UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RODRIGO POSSAMAI BASTOS

Design of a Soft-Error Robust Microprocessor

Thesis presented in partial fulfillment of the requirements for the degree of Master of Computer Science

Prof. Dr. Ricardo Augusto da Luz Reis
Advisor

Porto Alegre, August 2006.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Bastos, Rodrigo Possamai

Design of a Soft-Error Robust Microprocessor / Rodrigo Possamai Bastos – Porto Alegre: Programa de Pós-Graduação em Computação, 2006.

120 f.:il.

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2006. Advisor: Ricardo Augusto da Luz Reis.

1. Fault-tolerant microprocessor. 2. Soft Errors. 3. SET. 4. SEU. 5. Integrated circuit design. I. Reis, Ricardo Augusto da Luz. II. Título.

CONTENTS

LIST OF ABBREVIATIONS

ASIC	Application Specific Integrated Circuit
BIST	Built-in Self-Test
CAD	Computer-Aided Design
CIF	Caltech Intermediate Form
CMOS	Complementary Metal-Oxide-Semiconductor
CPU	Central Processor Unit
CTLF	Compiled Timing Library Format
CWSP	Code Word State Preserving
DD	Displacement Damage
DEF	Design Exchange Format
DRC	Design Rule Check
EDA	Electronic Design Automation
EDAC	Error Detection and Correction
FPGA	Field Programmable Gate Array
GDSII	Graphical Design System II
HC	Hamming Code
HCMOS	High-density Complementary Metal-Oxide-Semiconductor
HDL	Hardware Description Language
IC	Integrated Circuit
IP	Intellectual Property
LEF	Library Exchange Format
LVS	Layout Versus Schematic
MBU	Multiple Bit Upset
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
PKS	Cadence Physically Knowledgeable Synthesis
PPGC	Programa de Pós-Graduação em Computação
PWM	Pulse Width Modulation

RAM	Random Access Memory
ROM	Read Only Memory
RSPF	Reduced Standard Parasitic Format
RT	Register Transfer
SDF	Standard Delay Format
SE	Soft Error
SE P&R	Cadence Silicon Ensemble Place-and-Route
SEB	Single Event Burnout
SEE	Single Event Effect
SEGR	Single Event Gate Rupture
SEL	Single Event Latchup
SER	Soft Error Rate
SET	Single Event Transient
SEU	Single Event Upset
SHE	Single Hard Error
SOC	System-on-Chip
SOI	Silicon-on-Insulator
SRAM	Static Random Access Memory
TCL	Tool Command Language
TID	Total Ionizing Dose
TLF	Timing Library Format
TCF	Toggle Count Format
VCD	Value Change Dump
TR	Time Redundancy
TMR	Triple Modular Redundancy
UFRGS	Universidade Federal do Rio Grande do Sul
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit
VLSI	Very Large Scale Integration

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

The advance of the IC technologies raises important issues related to the reliability and robustness of electronic systems. The transistor scale by shrinking its geometry, the voltage reduction, the lesser capacitances and therefore smaller currents and charges to supply the circuits, besides the higher clock frequencies, have made the IC more vulnerable to faults, especially those faults caused by electrical noise or radiation-induced effects.

The radiation-induced effects known as Soft Single Event Effects (Soft SEEs) can be classified into: direct Single Event Upsets (SEUs) at nodes of storage elements that result in bit flips; and Single Event Transient (SET) pulses at any circuit node. Especially SETs on combinational circuits might propagate itself up to the storage elements and might be captured. These erroneous storages can be also called indirect SEUs. Faults like SETs and SEUs can provoke errors in functional operations of an IC. The known Soft Errors (SEs) are characterized by values stored wrongly on memory elements during the use of the IC. They can make serious consequences in IC applications due to their non-permanent and non-recurring nature. By these reasons, protection mechanisms to avoid SEs by using fault-tolerance techniques, at least in one abstraction level of the design, are currently fundamental to improve the reliability of systems.

In this dissertation work, a fault-tolerant IC version of a mass-produced 8-bit microprocessor from the M68HC11 family was designed. It is able to tolerate SETs and SEUs. Based on the Triple Modular Redundancy (TMR) and Time Redundancy (TR) fault-tolerance techniques, a protection scheme was designed and implemented at high level in the target microprocessor by using only standard logic gates. The designed scheme preserves the standard-architecture characteristics in such way that the reusability of microprocessor applications is guaranteed. A typical IC design flow was developed by means of commercial CAD tools. Functional testing and fault injection simulations through benchmark executions were performed as a design verification testing. Furthermore, fault-tolerant IC design issues and results in area, performance and power were compared with a non-protected microprocessor version. The core area increased by 102.64 % to protect the target circuit against SETs and SEUs. The performance was degraded in 12.73 % and the power consumption grew around 49 % for a set of benchmarks. The resulting area of the robust chip was approximately 5.707 mm^2.

Keywords: fault-tolerant microprocessor, Soft Errors, SET, SEU, integrated circuit design.

Projeto de um Microprocessador Robusto a Soft Errors

RESUMO

O avanço das tecnologias de circuitos integrados (CIs) levanta importantes questões relacionadas à confiabilidade e à robustez de sistemas eletrônicos. A diminuição da geometria dos transistores, a redução dos níveis de tensão, as menores capacitâncias e portanto menores correntes e cargas para alimentar os circuitos, além das freqüências de relógio elevadas, têm tornado os CIs mais vulneráveis a falhas, especialmente àquelas causadas por ruído elétrico ou por efeitos induzidos pela radiação.

Os efeitos induzidos pela radiação conhecidos como Soft Single Event Effects (Soft SEEs) podem ser classificados em: Single Event Upsets (SEUs) diretos em nós de elementos de armazenagem que resultam em inversões de bits; e pulsos transientes Single Event Transients (SETs) em qualquer nó do circuito. Especialmente SETs em circuitos combinacionais podem se propagar até os elementos de armazenagem e podem ser capturados. Estas errôneas armazenagens podem também serem chamadas de SEUs indiretos. Falhas como SETs e SEUs podem provocar erros em operações funcionais de um CI. Os conhecidos Soft Errors (SEs) são caracterizados por valores armazenados erradamente em elementos de memória durante o uso do CI. SEs podem produzir sérias conseqüências em aplicações de CIs devido à sua natureza não permanente e não recorrente. Por essas razões, mecanismos de proteção para evitar SEs através de técnicas de tolerância a falhas, no mínimo em um nível de abstração do projeto, são atualmente fundamentais para melhorar a confiabilidade de sistemas.

Neste trabalho de dissertação, uma versão tolerante a falhas de um microprocessador 8-bits de produção em massa da família M68HC11 foi projetada. A arquitetura é capaz de tolerar SETs e SEUs. Baseado nas técnicas de Redundância Modular Tripla (TMR) e Redundância no Tempo (TR), um esquema de proteção foi projetado e implementado em alto nível no microprocessador alvo usando apenas portas lógicas padrões. O esquema projetado preserva as características da arquitetura padrão de tal forma que a reusabilidade das aplicações do microprocessador é garantida. Um típico fluxo de projeto de circuitos integrados foi desenvolvido através de ferramentas de CAD comerciais. Testes funcionais e injeções de falhas através da simulação de execuções de benchmarks foram realizados como um teste de verificação do projeto. Além disto, detalhes do projeto do circuito integrado tolerante a falhas e resultados em área, performance e potência foram comparados com uma versão não protegida do microprocessador. A área do core aumentou 102,64 % para proteger o circuito alvo contra SETs e SEUs. A performance foi degrada em 12,73 % e o consumo de potência cresceu cerca de 49 % para um conjunto de benchmarks. A área resultante do chip robusto foi aproximadamente 5,707 mm^2.

Palavras-Chave: microprocessador tolerante a falhas, Soft Errors, SET, SEU, projeto de circuito integrado.

1 INTRODUCTION

The constant technology evolution on the electronic circuitry has already been allowing the manufacture of integrated circuits (ICs) using semiconductors built with nanometer-scale features that near of the physics limits. Indeed, nowadays some popular microchips can be already called nanochips (HUTCHESON, 2004).

If on one hand, the evolutions allow expressive innovations on the engineering of designs optimized in area, performance and power and thus enabling the building of more sophisticated and complex electronic systems. On the other hand, the increasing importance, which ICs have been placing in many spheres of life activities, obliges them to perform their functional tasks within higher levels of safety and correctness. It is even more required in perturbed environments, where ICs are potentially more susceptible to errors.

In fact, the advance of the IC technologies has raised important issues related to the reliability and robustness of the electronic systems. The transistor scale by shrinking its geometry, the voltage reduction, the lesser capacitances and therefore smaller currents and charges to supply the circuits, besides the greater clock frequencies, have made the ICs more vulnerable to faults, especially those faults caused by electrical noise or radiation-induced effects. These scaling and technology issues of the Very Deep Submicron (VDSM) ICs reduce significantly their noise margins and thus their reliabilities regarding various internal or external sources of upset (LIMA, 2003-b; KASTENSMIDT; CARRO; REIS, 2006).

