POLITECNICO DI TORINO

Faculty of Electonic Engineering

Master degree course in Electronic Engineering

Master Degree Thesis

Configurable fault tolerant Instruction Fetch stage for cv32e40p core

Based on cv32e40p core of OpenHW Group organization

Advisors
prof. Stefano Di Carlo

Correlatore:
prof. Alessandro Savino

Candidate:
Elia Ribaldone

Marzo 2021

To my family and Anna.

Acknowledgements

Abstract

The miniaturization of the microelectronic components together with the use of integrated circuits in more and more application leads to an increasing use of FT (Fault Folerant) architecture. This thesis investigate the use of FT techniques in a stage of the cv32e40p open source core. We used fault injection simulation to divide our stage in three blocks with increasing level of

Summary

A **FT (fault tolerant) system** continues to work properly even if some of the internal components are broken; this feature is necessary when a failure may cause damage to people, dangerous destruction, military upset or loss of data. FT systems are essential in aerospace, transport, medical and utility industries and they are usually composed by a power source, an hardware system and a software system, each of these parts are fault tolerant depending on application.

This work concerns the hardware system and in particular the chip architecture design. In this context the faults are **transient**, **intermittent** or **permanent** and they are generated by manufacturing defects, system degradation or particle strikes. **Manufacturing** defects generate permanent faults and they reduce the yield with an increase in the cost per piece since the affected chips are discarded during quality control process. **System degradation** produces permanent faults and it is a life-limiting phenomenon that brings the chip to wearout phase. Finally, **particle strikes** produce both transient or permanent faults. These problems rely on the application: system degradation generally depends on chip temperature, clock speed and the workload, otherwise particle strikes rely on sources of alpha particles or neutrons, which are generated by the cosmic rays or radioactive materials.

For these reasons an ideal fault tolerant system should be protected against transient faults caused by particle strikes and it should manage permanent faults in order to increase the yield and chip life. A complete protection against faults creates drawbacks in speed, area and power budgets. This is the reason why we create a configurable architecture where faults coverage can be changed according to the specific application and the project constrains.

In this Master Thesis the **Instruction Fetch of cv32e40p** core is converted in a **configurable fault tolerant stage** in order to reduce failures in fetching instructions. The core used was designed by the "OpenHW Group" organization and it can be integrated in PULPissimo platform in order to create a complete microcontroller architecture.

Before the design of the architecture we study the cv32e40p core and then we proceed with the creation of the simulation environment, building the script used for all simulations. These tools born in the cv32e40p *core-v-verif* repository with te purpose of automatize compilation of testbenches and their simulation using QuestaSim and Modelsim.

The most important feature of the tool is the **optimized fault injection method** used to simulate faults in the architecture. We use a worst case approach injecting faults only in sequential parts (FF and memory), in this way we consider that all faults injected in the combinatory path (for example after a particle strike) reach a FF and are sampled.

This is not always true since some bits can be logical masked in the following cases: if they are don't care bits when fault occurs, if they can be electrical masked due to attenuation before latch or if the fault don't have the time to reach a FF (latch-widows). All this masks can be clustered in AVF (Architecture Vulnerability Factor) which is 1 if every fault generate a failure, otherwise it is lower. Knowing this we can assert that our tool works in the worst-case scenario since the AVF of each combinatory path (excluding inputs) is consider equal to one.

Apart from this first considerations the tool optimizes simulation times also takes advantage of *vcdstim* feature in the case of single stage simulations. Indeed the whole core is initially simulated using a benchmark firmware, meanwhile the input and output data of a specific stage are saved into .vcd and .wlf files, finally a stage-specific simulation started using .vcd as input (*vcdstim* feature). In this way only one stage is simulated and we reduce working times. **Stage-specific simulation** can be repeated a specified number of times using fault injection and the output of the stage is finally compared with .wlf output file in order to find failures.

Using our tool we first simulate fault injection in the reference IF stage of cv32e40p core and we find a fault tolerance equal to 30%. Later we use simulation results to understand the fault masking for each signals and then we divide **IF stage** in **three main blocks**. *Each block have increasing level of intrinsic fault tolerance in original architecture* and knowing this we apply FT techniques to each block. Anyway during simulations and synthesis we can enable FT for one, two or all blocks, depending on these settings we can manage area, speed, power and FT trade-off.

In the design of architecture we use a FT technique that is able to detect and correct transient faults using **TMR** (Triple Modular Redundant), this methods works only if each of the three identical blocks compared don't have permanent faults, and if we assume to neglect multiple particle strikes. Although multiple strikes is improbable, permanent faults is already present after manufacturing process and increase during time, for this reason we implement a technique to detect and correct this faults using some backup stages. Summarizing *we use TMR to manage transient faults and additional logic to protect against permanent faults.* In this way if all FT architecture of the IF stage are enabled during syntesis the final result is an higher yeld, a longer life and a protection against noise and particle strikes. The last important feature of the design is the saving of *permanent faults information* in the **CSR** (Common and Status Registers), in this way we could restore permanent faults settings after a reboot.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

1.1 General context

1.2 Objectives

1.3 Thesis structure

Chapter 2

Technical Background And State of Art

Safety critical application system

2.0.1 Dependability Model

Dependability is the ability of a system to provide a predetermined level of service to the user [**Dubrova2013**]. This capacity depends on the system application, for example a wrong use or high workload make the level of service offered go down. From the designer's point of view, the dependability of a system must be verified through tests and simulations , in order to verify the correct functioning of the system in various environment. For system that works in critical applications, in addition to the functional tests must be made tests that verify the level of service required despite environment conditions. For example in satellites it is not possible to do maintenance and the correct behavior of on-board systems is necessary to avoid the fall of the asset, so when the Dependability required to the system is high, many stress tests must be done to have a complete technical testing. For these reasons to guarantee the dependability in a given application the main factors are how the system is designed and which kind of tests is performed on it.

