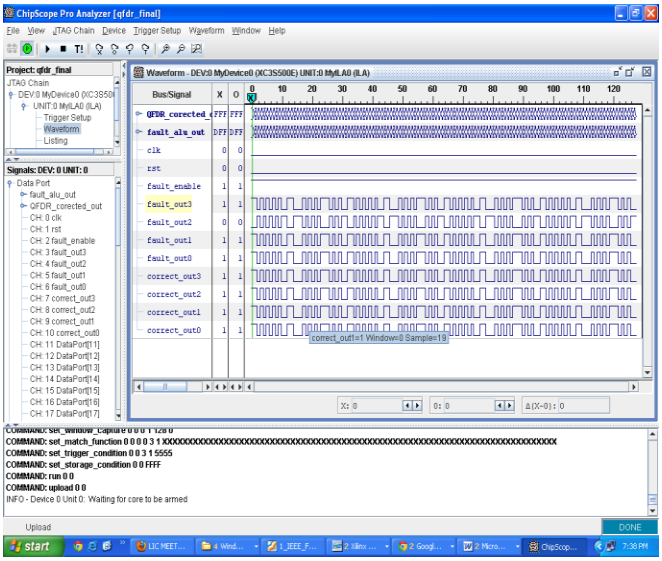
$$k_1 = f_1(i_1, i_3, j_1, j_3) = \begin{cases} f(i_1, j_1) & i_1 = i_3 \;\&\; j_1 = j_3 \\ 0 & otherwise \end{cases}$$

$$g_1(k_1, k_2, l_1, l_2) = \begin{cases} g(k_1, l_1) & k_1 = k_2 \;\&\; l_1 = l_2 \\ g(k_1, 1) & k_1 = k_2 \;\&\; l_1 \neq l_2 \\ g(1, l_1) & k_1 \neq k_2 \;\&\; l_1 = l_2 \\ g(1, 1) & k_1 \neq k_2 \;\&\; l_1 \neq l_2 \end{cases}$$

Now assume a failure occurs at input i1. It will not be equal to i3 anymore. So here set the output in this condition to a zero forced value. Two scenarios are possible. The first scenario is that f1 and f3 are forced to the default value, and it is equal to the correct output value which comes from f2 and f4.

Hence, the next level which contains function g will have four equal inputs from the previous level (f) and, accidentally, the forced value in this case was equal to the correct output value. Where the forced output value of f1 and f3 is not equal to the correct output of f2 and f4. Because the next level, including function g, knows about the forced output value of the previous level, it will select the correct result in case of difference.

Because all fine grain fault tolerant parts are small parts of the system, each of them can deal with one single failure without disturbance of the overall system. The probability of multiple failure occurrences in two redundant fine grains is much less than the coarse grain one. Multiple distributed failures affect the fine grains and the fine grains deal with the failures separately. This way the fine grain redundancy protects the system against multiple distributed failures.



3.3 ADVANTAGES:

- Voting mechanism is not used in the system. So huge area overhead is eliminated.
- Probabilities of occurrence of SETs (single event transitions) are less.

3.4 APPLICATIONS:

- Used in FPGA for Aerospace and space mission application.

- Used in applications where high reliable FPGAs are required.
- One of the most important applications, which SRAM-based FPGAs are increasingly in attention, is space based applications.

IV. CONCLUSION

SRAM-based FPGAs are increasingly in attention using in a space based applications.
Single event upsets have had an increasing influence in future FPGA architectures, particularly in domains such as space applications. We introduce methodologies based on fine grain approach to the fault tolerance problem. Here we used in fine grain fault tolerance as QFDR logic.

FUTURE SCOPE

Here in fine grain fault tolerance we are using QFDR instead of TMR.
Future work focuses on two key issues: Experiencing the fault tolerance technique in real radiation environments, and optimizing the additional area overheads, which arises from LUT partitioning.
This can be done by new CLB structure definitions for future FPGA generations

REFERENCES

1. J. Barth, K. LaBel, “Radiation assurance for the space environment,” in ICICDT ’04. International Conference on, 2004, pp. 323 – 333.
2. D. Mavis, P. Eaton, and M. Sibley, “Multiple bit upsets and error mitigation in ultra-deep submicron SRAM,” Nuclear Science, IEEE Transactions on, vol. 55, no. 6, pp. 3288 –3294, Dec. 2008.
3. M. Berg, https://nepp.nasa.gov/mafa/talks /mafa0734berg.pdf.
4. N. R. Effects and A. home page, in http://radhome.gsfc.nasa.gov.
5. N. R. J., R. P. A., K. K., S. W. J., M. P. T., and H. W., “Radiation characterization of a hardened 0.22 956;m anti-fuse field programmable gate array,” Nuclear Science, IEEE Transactions on, vol. 53, no. 6, pp. 3525–3531, dec. 2006.
6. C. M., G. P., J. E., and Wirt, “Single event upsets in sram fpgas,” Proc of MAPLD, Sep. 2002.