About radiation, the physics explains as the process of emitting radiant energy in the form of waves or particles. The **Soft Single Event Effects** (Soft SEEs) are caused specially by alpha particles (released by radioactive impurities) and, more importantly, cosmic rays (neutrons) hitting on the silicon chips and transferring charge to the circuit nodes with enough energy able to perturb its storage elements (BORKAR, 2005). These effects are classified in accord to the localization of the attacked node on the IC: direct upsets at nodes of storage elements causing alteration in their information as bit flips can be called **direct Single Event Upsets** (SEUs) (MASSENGILL et al, 2000); transient voltage fluctuations at any circuit node due to radiation-induced particles as well electrical noise are characterized as **Single Event Transients** (SETs) (KRISHNAMOHAN, MAHAPATRA, 2004). Especially SETs on combinational circuits are modeled like transient pulses that might propagate up to the storage elements and might be captured. It basically depends on the delays of the combinational network gates, on the widths of the created pulses and if these pulses meet the set-up and hold time requirements of the memory elements at a clock transition for storing. These erroneous storages can be also called **indirect SEUs**.

Faults like SETs and SEUs may provoke errors in functional operations of an IC. The known **Soft Errors (SEs)** are characterized by values stored wrongly on memory elements during the use of the IC and not due to design errors, fabrication defects or permanent physical failures. They can make serious consequences in IC applications as a result of their non-permanent and non-recurring nature (SHIVAKUMAR et al, 2002; KARNIK; HAZUCHA; PATEL, 2004). The increase in Soft Error Rates (SERs) on ICs has been a great source of concern for researchers in the last years. Some techniques have been developed to decrease the SER on ICs. Alpha particle flux has been gradually reduced by the use of purified materials. Fabrication process improvements in the 0.18 μm technology generation made the low-energy (lesser than 1 MeV) neutron SER negligible. Even though such techniques have reduced the SER, the high-energy (1 MeV to 1000 MeV) neutrons often dominate it in advanced CMOS logic (TOSAKA et al, 1998). Experiments, which replicate the sea level conditions for energies from 10 to 500 MeV, showed that the SER per bit of SRAMs in 0.25 μm, 0.18 μm, 0.13 μm and 90 nm technologies increases by 8% per generation (HAZUCHA et al, 2003). In addition, the situation is worse for ICs operating at flight altitudes or in space due to the even higher energies of the particles from there (LIMA, 2003-b). Years ago, studies related to the fault tolerance in semiconductor devices had larger developments especially for space and physics applications. Unlike today at which the concern is also focused at the debilities of circuits on terrestrial applications like servers and many embedded systems that usually have a large amount of embedded memory elements. By all these reasons, protection mechanisms to avoid SEs by using fault-tolerance techniques, at least in one abstraction level of the design, are currently fundamental. It improves the reliability and guarantees the correct operation of the systems. Several commercial microprocessors from AMD, Freescale, IBM, Intel and Sun are real implementations of robust systems by using detection and recovery techniques (IYER et al, 2005).

The current system complexities, the usual time-to-market and the project budget constraints have led designers to investigate fault-tolerance techniques and design flows more versatile. Reusable Intellectual Property (IP) cores developed at the higher abstraction levels of design, like the Register Transfer (RT) level, support engineers to faster cope with even more complex requirements such as System-On-Chips (SOCs). The reusability of hardware IPs and also software applications avoids redesigning and redeveloping the same features repeatedly and thus saves effort (i.e., development cost) and design time (HERRERA et al, 1999). Furthermore, the industry of EDA tools or CAD environments in the last years has been making easier and quicker the development of IC designs. Starting from higher design levels, EDA tools are able to provide very accurate estimated results of the IC design for a preliminary evaluation. On the other hand, making robust a system by using any fault-tolerance mechanism inherently involves additional overheads. There are many fault-tolerance techniques with different characteristics aiming different design levels, each one can be better adapted to a distinct design purpose. Therefore, a carefully preliminary analysis of the robustness features applied to the target system is mandatory before starting the design of the robust system. Moreover, a preliminary evaluation of the design costs before the IC manufacture is also fundamental through estimated IC design results such as in area, performance and power consumption.

Some requirements for the fault-tolerance implementation may implicate undesired modifications at standard characteristics of a system, especially when the target is the reusability of systems based on standard architectures like commercial microprocessors.

For instance, some typical fault-tolerance techniques require additional clock networks for fault detection and extra clock cycles for fault correction. In addition to the inherent cost of the fault-tolerance mechanisms, some consequences, which may be undesired, are the necessity for other clock signals and extra clock-tree implementations besides unexpected overheads at the execution time of its software applications. To save design time and development cost in a robust IC design, the chosen fault-tolerance techniques usually are desired not only to guarantee the reliability and reusability of their existing hardware and software applications. They are also desired to be easily or at least applicable at the target design level (for example, at the RT level) and that they adapt themselves to commercial standard cores.

Commercial microcontrollers like Freescale M68HC11, Intel 8051 and Microchip PIC are commonly mass-produced for electronic systems or embedded systems. Such systems have a wide range of applications in instrumentation, automation, control, telecommunication or even domestic appliances. These microcontrollers and their microprocessors are also largely used as cores or parts of SOCs. As these commercial circuits are consolidated in the market because they are simple and cheap, there are many systems and applications based on them. The utilization of these circuits allows the reusability of those already existing systems and applications. Consequently, design time and development costs can be saved. Typically, these commercial circuits are not prepared to operate under hostile environments. On the other hand, as circuits based on the new technology generations are more vulnerable to SEs. Thus, in the new manufactures, such commercial circuits trend to require some embedded fault-tolerance mechanism to guarantee their functionalities (i.e., to ensure the circuit reliability).

Another issue is that many commercial microprocessor systems such as some AMD, IBM and Intel architectures (LIMA et al, 2000-a, 2000-b; COTA et al, 2001; IYER et al, 2005) generally are protected against direct SEUs but not usually against indirect SEUs. The most commonly used mechanisms against SEs in modern processor are based on parity and Error Detection and Correction (EDAC) codes (IYER et al, 2005). Such techniques are essentially focused on protecting memory arrays and they usually do not mitigate indirect SEUs. Nevertheless, the scaling and technology issues tend to require protections against such faults too (SHIVAKUMAR et al, 2002). By this reason, many fault-tolerance techniques dedicated to mitigate indirect SEUs have been currently developed like those in (NICOLAIDIS, 1999; ANGHEL; NICOLAIDIS, 2000-a; KRISHNAMOHAN, MAHAPATRA, 2004; ZHANG; SHANBHAG, 2005).

The purpose of this dissertation work is to make robust to Soft SEEs or Soft Errors a commercial digital circuit, such as the 8-bit microprocessor from the microcontroller family M68HC11 (FREESCALE, 2002), for a future IC manufacture. In order to save design time, some initial design constraints were established. The fault-tolerant circuit design should be developed at high level like the RT level. The implemented fault-tolerance techniques should not use multiple clock networks. For any application, the techniques should preserve the total number of clock cycles, even so under a fault occurrence. Such initial constraints keep the standard-architecture characteristics and thus the reusability of microprocessor applications. In addition, they save development cost.

SETs on combinational circuits of the microprocessor, which can potentially cause indirect SEUs, are mitigated by using a Time Redundancy (TR) technique. The work in (NICOLAIDIS, 1999) suggests but does not implement a TR approach based on a special element called Code Word State Preserving (CWSP) like that from Figure 1.1

(a). Another work (ANGHEL; ALEXANDRESCU; NICOLAIDIS, 2000-b) evaluates this approach in area and performance by using simple test circuits, like adders and multipliers, and non-standard gates, such as that from Figure 1.1 (b), to implement the CWSP elements. In (LAZZARI; ANGHEL; REIS, 2005), the same evaluation is made for two microprocessors, MIPS and 8051, but a special automatic layout generator implements the non-standard gates that characterize the CWSP elements. In order to mitigate direct SEUs, in (LAZZARI; ANGHEL; REIS, 2005) a Triple Modular Redundancy (TMR) version that requires three clock signals was also implemented. In the present dissertation work was implemented a simpler and faster alternative to design by using only standard gates, like that from Figure 1.1 (a), and without an extra layout tool like that presented in (LAZZARI; ANGHEL; REIS, 2005). The defined initial design constraints are met through this alternative. The goal was to evaluate the costs in area, performance and also power and other design results of this fault-tolerance approach in the target microprocessor. In addition, the TR+CWSP elements and microprocessor registers were protected in accord to Figure 1.1 (c) by using a TMR version that require just one clock signal for mitigating direct SEUs.