Dependability in characterized by: Metrics, Attributes, Impairments and Means. These four categories allow us to completely define the dependability in a system and they are explained below:

Dependability Metrics Dependability metrics are used to measure the dependability of a system and they are used to verify Dependability Attributes. The Metrics are experimentally measured or estimated through various techniques. These are the main metrics used:

- ***TTF* :** Time To Failure is the time to a error in a specific system [**Mukherjee2008**]. For example a device with TTF equal to 1 year will have an error after one year of correct work.
- ***MTTF* :** Mean Time To Failure is the mean time between two failure in a system.

Under certain condition (e.g. formula 2.6) we can combine the MTTF of various parts to find the MTTF of overall system, to do this we should use the following formula:

$$MTTF_{system} = \frac{1}{MTTF_{part1}^{-1} + MTTF_{part2}^{-1}} = \frac{1}{\sum_{i=0}^{n_{parts}} \frac{1}{MTTF_i}} \tag{2.1}$$

- ***FIT*** **:** Failure In Time is the number of errors in a billion of hours. The relation between MTTF (expressed in year) and FIT is:

$$FIT = \frac{140000}{MTTF_{year}} \tag{2.2}$$

The FIT metric is used instead of MTTF because it makes calculation easier, in fact system FIT can be easly calculated in this way:

$$FIT_{system} = \sum_{i=0}^{n_{parts}} FIT_i \tag{2.3}$$

- ***MTTR*** **:** The Mean Time To Recover is the time needed to a system to repair an error once it is detected [**Mukherjee2008**].
- ***MTBF*** **:** The Mean Time Between Failure is the mean time between the start/restart and an error detection, for this reason we have:

$$MTBF \ = \ MTTF + MTTR \tag{2.4}$$

Dependability Attributes Attributes are the properties which are expected from a system that experiencing faults to be dependable [**Dubrova2013**]. These attributes are evaluated from Dependability Metrics according to a fault model. The most used Attributes are Reliability, Safety and Availability defined below:

- ***Reliability*** **:** it is the probability that a system will operate without failures in a given time interval. This type of Attribute is widely used for example in space applications, where it is necessary to guarantee operation for certain period. At the integrated circuit level many techniques have been adopted over time to increase reliability by improving production processes, usually are used old processes experiences to predict the reliability of a new product, this is done on all ICs but especially on memories [**An_Extended_Building-In_Reliability_Methodology_on_Evaluating_SRAM**]. Reliability can be expressed according to *exponential failure law* :

$$R(t) = e^{-h(t)\ t} \simeq e^{-\lambda\ t} \tag{2.5}$$

Where $h(t)$ is the *Instantaneous Error Rate* considered as the probability that the system has an error in a certain interval Δt which start at instant t, so it is the probability of error in the time interval $(t,\ t\ +\ \Delta t)$. To simplify calculation $h(t)$ is usually approximated with the constant error rate λ, that is equal to $1/MTTF = FIT$ [**Mukherjee2008**]. For these consideration when we have the FIT of each part of

a system we can use formula 2.5 to find total reliability, in this case we consider to have n independent parts each with a certain failure rate h_i:

$$R(t)_{system} = \prod_{i=0}^{n-1} R_i(t) = e^{-\left(\sum_{i=0}^{n-1} h_i\right)} \tag{2.6}$$

This model is valid if we consider the failure rate constant. From formula 2.6 we can states that the FIT of a system is equal to the sum of the FIT of each part.

- ***Availability*** **:** It is the percentage of time the system remains active and it can be used. This Attribute is employed a lot in the IT field, for example to characterize servers or a communication network [**Availability_requirement_for_a_fault-management_serv** [**Guaranteeing_High_Availability_to_Client-Server_Communications**]. It is therefore required in areas where it is expected that the system may not work for some periods, so in this case we are interested to know how long it will actually work properly. Availability is usually expressed as a percentage or by the downtime at a certain instant. For example, a system with Availability of 99.999% will have a downtime of 5 minutes over a year. The common expression for Availability is:

$$Availability = \frac{MTTF}{MTTF + MTTF} = \frac{MTTF}{MTBF} \tag{2.7}$$

- ***Safety*** **:** For this attribute, two types of failures are considered : *fail-safe* if the fail does not cause danger or damage, while *fail-unsafe* if the fail causes safety problems. A simple example is a RADAR that detects airplanes, if an airplane that doesn't exist is detected there is no serious damage and therefore we consider this failure as fail-safe, instead if an airplane is not detected we have a fail-unsafe failure. The safety of a system is the probability that it remains fail-safe over a certain period of time. It is used in critical sensing, safety and control systems.

Dependability Impairments Dependability Impairments are used to communicate that something in the system has gone wrong [**Dubrova2013**]. There are three types of Impairments and each indicates a problem at a different level:

- ***Faults*** **:** They indicate a problem at the physical level. For example in a PCB circuit a fault can occur when a component desoldered due to incorrect manufacturing process. In the field of integrated circuits a fault is usually due to a bit flip caused by external particles, by a manufacturing defect or a bug in the microcode or software. Any failure of a system always starts with a fault, this fault may or may not cause a problem depending on how the design was done. In integrated circuits faults can be masked by certain architectural design techniques and their number can be limited by special layouts and processes. However, they cannot be eliminated entirely.

- ***Errors*** **:** They indicate a problem at computational level caused by a Fault. Errors are caused by Faults that are not masked by the system, for example if there is a bit flip in an input register of the ALU, there will be an Error in the output register because the operation has a wrong result.

- ***Failures* :** They indicate system failure due to an Error. The failure of the system is an Impairments that you never want to have in a critical application since the behavior of the circuit is unpredictable and so unsafe.

To summarize a Fault can cause and Error and this can cause a Failure. For these reason the designer of a critical application system should have the ability to mask Fault and Errors in order to avoid Failure.

Dependability Means Dependability Means are that set of techniques and methods needed to create a Dependable system[**Dubrova2013**]. Fault Tolerance is the method that is used in this thesis but it is normally followed by other techniques, these are the most important ones:

- ***Fault Tolerance (FT)* :** Fault Tolerant systems continue to work even in the presence of Faults, this result is achieved through redundancy and a set of processes: The first is called Fault Masking and consists in avoiding the propagation of a fault by correcting the values in the system. In fact Fault Masking consists both in the reduction of errors and in their masking to avoid failures. Common examples of Fault Masking techniques are TMR (Triple Modular Redundancy) and ECC (Error Correcting Code) that allow to reduce Errors in memories and circuits. The second process is the Fault Detection that allows to recognize the presence of an error in the system, for example using the TMR in order to detect a Fault we can just verify that there is a module with different results from the others. This technique is also used in systems without redundancy where you want to understand if the system is working properly.

  When a fault is detected in a FT system, you can decide to correct it and continue with the execution, or you can disable the system part from which the fault started, in the case of permanent fault. This mode of performances decay of a system is called Graceful Degradation.

- ***Fault Prevention (FP)* :** FP is a very broad field because it is the set of processes that allow to reduce the introduction of faults in the system. This goal is achieved by controlling all processes from specification to manufacturing.

- ***Fault Forecasting* :** Fault Forecasting is the set of techniques that allow to predict the trend of the number of Faults and their effects in a system.

- ***Fault Removal* :** Fault Removal is the set of techniques used to eliminate errors already present in the system. This is done through verification of circuit operation and maintenance.

We have seen the basic vocabulary used in dependable system design and maintenance, in figure 2.1 are summarized all concept explained in order to give a graphical overview of the design of a Dependable System.

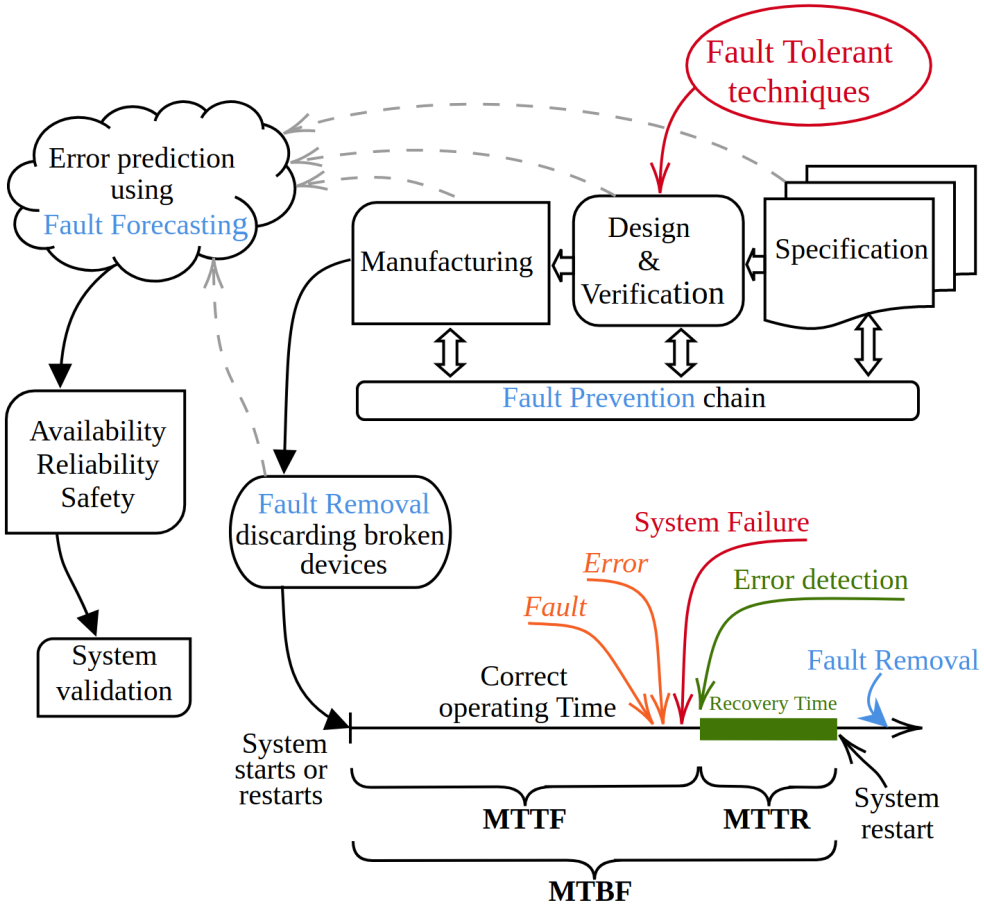

Figure 2.1: Design and life of a Dependable System

The block diagram in Figure 2.1 start with the specification of the system, then the designer use Fault tolerant techniques to design and verify the system, finally the product is manufactured, in these tree steps is applied Fault Prevention in order to reduce unwanted errors. After manufacture, the manufacturer apply a selection in order to discard broken devices and finally the systems is sold and it begins to be used. Meanwhile we gather data from all production chain in order to use Fault Forecasting to predict MTTF,MTTR and MTBF. Then using predicted data are evaluated required Dependability Attributes and finally system is validated and can be sold.

When the system begins to be used there are some periods of correct operations (estimated as MTTF), then at a certain instant a fault occur, this fault can propagate in an Error and this can became a System Failure. If the Failure is detected the system begins the Recovery Time (estimated as the MTTR) in which the failure is fixed. In the diagram we select a time interval in which fault is propagated but in a Dependable system this should happen rarely. It is also indicated the removal of defected parts using Fault Removal, this techniques can be also applied during Recovery time.

In the next section we contextualize this thesis work analyzing the parts of a critical electronic system.