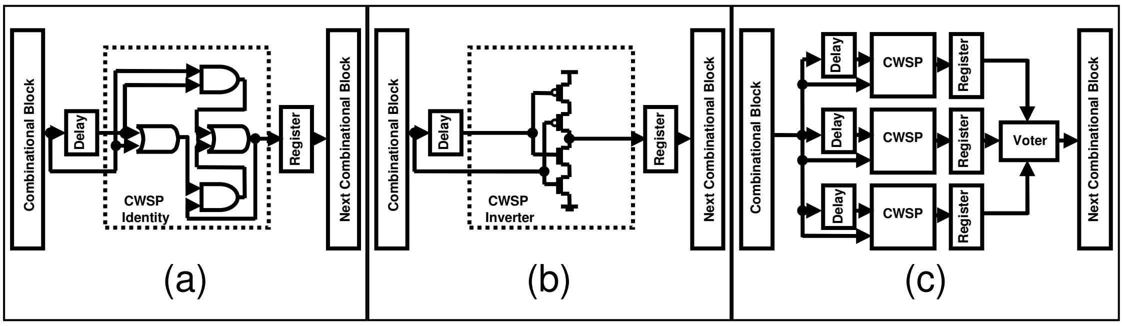


Figure 1.1: A TR+CWSP mitigation scheme by standard gates (a) and by non-standard gates (b). In (c), the TMR+TR+CWSP mitigation scheme that was used

The present dissertation is organized by chapters in the following way. Chapter 2 characterizes the target faults on integrated circuits by means of a radiation-induced fault model. Chapter 3 presents an overview about usual soft error mitigation techniques and details concerning the techniques implemented in the target microprocessor of this work. Chapter 4 introduces the target microprocessor to be protected and emphasizes the strategies applied in the robust microprocessor design. Furthermore, it shows the design steps from the microprocessor RT-level descriptions up to the GDSII stream files that are used to specify the physical design characteristics in an IC manufacture process. Chapter 5 presents the design verification simulation methods performed with the implemented microprocessor models in order to avoid design errors. In chapter 6, microprocessor design results by means of the circuit area, performance, power and other resulting information are analyzed. Some final remarks, conclusions and future works are discussed in chapter 7.

2 A RADIATION-INDUCED FAULT MODEL

Noxious effects on integrated circuits caused by internal or external sources of upset have been increasing due to the current scaling and technology trends. The direct consequences of these trends are smaller noise margins and thus circuits more vulnerable to external effects, like charged particles from different sources of radiation, as well to internal or external electrical noise. If a charge disturbance on a circuit node is smaller than the noise margin, the circuit will continue to operate properly. Otherwise, the disturbed voltage may be interpreted as the opposite logic state and the circuit will malfunction (KARNIK; HAZUCHA; PATEL, 2004).

Many different sorts of particles are found in environments where integrated circuits usually work. In space, particles from cosmic rays consist mostly of protons, but also of helium, oxygen and other ions (TOSAKA et al, 1998). At atmospheric and ground levels, alpha particles released by radioactive impurities in the device materials and mainly terrestrial cosmic rays in the form of high-energy neutrons are the major contributors for perturbations on circuit nodes (TOSAKA et al, 1998; BAUMANN; SMITH, 2000; BAUMANN, 2001; LERAY et al, 2004; BORKAR, 2005). Even so neutrons do not have electrical charges, their effects occur through nuclear collisions that give rise to charged particles. When such particles interact with the silicon atoms, they create a direct ionization in the semiconductor device causing transient currents that are able to make faults on the circuits. The amount of ionization and the current surge in a given semiconductor device are directly proportional to the energy lost by the radiation particles (KARNIK; HAZUCHA; PATEL, 2004; LIMA, 2003-b).

The type and the flux of hadrons like neutrons, protons and pions exhibit strong altitude and latitude dependence (NORMAND, 1996-a; CONSTANTINESCU, 2005). At sea level, the neutron flux is several hundred times lower than at aircraft altitudes. For instance, the neutron flux at 12 km (~40000 ft) altitude is around 300 times higher than at sea level and at 20 km (~60000 ft) it has its maximum peak. For this reason, integrated circuits operating at aircraft altitudes are more susceptible to faults induced by such particles than at ground level (NORMAND, 1996-b; GRANLUND; GRANBOM; OLSSON, 2003).

2.1 The Basic Radiation-Induced Effects on Integrated Circuits

The radiation-induced effects by means of their charged particles can cause different serious consequences on semiconductor circuits. Energetic particles incident on a solid lose their energy to ionizing and non-ionizing processes as they travel through a given material. The result of this energy loss is the production of electron-hole pairs (ionization) and displaced atoms (displacement damage) (SROUR; MARSHALL; MARSHALL, 2003). Especially three classes of these effects (Figure 2.1) are deeply explored by researchers due to their random natures and occurrence rates: Total Ionizing Dose (TID), Displacement Damage (DD) and Single Event Effects (SEEs).

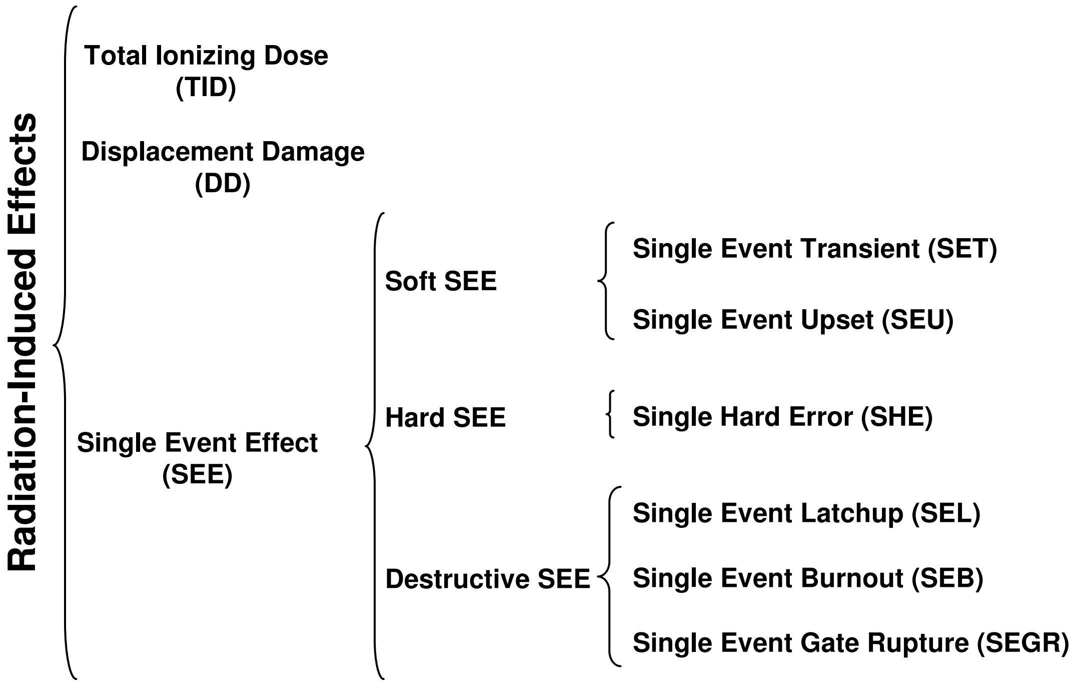

Radiation-Induced Effects
Total Ionizing Dose (TID)
Displacement Damage (DD)
Single Event Effect (SEE)
Soft SEE
Hard SEE
Destructive SEE
Single Event Transient (SET)
Single Event Upset (SEU)
Single Hard Error (SHE)
Single Event Latchup (SEL)
Single Event Burnout (SEB)
Single Event Gate Rupture (SEGR)

Figure 2.1: Main radiation-induced effects on integrated circuits

Total Ionizing Dose (TID) is due to long-term degradation of electronic circuits as a result of the cumulative energy deposited in some materials used by ICs. In space environment, significant sources of TID include trapped electrons, trapped protons and solar protons. Its effects include parametric failures or variations in device parameters like leakage current, threshold voltage, timing changes, etc (LABEL et al, 2000). These effects usually take a long time to occur, but they are permanent and can induce functional failures to ICs as putting out of use some of their functional blocks.

Displacement Damage (DD) is non-ionizing radiation effect that often has similar long-term degradation characteristics like TID. This effect leads to the degradation of material and device properties and is a consequence of the incident particles that displace atoms. The resulting defects give rise to new energy levels that alter the materials and devices in their electrical and optical properties. The effectiveness of radiation-induced DD depends basically on the defect rate and on the time exposure. Prime sources of DD include trapped protons, solar protons, neutrons, and in a lesser extent, trapped electrons (LABEL et al, 2000; SROUR; MARSHALL; MARSHALL, 2003).