2.0.2 Electronic system parts

This section describe how this Thesis is positioned in a complete dependable electronic system. In figure 2.2 we give an example of electronic system, it receives information from *sensors* and it controls some *actuators* according to their specification. The circuit is powered by a battery or by power network and this energy should be converted inside the board to be used. For this reason there is a part of the PCB dedicated to *voltage conversion*, this block is composed by analogue and digital components that together create the Power Conversion and Distribution system.

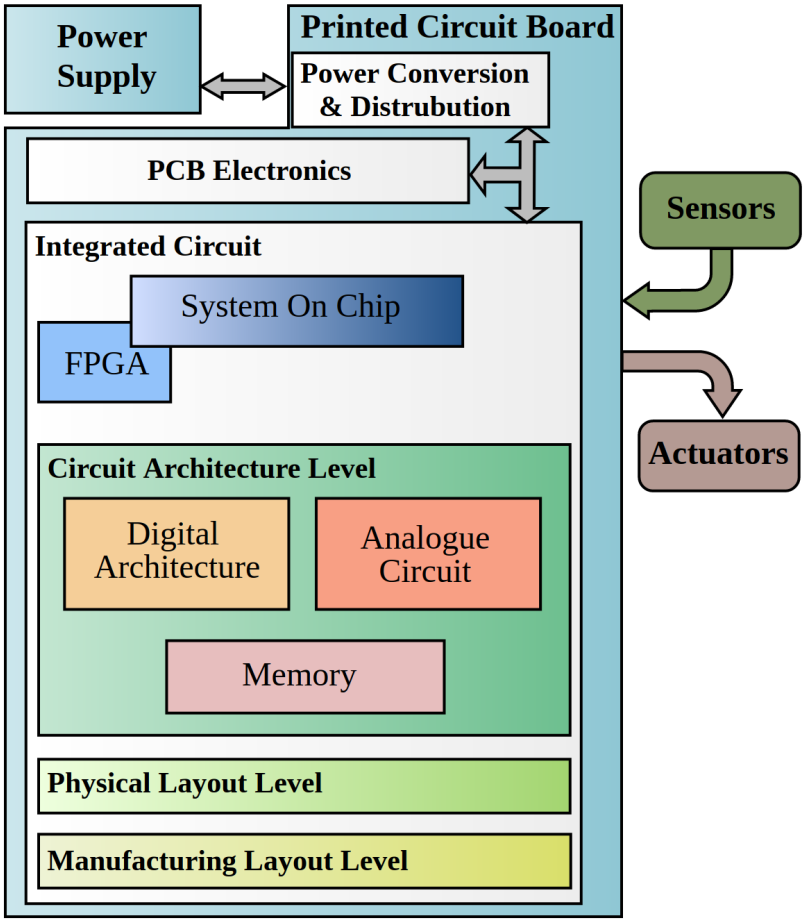

Figure 2.2: Example of Electronic System

The elaboration part instead is composed by integrated circuits that analyze the data received from analog and digital sensors and they use this data to decide how to control the actuators. This elaboration is done by a microcontroller or an FPGA and the design of these ICs have four main design level [**ECSS2016**] as you can see in Figure 2.2:

- **Manufacturing Process Level** (lev. 4) : This is the level of manufacturing processes, in this step are defined all technique to create the die from a silicon wafer.

In the case of hardened chip the manufacturer apply fault tolerant and fault prevention techniques in order to improve system dependability.

- **Physical Layout Level** (lev. 3) : It is the set of techniques used to place transistors properly. In the case of robust systems the layout is improved in order to decrease the sensitivity of the circuit to radiation.
- **Circuit Architecture Level** (lev. 2) : At this level circuits design is carried out at the RTL level; the circuits may be digital, analogue or a mixed signal. Generally to make this level robust are used fault tolerance redundancy and error correction techniques.
- **Electronic System Level** (lev. 1) : In this case we can still work at the RTL level using components previously created at the architectural level, or at the unit level (e.g. cluser computers). In the case of robust systems is used processor redundancy (e.g. lockstep technique) or redundancy of computers.

As we have seen, an electronic system is made up of many parts which must all be dependable in order to have a dependable system. *This Master Thesis will deal with the second design level, which is the architectural one.* In order to be able to use the proposed rtl project correctly, it is necessary to use hardening techniques in all the lower and higher levels. In fact what is important for the final application is the depandability of the system, so it would be almost useless to use a hardened processor in a device where the power supply part is not dependable.

2.0.3 IEC61508 Standard

2.1 Dependability of Integrated Circuits

2.1.1 Physical origins and mechanisms of faults

Faults in integrated circuits are due to both bit flip or electrical problems such as broken interconnects. The origins of these problems are due both to the aging of integrated transistors and their susceptibility to charge injection by external particles, such as cosmic rays.

These two phenomena are influenced by the field of use of the IC and by the working conditions. For example, aging is accelerated by high temperatures and high workloads, which wear out the interconnections. On the other hand the influence of external particles increases in space applications due to the increased cosmic ray flux, as well as in nuclear power plants or where some radioactive materials are present.

The understanding of these phenomena is essential to improve fault tolerance techniques applied to integrated circuits also at RTL level, therefore the causes and mechanisms of faults are now investigated by dividing them into *internal factors* (due to degradation) and *external factors* (due to particle flux or EMI).

Internal Factors of Faults As already mentioned, the internal factors of faults are due to electrical problems, which can be caused either by the breakage of the interconnections or by problems related to the gate oxide of the transistors.

As far as interconnections are concerned, there are two origins of failure:

- **Electromigration** (EM) : EM is a phenomenon known since 1966 [**EM1989**], whereby the electrons generating the electric current in the interconnections impart a momentum to the metal atoms. This momentum transfer can create void in the very small interconnections of ICs. The phenomenon is directly proportional to the square of the charge density (j_e; $[A/cm^2]$) and depends exponentially on the *activation energy* of the material (E_a; $[eV]$) and on the temperature (T $[K]$). In fact, the Median Time To Failure can be calculated according to the Black's formula [**Mukherjee2008**]:

$$MeTTF_{system} = \frac{A_0}{j_e^2} e^{\frac{E_a}{kT}} \tag{2.8}$$

  Where A_0 is a technology dependent constant and k is the Boltzmann constant.

  The opposite effect to EM is due to mechanical stress which tends to compensate for the displacement of metal atoms, this principle is the basis of the Blech effect for which below a certain length (called the Blech length) EM has no effect because the two forces are balanced. Normally the length of the interconnections is greater than the Blech length and for this reason EM is reduced by various techniques. For example, the use of metal alloys (Al+Cu, Al+Pd) or by creating *Bamboo Structures* that reduce the number of metal grains. In fact, the creation of a void in a connection starts at the interface between two or more grains of metal, where the mobility of the atoms is greater and this initial phenomenon leads to an avalanche effect which creates the final voids.

  Electromigration create both permanent or intermittent faults and leads the chip in the wear-out phase, it is related to current density that normally depends on workload so fault tolerant strategy that reduce the EM lead with resource multiplexing and oversizing.

- **Migration Stress** (MS) :

2.1.2 Fault classification

2.1.3 Masking

2.1.4 General Hardening strategy for IC

2.2 Hardening techniques for digital circuit architectures

2.2.1 Clock Protection

2.2.2 Logic and Arithmetic circuit protection

2.2.3 Memories protection

2.2.4 Combinational and Sequential circuit protection

2.3 Validation techniques for digital circuit architectures

2.3.1 Real life testing

2.3.2 Ground Accelerated Radiation testing

2.3.3 Analytical approach

2.3.4 Fault Injection (FI)

Chapter 3

Conclusion

Chapter 4

Articles Summary

4.1 A Co-Design Approach for Fault-Tolerant Loop Execution on Coarse-Grained Reconfigurable Arrays

Article: [**FT_adaptive_loop_execution**]
Year: 2015
This article create a hardware/software co-design configurable on reliability of application and on real time SER (Soft error rate). The techniques used are Coarse-Grained Reconfigurable Arrays (CGRAs), DMR and TMR.

4.2 A dependence graph-based approach to the design of algorithm-based fault tolerant systems

Article: [**ABFT_method_graph_based**]
Year: 1990
This article propose a two stage ABFT (Algorithm-based fault tolerance) system for computation intensive applications[**ABFT_method_graph_based**] ABFT is a method invented in 1984 in this article "Algorithm-Based Fault Tolerance for Matrix Operations" [**ABFT_method**], the basic method encode data at high level and then the algorithm work on this data producing an encoded output. The computation is distributed along many unit to enhance fault tolerance, the method was first applied to matrix operation which are the basis of many intensive calculation. ABFT method can detect and correct errors in many matrix operation, to do this many processors are needed [**ABFT_method**] . ABFT is a low cost CED (Concurrent error detection) scheme and fault location scheme [**ABFT_method_graph_based**]. ABFT is applied to: Multiplication, triangularization, fft, sorting, mesh array and hypercubes (in a hypercube processor each processor is the corner of a cube and can communicate only with n other corner). This paper present a method to synthesize ABFT system based on graph-theoretic model and the use of dependence graph. The basic idea of the graph theoretic model is to use a checksum in the

matrix for each processor, at the end of the computation the results are compared and a fault processor can be found.

4.3 A Methodology for Alleviating the Performance Degradation of TMR Solutions

Article: [**Alleviate_TMR_preformance_degradation**]

Year: 2010

Reliability degradation has increasing importance in faster and complex architecture due to sub 65nm technologies. In FPGAs the faults are more dangerous since can change design not just user data. TMR introduced by Xilinx ensures fault masking but increase chip area and power consumption, there is also a delay degradation due to redundancy that is catastrophic for critical application. This paper provide a method with reasonable balance between fault masking (F.M.) and performance/power overhead. Recent works use TMR in most sensitive subcircuits. In this letter, we introduce a software supported frame-work that provides a trade-off between the desired level of fault masking and the performance/power degradation due to hardware redundancy. This is done by removing TMR from non sensitive subcircuits. The methods consist in the application of TMR to all processor, then the core is P&R (Place and route) to FPGA and finally the temperature distribution is analyzed. Since temperature distribution give guidelines about where it is most likely faults to occur!! since the failure probability for a device region increases with temperature. So it is possible to eliminate redundancy from parts of the design that operate under low temperatures without affecting practically fault masking.Given the affordable level of errors for a design, our methodology guarantees to find the maximum redundancy that can be selectively removed from noncritical for failure regions of the device.