Single Event Effects (SEEs) are due to transient physical faults such as single ions that impact on the circuit sensitive area. Sometimes, these events can deposit sufficient energy in the device that give rise to current pulses able to disturb the correct functionality of the system. Significant sources of SEEs include trapped protons, solar protons, neutrons and heavy ions from galactic cosmic rays. SEEs faults are composed by three distinct categories, depending on the consequences of the involved current pulse (LIMA, 2000-c; O'BRYAN et al, 1998):

- **Soft SEE**: during the operation of a device, a transient current pulse or a bit flip in its circuit can cause errors in its functionalities. Due to their non-permanent and non-recurring nature, these physical failures were called **Soft Errors (SEs)** (KARNIK; HAZUCHA; PATEL, 2004). They disappear when the system is reset or a data is rewritten in the memory. By this reason they can also be considered as intermittent events. Such errors are entirely device specific and are better categorized by their impacts on the device (LABEL et al, 2000). When a radiation-induced particle hits a node of a circuit, a **Single Event Transient (SET)** pulse can be created with enough energy to switch the node to a different voltage level. Indeed, SETs are characterized as transient voltage fluctuations on circuit nodes. They can be caused by radiation-induced particles as well electrical noise like noisy power supply, crosstalk noise, electromagnetic interference (EMI), radiation from lightning, etc (ZIEGLER et al, 1996; CALIN; VARGAS; NICOLAIDIS, 1995; MAHESHWARI; KOREN; BURLESON, 2003; KRISHNAMOHAN, MAHAPATRA, 2004). In a digital device, a **direct Single Event Upset (SEU)** occur when a storage element is directly affected by a SET in such way that it causes an undesired change on the memorized information as a bit flip (MASSENGILL et al, 2000). A SET pulse can be generated on a combinational logic circuit. Depending on the delay of the combinational gates and on the width of the pulse, it also can propagate up to the output of the combinational logic block. If the storage element succeeds to capture this undesired pulse, an **indirect SEU** is characterized;

- **Hard SEE**: hard errors are permanent functional effects to the device. An event of **Single Hard Error (SHE)** causes an undesired permanent change to a circuit node. A common example would be a stuck bit in a memory element (LABEL et al, 2000);
- **Destructive SEE**: events that can cause permanent physical destruction of the circuit. A **Single Event Latchup (SEL)** is the most common Destructive SEE. It is a potential destructive condition involving parasitic transistors on which currents might exceed their maximums specified. These parasitic transistors in the circuit can be activated by spurious currents, like those from radiation-induced effects. It would create a short between internal circuit nodes that may destroy the device by thermal effect, unless the power supply is removed. A **Single Event Burnout (SEB)** is a highly localized destructive burnout of the drain-source in a MOSFET. On the other hand, a **Single Event Gate Rupture (SEGR)** is the destructive burnout of a gate insulator in a MOSFET (LABEL et al, 2000).

2.1.1 Occurrence Rate of Radiation-Induced Effects

The SEE rates are not described as Mean-Time-To-Failure (MTTF). If an SEE rate is one per five years, it may happen at any time during that five year period with nearly equal probability. Otherwise, cumulative effects, such as TID or DD, the MTTF numbers are useful. The time-to-failure is the amount of operation time until the device has encountered enough degradation to cause failure (LABEL et al, 1996).

The high relevance of such effects can be supported by recent and frequent researches. At least since 1998, NASA's researchers have annually published at IEEE conference experimental results about the susceptibility of commercial and emerging technology devices to TID, DD and SEEs (O'BRYAN et al, 1998; COCHRAN et al, 2005; O'BRYAN et al, 2005).

This large concern, especially with the Soft Error (SE) occurrences on ICs, has resulted in a lot of researches for measuring, estimating and evaluating the **Soft Error Rate (SER)** of semiconductor devices. Many works characterize and evaluate the SE effects on ICs like those in (HARBOE-SORENSEN; SUND, 1992; VELAZCO; KAROUI; CHAPUIS, 1992; NORMAND et al, 1994; TOSAKA et al, 1998; ZIEGLER et al, 1998; BAUMANN, 2001; HOWARD et al, 2001; LIMA et al, 2001-a, 2001-b, 2002-a, 2002-b; DODD et al, 2002; MAIZ et al, 2003; KARNIK; HAZUCHA; PATEL, 2004; LERAY et al, 2004; LAMBERT et al, 2004; CONSTANTINESCU, 2005; SAGGESE et al, 2005).

Some real examples about SE evidence on ICs at ground level were discussed in (NORMAND, 1996-b). The computer system ACPMAPS at Fermilab is a very large system of individual computers, which when joined together, it contains about 160 Gbits of DRAM memory. This system is protected only by parity mechanisms. In a monitorial experiment, it had 2.5 upsets per day or a SER of $0.7 \cdot 10^{-12}$ upset/(bit·hour). Another case, 58 off-the-shelf Nite Hawk computers were monitored. Each computer is constituted by 1 Gbits of DRAM protected by EDAC codes. On the average, each machine showed around one upset per month (assigned as 624 hours), which was equivalent to a SER of $1.6 \cdot 10^{-12}$ upset/(bit·hour).

Additionally, there is a set of other works related to the SE effects on ICs at ground, atmospheric and space levels (NORMAND; BAKER, 1993; LABEL et al, 1996; NORMAND, 1996-a; BARTH, 1997; LABEL et al, 2000; NORMAND, 2001). Others show the SER increase in the new technology generations of ICs due to the scaling and technology trends (HAZUCHA el al, 2003; GRANLUND; GRANBOM; OLSSON, 2003; BORKAR, 2005).

2.1.2 Modeling of Radiation-Induced Effects

The radiation-induced effects on an integrated circuit can be modeled at different abstraction levels of the circuit design. In accord to (ABRAMOVICI; BREUER; FRIEDMAN, 1990; SMITH, 1997; WAGNER, 2004), the usual design levels from the lowest to the highest are classified into:

- **Real circuit level**, the circuit prototypes or the circuit fabricated by physical materials from a fabrication technology;
- **Electrical level**, the circuit layout mask at a geometric axis or the circuit models based on transistors, resistors, capacitors and inductors at a structural axis. Some authors consider the layout mask issues as part of the labeled **physical level**, even so such label is also used for the real circuit level. In addition, others define **switch level** as transistors modeled discretely and **transistor level** as transistors characterized by non-linear models;
- **Logical level**, the circuit models at a structural axis based on flip-flops, latches and logic gates, besides library cells at a geometric axis. EDA tools usually label the model of logic gates as **gate level**;
- **Micro-architectural level** or the well-known **Register Transfer (RT) level**, the circuit models at a structural axis based on registers, multiplexers, operators like adders, subtracters, multipliers and dividers, besides macro cells at a geometric axis. Some authors label this level as **behavioral level** or even **functional level** in accord to the delay model that is used;

- **Algorithmic level**, circuit models at a structural axis based on hardware modules. Modules, cores, plans of power, ground and clock at a geometric axis;
- **Systemic level**, circuit models at a structural axis based on processors, memories and other peripherals. Components and boards at a geometric axis.

As defined in chapter 1, the goal in this work is to make robust against radiation-induced effects a digital system such as a Central Processor Unit (CPU). A digital system denotes a complex digital circuit. The complexity of a circuit is related to the abstraction level required to describe its operation in a meaningful way (ABRAMOVICI; BREUER; FRIEDMAN, 1990). Typically, highest abstraction levels are used to design complex circuits because they provide a better management for designers. In this way, digital system designs usually require high-level abstraction resources like the VHDL, in which the lowest abstraction level that designers can deal with is the logical level.

At the abstraction logical level of a digital circuit, a further distinction can be made between combinational and sequential circuits. Unlike a combinational circuit, whose output logic values depend only on its present input values, a sequential circuit can also remember past values and hence it processes sequences of logic values (ABRAMOVICI; BREUER; FRIEDMAN, 1990).

In such circuit design abstraction, the radiation-induced effects feasible to be treated are those in which their tolerance mechanisms are able to be implemented at least at the logical level. The Soft and Hard SEEs are feasible. However, Destructive SEEs, TI and DD are typically treated at lower abstraction levels.

As emphasized in chapter 1, this work focuses on the Soft SEE on ICs. The serious effect of such physical fault can be modeled at the logical level as a bit flip. As seen in the last sections, it is an undesired change on the memorized information of storage elements or in other words a Soft Error (SE). At the logical level of an IC, storage elements are sequential circuits or memory cells such as flip-flops or latches.

This problem of Soft SEEs on ICs can be summarized based on the traditional definitions of fault, error and failure for a computer system (LAPRIE, 1998). The transient current pulse caused by a source of upset on a combinational or sequential circuit is a system **fault,** the bit flip on the memory cell is a system **error** and the reading of wrong values stored in the register is a system **failure**.

From the Soft SEEs, the sequential elements (memory cells) can be affected by direct or indirect Single Event Upsets (SEUs). In following sections, such effects and their characteristics are modeled and discussed.

2.1.2.1 Direct SEUs

A direct SEU is modeled as a logic perturbation or a direct logic inversion on a bit memorized by a sequential element. To illustrate such fault at the logical level, firstly the ideal timing behavior of a memory cell such as a positive edge-triggered flip-flop is shown in Figure 2.2 (a). After, Figure 2.2 (b) shows the timing behavior of the memory under a direct SEU.

Note in Figure 2.2 (a) that the memory input is switched from 0 to 1 at an instant before the clock event. This instant need respect the set-up time, thus such switch can not occur within a set-up time before the clock event. In the same way, the memory input need be kept on that value (value 1) at least a hold time after the clock event. Thus, the memory output switches logically from 0 to 1 and such value 1 is kept in the memory.

On the other hand, in Figure 2.2 (b), a direct SEU makes an inversion from 1 to 0 at the memory output (i.e., a SE) without any input or clock event. Note that the SE might be transient if new events occur. The memory output will be kept on 0 until new input switch (respecting the set-up and hold time requirements) or even until new SEU.

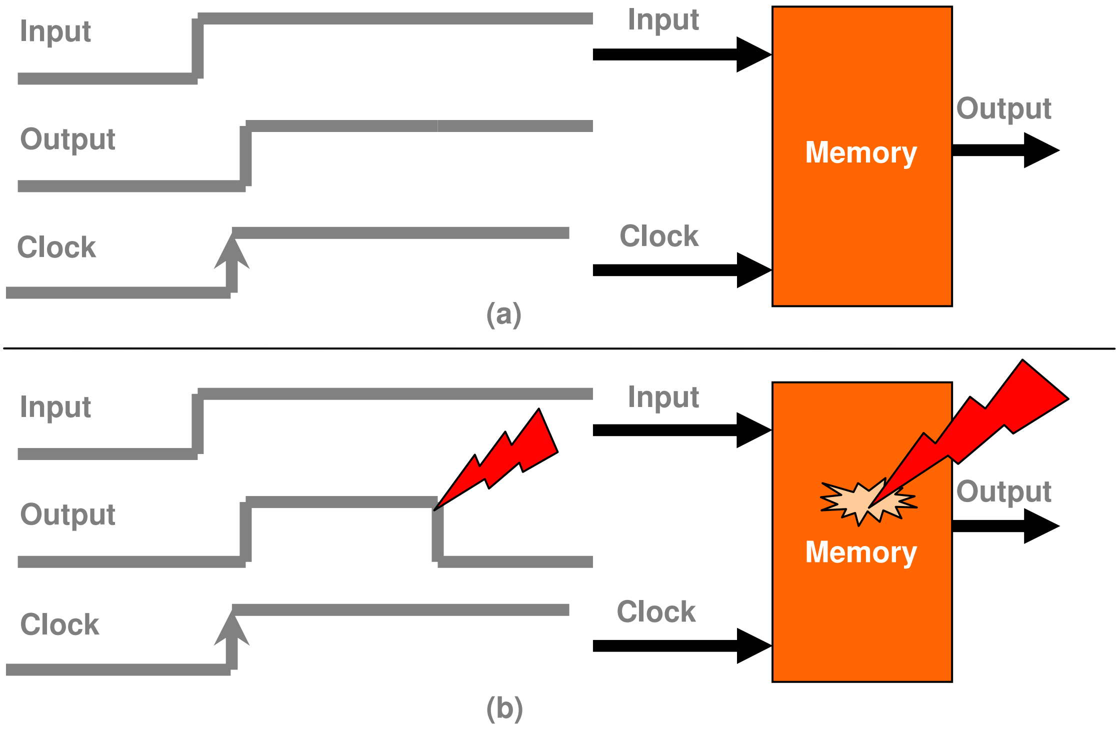


Figure 2.2: The timing behavior of a memory cell without SE (a) and with SE (b)

2.1.2.2 Indirect SEUs

An indirect SEU is due to a Single Event Transient (SET) modeled as a rectangular transient pulse that occurs on a combinational circuit and propagates itself up to a sequential element.

To illustrate such fault at the logical level, initially the ideal behavior of a sequential element (flip-flop) and a combinational circuit by means of its logic gates are presented in Figure 2.3 (a). After in Figure 2.3 (b), the fault is characterized.

Observe in Figure 2.3 (a) that the combinational circuit processes properly the values 0 at its three inputs. In addition, the sequential element memorizes appropriately the value 0 from the resulting combinational circuit output.

On the other hand, in Figure 2.3 (b), a SET occurs on a gate of the combinational circuit. The SET succeeds in propagating up to the output of the combinational circuit that is temporarily switched to 1. It hypothetically occurs at an instant and lasts enough time to meet the requirements of the set-up and hold times. Thus the undesired value 1 is memorized as an indirect SEU and a SE is characterized.

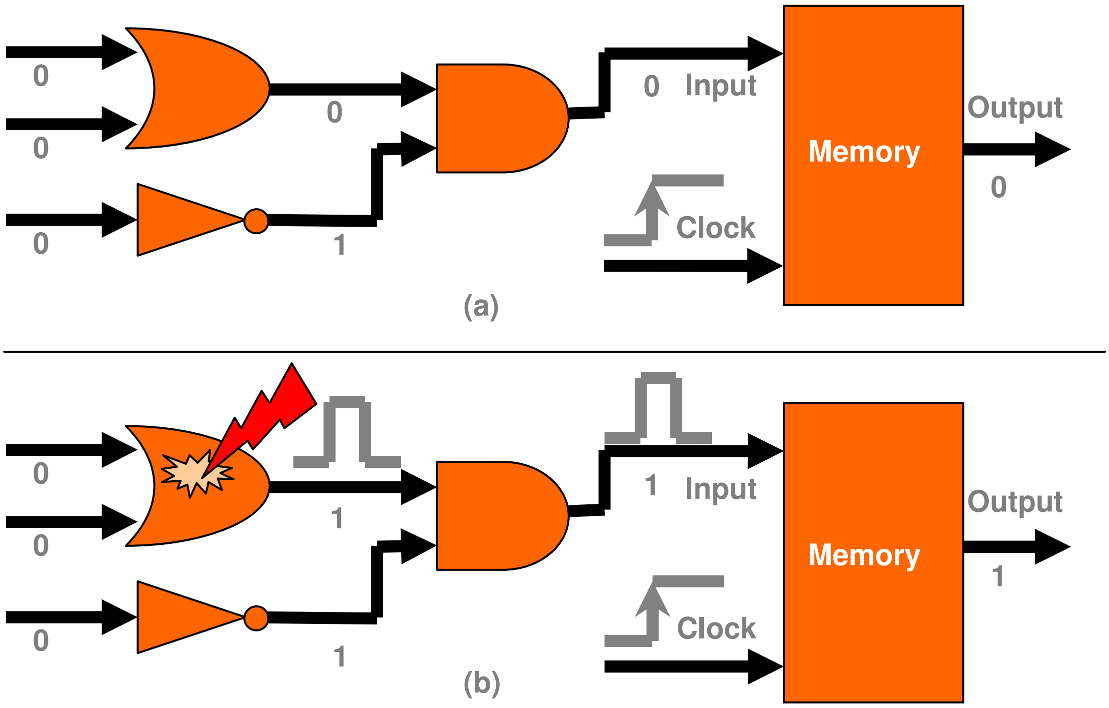


Figure 2.3: A combinational circuit without SETs (a) and with a SET (b)

2.1.2.3 SET Issues

A SET on a combinational circuit does not always give rise to an indirect SEU, it may not cause any unfavorable consequence to the IC. Such masking effect can be due to one of the following factors (SHIVAKUMAR et al, 2002):

- **Logical Masking**: a SET does not propagate up to output of the combinational circuit because makes some combinational logical operation that masks it;
- **Electrical Masking**: a SET is sufficiently attenuated due to the electrical properties of gates in the propagation path. In fact, if the duration of a SET pulse is larger than the propagation time (logic transition time) of a gate, it typically should not be attenuated. However, when a SET width is lesser than the propagation time of a gate, it starts to be slight attenuated and usually when it is lesser than half of the propagation time, it is sufficiently attenuated (NICOLAIDIS, 1999);
- **Latching-Window Masking**: a SET reaches the input of a sequential element, however does not meet the time window such as the set-up and hold times, which is required for the circuit latching a value at the clock event.

Such three masking phenomena provide the combinational circuits a form of natural resistance to SEs (SHIVAKUMAR et al, 2002). Past research has shown that combinational logic is much less susceptible to allow SEs than memory elements (LIDÉN et al, 1994; GAISLER, 1997). The memories always were considered most vulnerable to SEs due to their spatial density and the amount of information that they store (MAHESHWARI; KOREN; BURLESON, 2003). However, in the current decade as a result of the current nanometer technologies and the consequent high complexity of the integrated circuits, the SER arisen in combinational circuits shall become as relevant as the SER in sequential elements. In the work (SHIVAKUMAR et al, 2002) were analyzed the trends in the SER for SRAM cells, latches and combinational circuits. It predicts that by 2011 the SER arisen in combinational circuits will be comparable to that of unprotected memory elements.

The timing nature of a SET pulse generated on a circuit, especially on the combinational parts, depends on the energy of the perturbation event, on the employed physical technology and on the design topology of the circuit. The work (ANGHEL; NICOLAIDIS, 2000-a) generically presumes the duration of a typical SET pulse at few hundreds of picoseconds. Currently, such order of SET width is common for micrometer (channel length above 0.1 µm) nearly nanometer (below 0.1 µm) technologies. It can be easily found in many related experiments and works discussed in dedicated conferences such as the (SEE SYMPOSIUM, 2006).