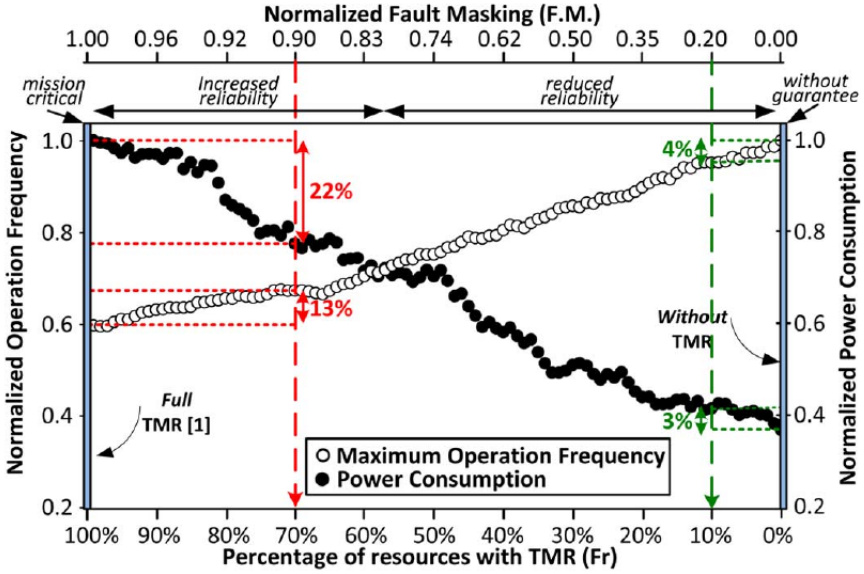

Figure 4.1: Different fault masking exploration

4.4 An Analytical Approach for Soft Error Rate Estimation in Digital Circuits

Article: [**SER_estimation_analytical**]
Year: 2005

Soft error or transient errors also called SEU (Single event upset) are caused by particles. SER (Soft error rate) is the error rate due to SEUs, which depends both from particle flux both to circuit characteristic. Normally memory is more susceptible to errors, but from 2011 combinational logic will be comparable due to scaling. This article propose the use of EPP (Error Propagation Probability) to estimate system failure due to soft errors. This method use the signal probability derived from power estimation calculus to derive a SER analytically. The complete algorithm is described. Final accuracy of 95% and 4/5 times faster then fault injection techniques.

4.5 A new analytical approach to estimate the effects of SEUs in TMR architectures implemented through SRAM-based FPGAs

Article: [**SEU_effect_in_TMR_analytical**]
Year: 2005

There are tree types of validation of an hardening techniques:

- accelerated radiation ground testing
- Fault injection
- Analytical approach

This paper propose and analytical approach. The main purpose of the proposed approach is to analyze the effects SEUs in both the user's memory elements and the FPGAs configuration memory early in the design phase.

4.6 Automated design flow for applying Triple Modular Redundancy (TMR) in complex digital circuits

Article: [**Automated_TMR_complex_digital_circuits**]
Year: 2018

The main problem is that generation of FT redundancy is not supported by EDA tools. There are some tools such as Xilinx TMRTool, Synopsys Synplify Premier, and Mentor Precision HiRel were launched in the market to apply TMR during the synthesis process. This article proposes an approach to automate the implementation, optimization, and verification of TMR circuits in commercial technologies. Three steps are added to the front-end design of ASICs. First, employing a post-synthesis netlist and according to the

desired granularity level the TMR technique is applied, three different TMR versions of the circuit can be implemented automatically. Afterwards, gate sizing is performed over the resulting circuit in order to improve performance. Third, equivalence checking is used to verify both correct functionality and fault-tolerant capability of the TMR circuit with regards to the original circuit. TMR is implemented using Cadence's synthesis tool Genus. The verification of FT architecture is done using Cadence's Logic Equivalence Checking (LEC). Finally simulation-based fault injections are performed to validate the effectiveness of the TMR implementation.

4.7 A voterless strategy for defect-tolerant nano-architectures

Article: [**Voterless_defect_tolerance**]

Year: 2008

This article explain how to use a reliable interconnect grid to create a voter for NMR hardening technique. It is an analog method!! The new communication mechanism is called LCDMA (Logic Code Division Multiple Access) and is implemented using an analog grid. Each signal that should be sent in the grid has to pass through a transmitter, the final signal should be analyzed to a receiver.

4.8 Combining Correction of Delay Faults and Transient Faults

Article: [**Combining_DelayF_and_TransF**]

Year: 2015

Delay fault are fault caused by an extra delay in high speed logic that create a fault like a Transient fault due to particle. This paper use previous works such as Razor and Bubble Razor architecture to create a combining approach that can correct both delay and transient faults.

4.9 Combining Fault Tolerance and Self Repair in a Virtual TMR Scheme

Article: [**Combining_FT_and_self_repairing_virtual_TMR**]

Year: 2013

This paper proposes a hardening architecture that allows both transient and permanent errors to be corrected using backup blocks. At startup, test vectors are used to understand the blocks with permanent errors and replace them by scoring the broken ones. In figure 4.2 you can see the time-sharing TMR.

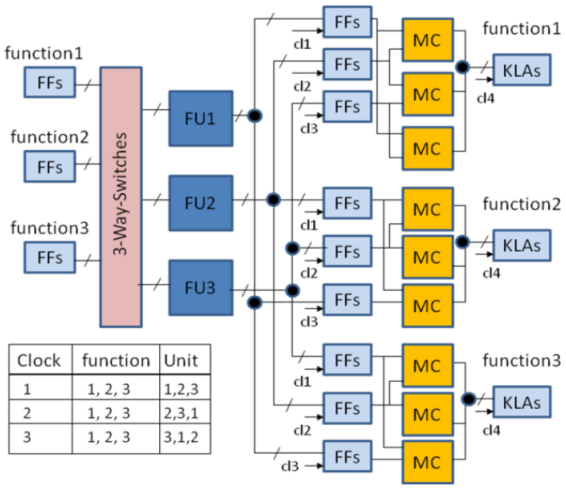

Figure 4.2: Time sharing TMR

The extension of TS-TMR instead is in figure 4.3.

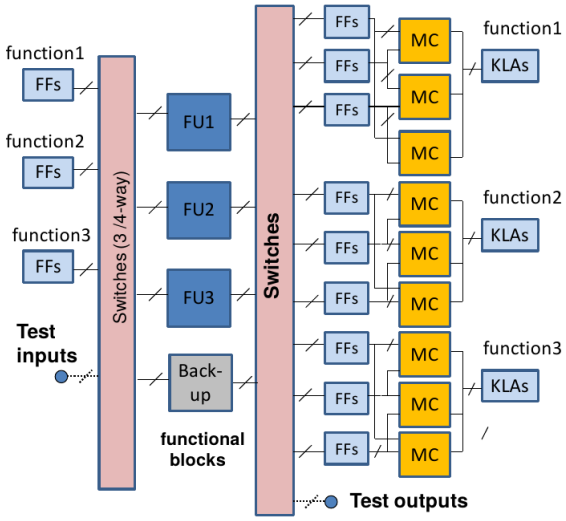

Figure 4.3: Time sharing TMR

In figure 4.4 you can see the Muller-C-element.