Since pulses wider than the logic transition time of a gate usually can propagate itself without attenuation. For circuits based on micrometer technologies in which typical delays of basic standard logic gates can be around 10 ps, a SET of width around 100 ps can diffuse itself through gates, reach sequential elements and make SEs. On the other hand, in nanometer technologies, the propagation time of gates can be even smaller than the SET duration. In this way, even SET pulses due to perturbations of lower energy, therefore smaller SET widths, might not be attenuated. Furthermore, as the clock frequencies have increased significantly, the probability of latching a SET have also increased. In fact, as more frequent are the latching edges of the clock, higher is the probability to have a SET coinciding with a latching edge (ANGHEL; NICOLAIDIS, 2000-a).

Another critical characteristic of a SET pulse is when it occurs on an internal node of a combinational circuit with a certain width. After propagating through some combinational circuit paths, it can result wider at the output of the combinational circuit (input of a sequential element). Such occurrences make difficult the prevision of the maximum width for a fault-tolerance implementation. This phenomenon is essentially due to reconvergent fan-outs with different delays (ANGHEL; ALEXANDRESCU; NICOLAIDIS, 2000-b). The original pulse can propagate itself through several paths which reconverge and concatenate several pulses into a single one. This pulse can be larger than the original one due to the different delays of the propagation paths (ANGHEL; ALEXANDRESCU; NICOLAIDIS, 2000-b). In the work (NICOLAIDIS; PEREZ, 2003) is proposed a circuit that can measure experimentally the SET widths.

A unique SET pulse arisen in a combinational circuit can sometimes also generate multiple pulses at the output of this circuit as a result of the delay differences among its paths. Nevertheless, by considering a balanced circuit, such multiple events are rare. It is because those delay differences must not exceed the width of the original pulse arisen in the combinational circuit. Modern logic synthesis tools and architectural solutions trend to generate balanced circuits. This kind of circuit has the delays of its paths close to the delay of its largest path (NICOLAIDIS, 1999; ANGHEL; ALEXANDRESCU; NICOLAIDIS, 2000-b).

In addition, depending on the topology of the circuit, a unique SET can also cause a unique or several indirect SEUs. Several SETs can also occur at the same time on any bit of a combinational or sequential circuit, at any moment during the use of the IC.

By reason of all these behaviors of the SET pulses, their evaluations become very complex in circuits composed by many paths. Some works dedicate special attention to analyze the probability of a SET becoming an indirect SEU (HASS et al, 1998; HASS, 1999; MASSENGILL et al, 2000; ALEXANDRESCU; ANGHEL; NICOLAIDIS, 2002). Other approaches like a SET propagation method based on topological timing analysis (NEVES et al, 2006-a, 2006-b) could be used to evaluate such probability too.

2.1.2.4 Multiple SEUs

When multiple indirect or direct SEUs happen at the same time on bits of memories, it is traditionally called of **Multiple Bit Upsets (MBUs)**. According to the number of upsets that occur at the same time in the circuit, bit upsets can be classified in first, second and third order effects. A single bit upset (SEU) is classified as a first order effect, while multiple bit upsets (MBUs) are classified as second or third order effects (LIMA, 2003-b). MBUs can occur when:

- A single particle hits two adjacent nodes, located in two distinct memory cells. This event is classified as a second-order effect and can be avoided by specific placement design;
- A single particle strikes two adjacent nodes located in the same memory cell. This event is classified as a third-order effect and can be avoided by physical layout constraints for separating critical nodes;
- Multiple particles strike the circuit causing upsets in multiple nodes. These events can be considered as a group of direct SEUs;
- A unique SET from a combinational circuit result in multiple indirect SEUs;
- Several SETs from a combinational circuit result in multiple indirect SEUs.

3 SOFT ERROR MITIGATION TECHNIQUES

The evolution of scaling down technology has raised relevant issues related to the reliability and robustness of circuits. Reliability is normally defined as the immunity of a circuit to faults like, for instance, those that cause Soft Errors (SEs). Design robustness is defined as the ability of a circuit to operate correctly under varying process, temperature, voltage, and noise conditions (KRISHNAMOHAN, MAHAPATRA, 2004).

In order to improve the reliability and guarantee the correct operation of systems, robustness mechanisms to mitigate SEs through fault-tolerance techniques, at least in one abstraction level of the IC design, are currently much used at the industry. Several commercial microprocessors from AMD, Intel, IBM, Freescale and Sun are real implementations of robust systems. As examples, there are processors from the families: Intel P6, AMD Hammer, Intel Itanium, IBM G5 and IBM Power 4. These microprocessors use typically Error Detection and Correction (EDAC) codes and parity focused on protecting memory arrays (IYER et al, 2005).

Many other fault-tolerance techniques were already proposed for protecting ICs. The inherent cost of the robustness can vary depending on the chosen technique. The extra cost can be evident as in area and power as in performance. The manufacture cost of the IC might also be higher when a specific robust technology is used. Each technique due to its different characteristics might attend to many design objectives, therefore a detailed selection of that fault-tolerance technique to be implemented on the target system should always be done before starting the design of a robust IC.

There are fault-tolerance techniques for all design levels. They can be classified into low-level and high-level techniques.

The **low-level techniques** involve specially those techniques applicable or developed at the physical, electrical, switch or transistor levels. Such techniques usually are based on a specific technological process like Silicon-On-Insulator (SOI) or the package shielding; transistor sizing; robust memory cells; or a combination of them.

Otherwise, **high-level techniques** are those able to be used at the logical or gate; RT or micro-architectural; algorithmic; or systemic levels. They are typically based on hardware or software redundancy like Triple Modular Redundancy (TMR); Time Redundancy (TR) in hardware or software; self-checking circuits; parity; EDAC codes like Hamming Code or Reed-Solomon Code; or even a combination of them.

Robust memory cells are suggested in (CALIN; NICOLAIDIS; VELAZCO, 1996; ZHANG; SHANBHAG, 2005) and many other works. A version of TR is proposed in (KRISHNAMOHAN, MAHAPATRA, 2004) modifying only the CMOS flip-flop in such way that it samples and latches its data input at different instants within a clock cycle. A fault-tolerance technique dedicated to FPGAs is presented in (LIMA; CARRO; REIS, 2003-a). Many concurrent checking schemes (self-checking circuits), as presented in (ANGHEL; NICOLAIDIS, 2000-a), combined with a retry procedure had already been discussed. The tolerance to SE can be achieved by a retry operation after the detection of an error. Several other design solutions for tolerating SE were proposed in (NICOLAIDIS, 1999), in which the idea is taking advantage of the temporal nature of transient faults and mitigating them by using TR.

Techniques based on TR avoid the large hardware overheads of hardware redundancy, since the same operation is computed multiple times on the same hardware (IYER et al, 2005). Nevertheless, they usually incur high performance overhead and also require additional blocks for collecting and comparing the multiple execution results.

Techniques based on any type of redundancy can fail in case of multiple faults affect the redundant parts at the same instant. However, these multiple faults usually have lower probability of occurrence. Triple or higher redundancy usually obtains a correct a correct answer through a majority-voting scheme (IYER et al, 2005). For double redundancy, the computation must restart to recover from an error.

Fault-tolerance techniques implemented in software often determine relatively high performance overheads and high error-detection latency (IYER et al, 2005). Otherwise, techniques implemented in hardware result in lower latency. Furthermore, software-implemented techniques generally are not able to observe a large part of hardware-level errors. It occurs due to masking effects as detailed in section 2.1.2.3 or even because some specific microprocessor registers usually cannot be accessed by software applications.

Many fault-tolerance techniques are designed to protect the system against faults arisen in sequential elements, i.e., to mitigate direct SEUs. However, currently as a result of the scaling and technology issues, the techniques are also developed to protect the system against faults arisen in combinational circuits, i.e., to mitigate indirect SEUs. They are particularly based on hardware and time redundancy due to the nature of the target faults.

Especially, two fault-tolerance techniques are functionally detailed in the next sections due to their closed characteristics to the purposes of this work. They are applicable at the RT level and they do not use multiple clock networks. In addition, for any application, they preserve the total number of clock cycles, even so under a fault occurrence. The TMR scheme is able to mitigate only the direct SEUs and the TR+CWSP scheme is able to mitigate SET and therefore possible indirect SEUs.

3.1 Triple Modular Redundancy (TMR)

The TMR scheme is the most traditional fault-tolerance technique due to its good efficiency on error detection and its simple principle. Such scheme can be considered as a high-level technique because it can be implemented on high-level modules. However, it can be modeled at lower levels.

As shown in Figure 3.1, the TMR technique consists on the triplication of the target component to protect, in this case a 1-bit register. The three resulting outputs from triplication are connected to a voter block that compares the three received data and elects that of majority. If one of the three components fails or suffers a direct SEU, in the case of a register, the error will not be reflected in the voter output (HENTSCHKE et al, 2002).

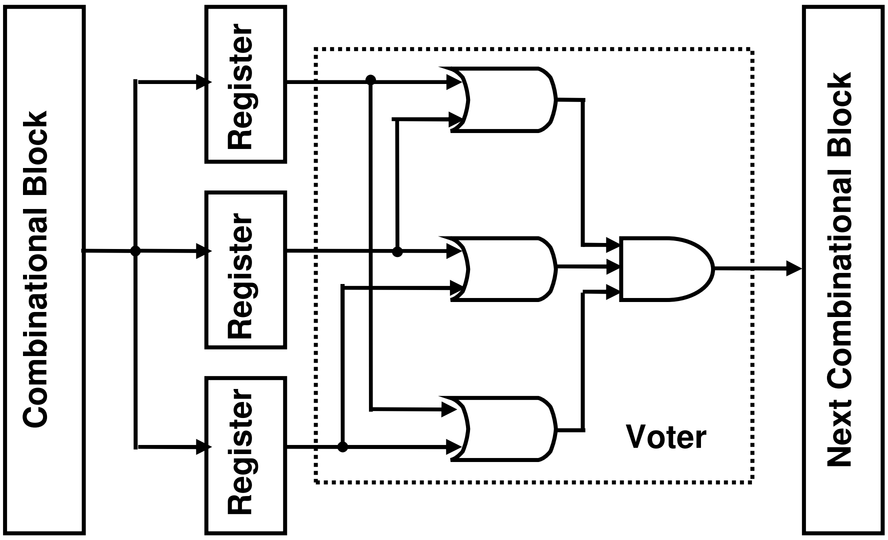


Figure 3.1: Block diagram of the TMR scheme for a 1-bit register

Observe that, in case of the register triplication, the voter block requires at least two registers without errors to elect a correct output. Therefore for the TMR mechanism working appropriately, direct SEUs, for example, cannot occur at the same time on two or three registers of the triplicated target register.

Another weakness of this technique is when a SET pulse occurs on the combinational block. The SET pulse might propagate itself up to the three registers of the TMR scheme and cause three indirect SEUs at the same time. Thus the voter block will not provide a correct output. By this reason, another technique which mitigates such faults is mandatory, as that presented in section 3.2.

3.1.1 Area and Performance Analysis

TMR technique implies in more than an increase of 200 % in area due to the component triplication. In case of the register triplication, the area related to registers is increased by 200 %. Furthermore, there is the voter that is implemented just with some OR and AND gates for each bit of the triplicated component.

In accord to Figure 3.1 and by considering only delays of the components (i.e., routing and parasitic issues are negligible), the **D**elay of the **C**ritical **P**ath of a **Non-Prot**ected **Circ**uit (**$D_{C\_P\_Non-Prot\_Circ}$**) is basically affected by the **D**elay of the **Voter** (**D_{Voter}**) when a TMR-based robustness is applied. It results in a **D**elay of the **C**ritical **P**ath of the **Rob**ust **Circ**uit (**$D_{C\_P\_Rob\_Circ}$**):

$$D_{C\_P\_Rob\_Circ} > D_{C\_P\_Non-Prot\_Circ} + D_{Voter} \quad (3.1)$$

3.2 Time Redundancy (TR) + Code Word State Preserving (CWSP)

Such approach was proposed by (NICOLAIDIS, 1999) and evaluated in (ANGHEL; ALEXANDRESCU; NICOLAIDIS, 2000-b; LAZZARI; ANGHEL; REIS, 2005). It exploits the pure TR principle, in which the output of the combinational circuit is duplicated at the time domain generally by using the delay of buffers or inverters. The two different instants of time are evaluated at two inputs of a peculiar element called Code Word State Preserving (CWSP). A block diagram of this scheme for a 1-bit register is shown in Figure 3.2.

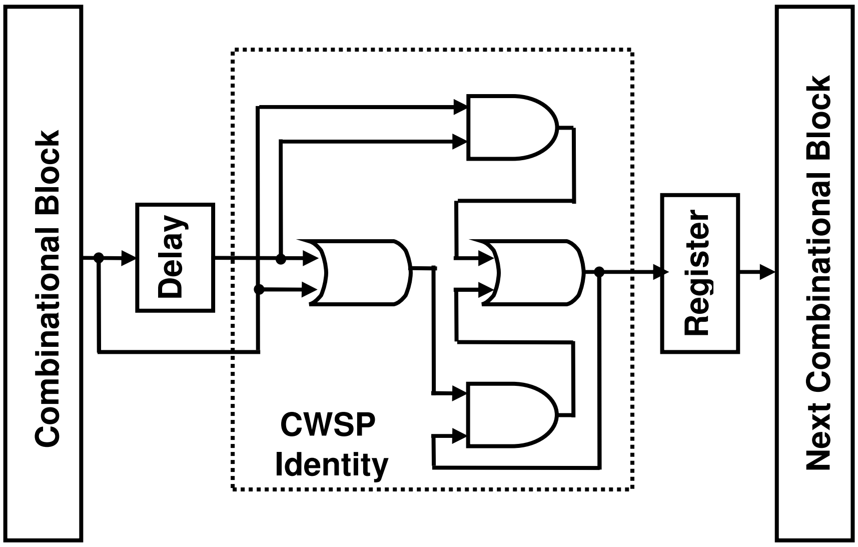


Figure 3.2: Block diagram of the TR+CWSP scheme for a 1-bit register

The CWSP element is an asynchronous sequential circuit able to mitigate SET pulse. It compares the values at its two inputs. When they are identical, the value at its output will be updated with the value of its inputs. On the other hand, when its inputs are not identical, the value at its output will be preserved.

Supposing a SET, which potentially causes an indirect SEU, occurs on a combinational block of a system. This SET arises at the output of the combinational block like that presented in Figure 3.4. The pulse shape meets the requirements of the set-up and hold times, i.e., the latching-window of the memory element as the vertical dotted lines in Figure 3.4. For a system without the TR+CWSP protection, such pulse gives rise to an indirect SEU at the output of the register.

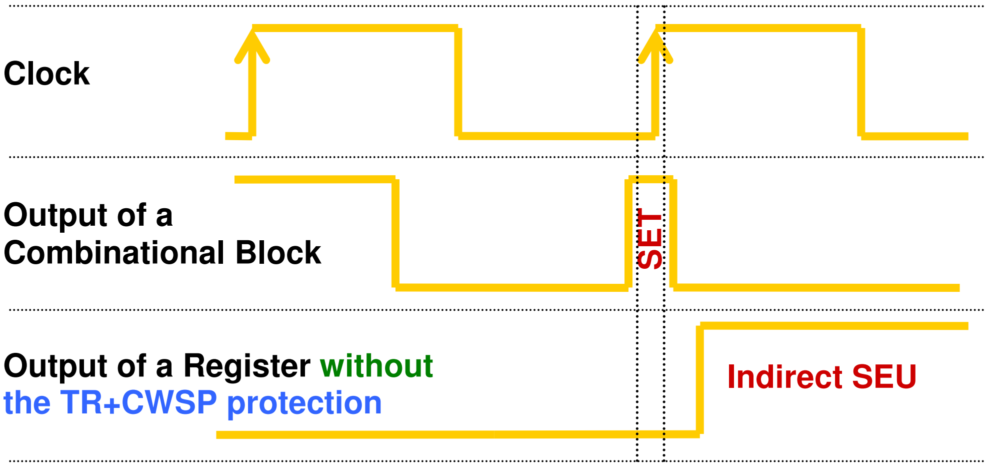


Figure 3.3: Timing behavior of a system without the TR+CWSP protection

In contrast, for a robust system with the TR+CWSP protection, this indirect SEU at the output of the register does not occur. As Figure 3.2 and Figure 3.4 illustrate, the CWSP element compares, by means of its two inputs, the output of the combinational block with the delayed output of the same block. The output of the CWSP element during the latching-window is preserved because the logic values at its two inputs are not equal. Thus the output of the register is not affected.