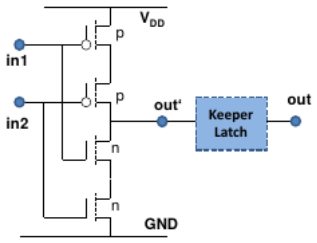

Figure 4.4: Muller-C-element

The proposed circuit has a 1/3 of the original throughput due to time sharing, the hardware is only one third of the original TMR. Also online self-repair is coupled with off-line self-repair.

4.10 Error Detection and Fault Tolerance in ECSM Using Input Randomization

Article: [**Error_Detection_and_Fault_Tolerance_in_ECSM_Using_Input_Randomization**]
Year: 2009

This paper discusses fault attacks on ECSM systems and proposes a technique based on parallel execution. This is similar to TMR but surpasses it by using the concepts of scaling and point randomization. This technique is more efficient and uses fewer blocks than TMR.

4.11 Evaluating the effectiveness of a diversity TMR scheme under neutrons

Article: [**Evaluating_the_effectiveness_of_a_diversity_TMR_scheme_under_neutrons**]
Year: 2013

The paper evaluates the DTMR (Diversity Triple Modular Redundancy) using an FPGA hit by a neutron flux of

$$3.98 * 10^4 n/cm^2/s$$

(standard deviation of

$$3.74 * 10^3 n/cm^2/s$$

) with an energy of 10MeV for 1268 minutes. The average number of upsets detected was about one per minute. It was found that the DTMR approach is better than TMR because each redundant block has a different reliability. In figure **??** you can see the DTMR implementation used in the paper.

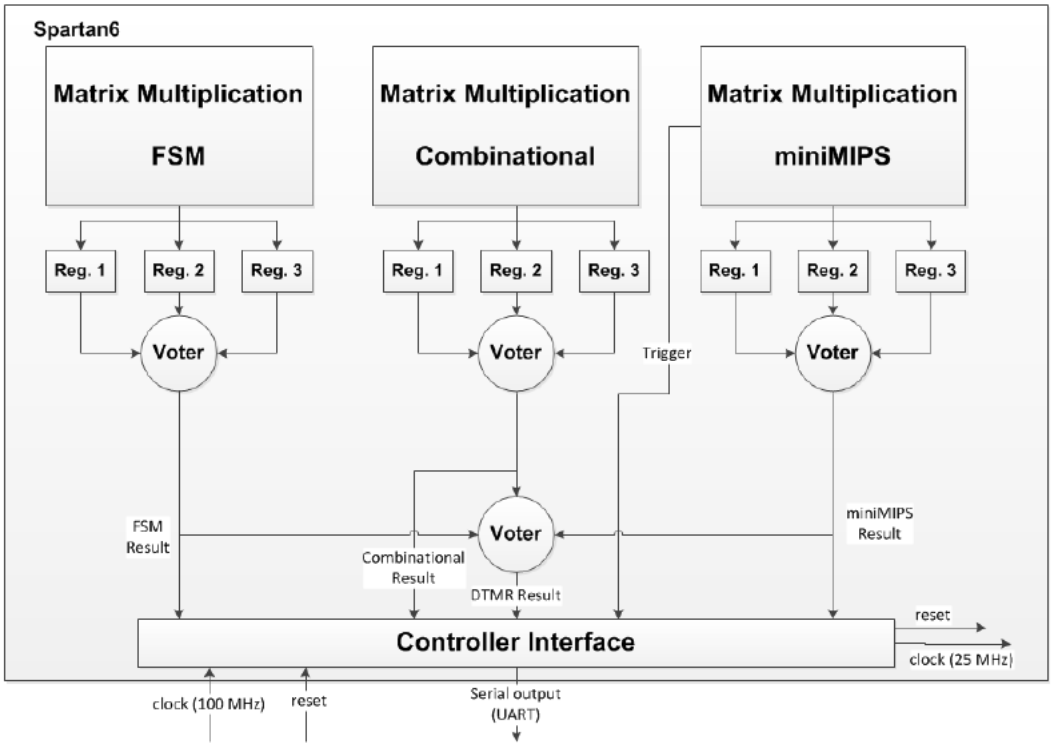

Figure 4.5: DTMR for matrix multiplication, tested using neutron beam

4.12 Fault and Soft Error Tolerant Delay-Locked Loop

Article: [**Fault_and_Soft_Error_Tolerant_Delay-Locked_Loop**]

Year: 2020

The paper presents a new method to create fault tolerance DLLs based on the previous FET-DLL technique. The FET-DLL technique consisted in using three DLLs and voting the output clocks, the problem was in the delay due to the voter called "output-lagging". To solve this problem dummy voter are added in the feedback network of DLLs. Thanks to this technique you can have a fault tolerant DLL without delays.

4.13 Hardware Error Correction using Local Syndromes

Article: [**Hardware_error_correction_using_local_syndromes**]

Year: 2017

This article presents a method called DSC (Duplication with Syndrome based Correction) applicable only to signal processing domains where the operations belong to the homomorphisms group. This method is the evolution of TMR but with a 32% simpler voter that allows an error correction very similar to that of TMR.

4.14 Highly-Reliable Approximate Quadruple Modular Redundancy with Approximation-Aware Voting

Article: [**Highly-Reliable_Approximate_Quadruple_Modular_Redundancy_with_Approximat**
Year: 2020

The paper proposes an approximate QMR redundancy method where are used three instances that approximate the true result and one that finds the exact one. A vote on the 3 approximate architectures is done first and then the result is voted with the exact architecture. The difference with the other works of approximate redundancy is in the use of approximators equal to each other while in the other works were used all different approximators.

4.15 Low-Overhead Fault-Tolerance Technique for a Dynamically Reconfigurable Softcore Processor

Article: molto lungo [**Low-overhead_fault-tolerance_technique_for_a_dynamically_reconfigurab**
Year: 2013

The paper proposes a reconfigurable softcore processor for FPGAs that outperforms previous ones in terms of hardware and time costs reduction. A Configuration Engine is created to identify processor faults and once a fault is identified it can be removed by partial hardware reconfiguration or dynamic reconfiguration. This article use an Enhanced Lockstep Scheme.

4.16 Low Overhead Soft Error Mitigation Techniques for High-Performance and Aggressive Designs

Article: [**Low_Overhead_Soft_Error_Mitigation_Techniques_for_High-Performance_and_A**
Year: 2012

Two types of cells are proposed to mitigate soft errors. The first is called SEM cell 4.6 4.7 and uses 3 clocks shifted between them to sample the input in 3 different registers . Register 1 is compared with register 2 to test an error, if there is an error the next block must recompute and the outputs of the other two registers are saved in register 1. The second cell is called STEM 4.8 (Soft and timing error mitigation) is similar but more complex because it allows to restore a previous state and redo the computation. The advantage of this solution is that there is no overhead during normal error-free operations, but we need to redo computation.

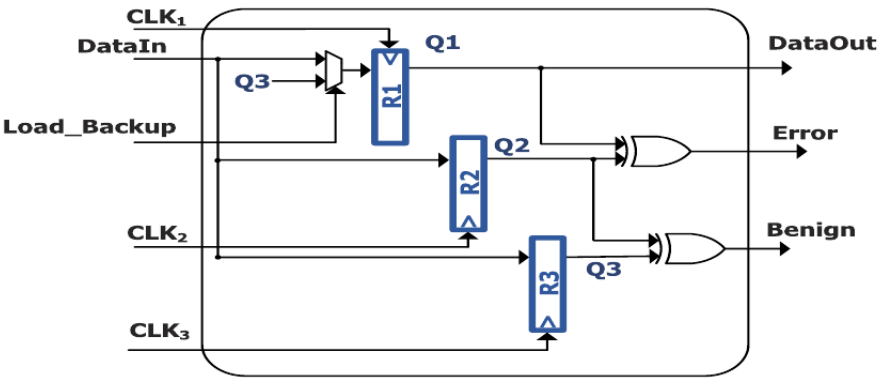

Figure 4.6: SEM cell

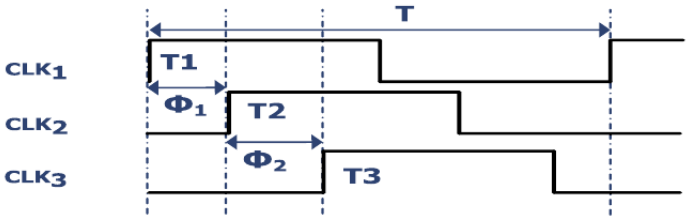

Figure 4.7: SEM cell timing

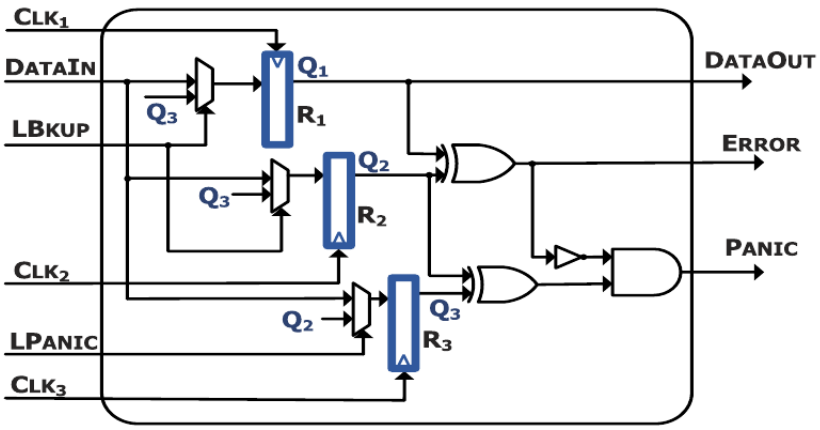

Figure 4.8: STEM cell

4.17 Optimization of a Cascading TMR System Configuration using Genetic Algorithm

Article: [**Optimization_of_a_Cascading_TMR_system_configuration_using_Genetic_Algori**
Year: 2012

This paper uses a genetic algorithm to find the best TMR configuration, the triplication of blocks or/and voters is done in order to optimize area and relability.

4.18 Resilient Hardware Design for Critical Systems

Article: [**Resilient_Hardware_Design_for_Critical_Systems**]
Year: 2019

This paper proposes a structure formed by 3 redundant blocks and a different one that has the same function, the outputs are compared and an FSM evaluates which signal to give an output through a multiplexer. Calculations are made on the reliability of the system and it seems to be better than TMR.

4.19 Scaling Analytical Models for Soft Error Rate Estimation Under a Multiple-Fault Environment

Article: [**Scaling_Analytical_Models_for_Soft_Error_Rate_Estimation_Under_a_Multiple-**
Year: 2007

The article starts from the analysis of Gurzi's method that allows to understand whether to use or not the TMR knowing the reliability of the voter and the circuit. Two methods are proposed to obtain the reliability of a circuit with TMR using the probability of the signals.

4.20 Towards Byzantine fault tolerant publish/subscribe A state machine approach

Article: [**Towards_Byzantine_fault_tolerant_publish_subscribe_A_state_machine_approach**
Year: 2013

The article discusses fault tolerance in the context of event-based interactions between systems. The BFT (Byzantine fault tolerance) pub/sub (publisher/subscriber) architecture based on an overlapping tree network is proposed.

Appendix A

An appendix