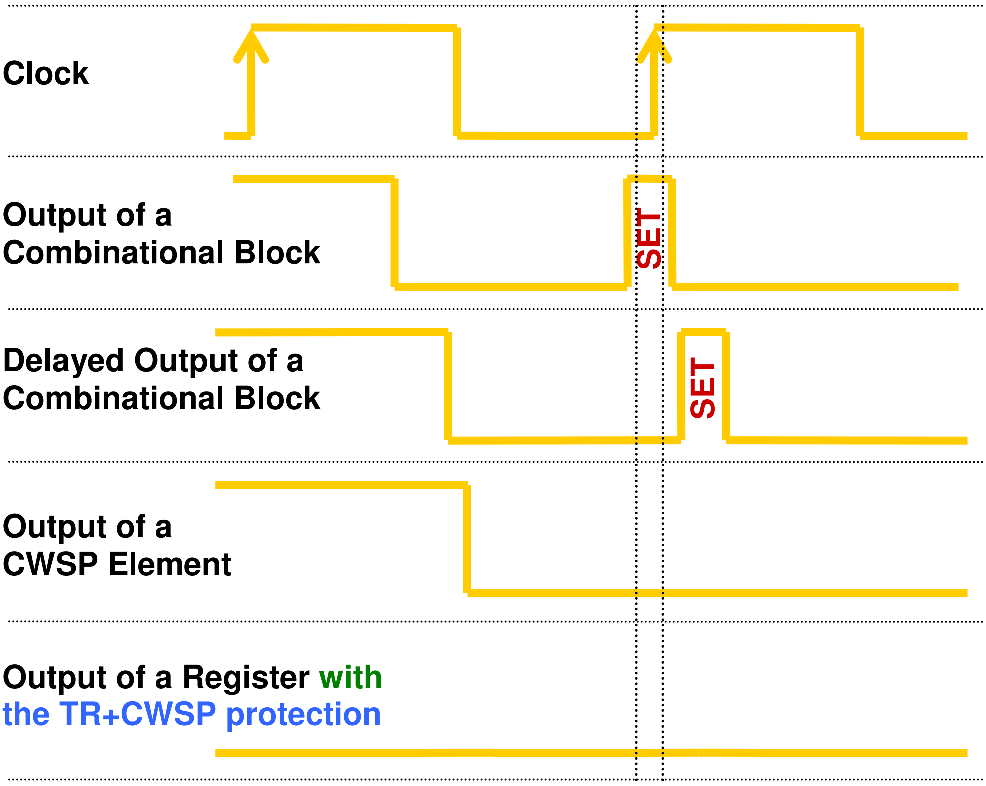

Clock
Output of a Combinational Block
SET
Delayed Output of a Combinational Block
SET
Output of a CWSP Element
Output of a Register with the TR+CWSP protection

Figure 3.4: Timing behavior of a system with the TR+CWSP protection

3.2.1 Area Analysis

Making robust a system by this TR+CWSP approach, the additional cost in area will be due to the buffers or inverters for implementing the delay blocks besides the CWSP elements.

The circuit of the CWSP element illustrated in Figure 3.2 uses standard combinational logic gates. This kind of CWSP element is an identity element. In other words, it does not make logic operation with its two data inputs such as NOT, NOR or NAND gates, but just transfers the data inputs to its output mitigating eventual SETs. In fact, the work proposed by (NICOLAIDIS, 1999) also suggests CWSP logic elements like NOT, NOR and NAND gates, as shown in Figure 3.5. The designs of these circuits are improved at the transistor level by reducing its overall number of transistors. Figure 3.6 illustrates these improved non-standard logic gates that keep on the same functionality of those from Figure 3.5.

Implementing a TR+CWSP scheme like that of Figure 3.7 characterizes this fault-tolerance mechanism as a low-level technique, since a peculiar non-standard gate is created at the transistor level to implement the CWSP element. Otherwise, a TR+CWSP scheme like that of Figure 3.2 can be considered as a high-level technique because only standard gates from any conventional library are used. In this case, the CWSP element is implemented at the gate level and it can be seen as a block at the RT level.

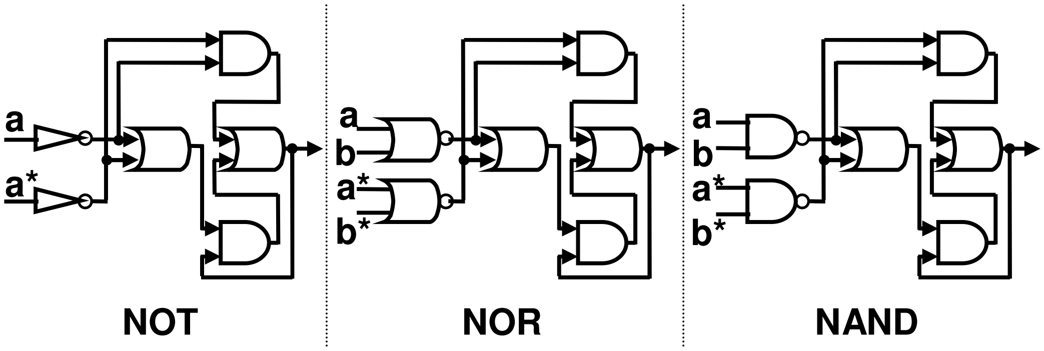


Figure 3.5: CWSP logic elements (NICOLAIDIS, 1999)

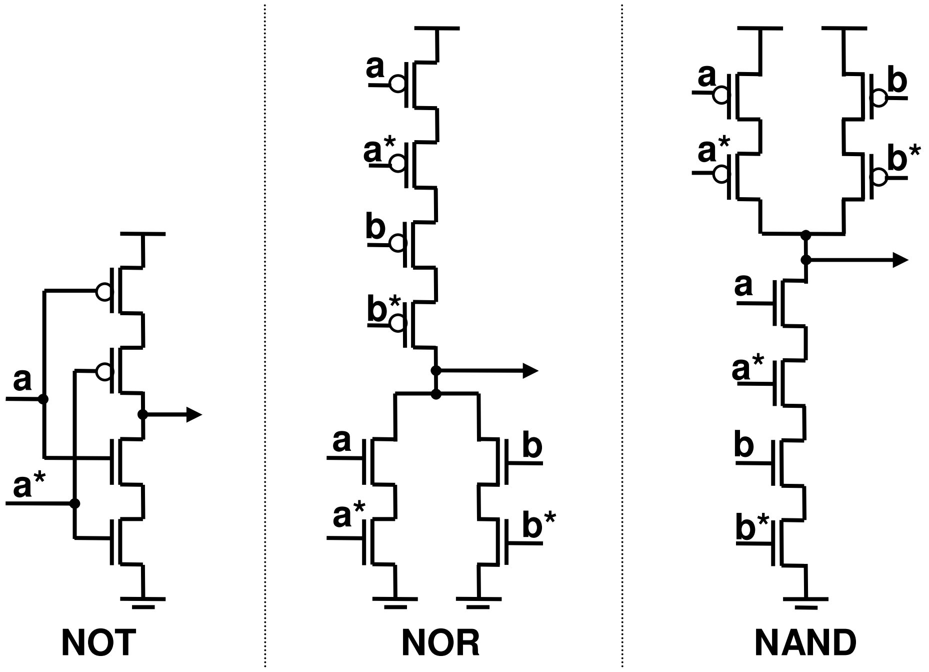


Figure 3.6: CWSP logic gates (NICOLAIDIS, 1999)

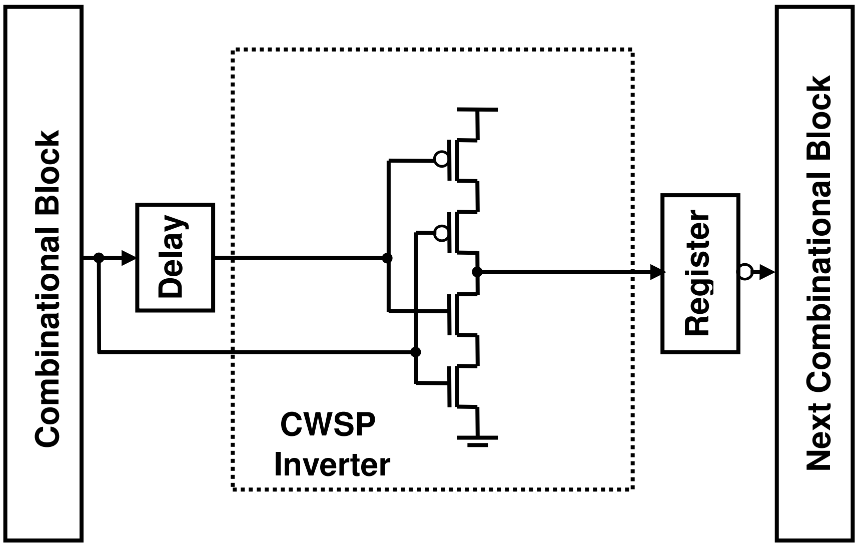


Figure 3.7: Block diagram of the TR+CWSP scheme using a non-standard gate

3.2.2 Performance Analysis

From the TR+CWSP scheme illustrated in Figure 3.2 or Figure 3.7, two characteristics can be observed:

- By reason of the TR principle, the delayed input of the CWSP block will be with the same value of its non-delayed input (i.e., the output of the Combinational Block) only after a given period $\mathbf{D}_{\mathbf{Delay\_Block}}$ (propagation time of the **Delay Block**).
- Based on the logic function of the CWSP block, explained in section 3.2, only if the values at its two inputs are identical, the value at its output will be updated with the value of its inputs. It would take a time interval $\mathbf{D}_{\mathbf{CWSP}}$ (propagation time of the **CWSP** block) to be completed. Otherwise, the value at its output will be preserved.

Therefore, as Figure 3.8 illustrates, the value at the output of the CWSP block (i.e., the register input) will only modify whether the value at its non-delayed input is equal to the value at its delayed input by at least a period D_{CWSP}. It usually occurs after the non-delayed input to reach its steady state within a clock cycle plus at least a time interval D_{Delay_Block}.

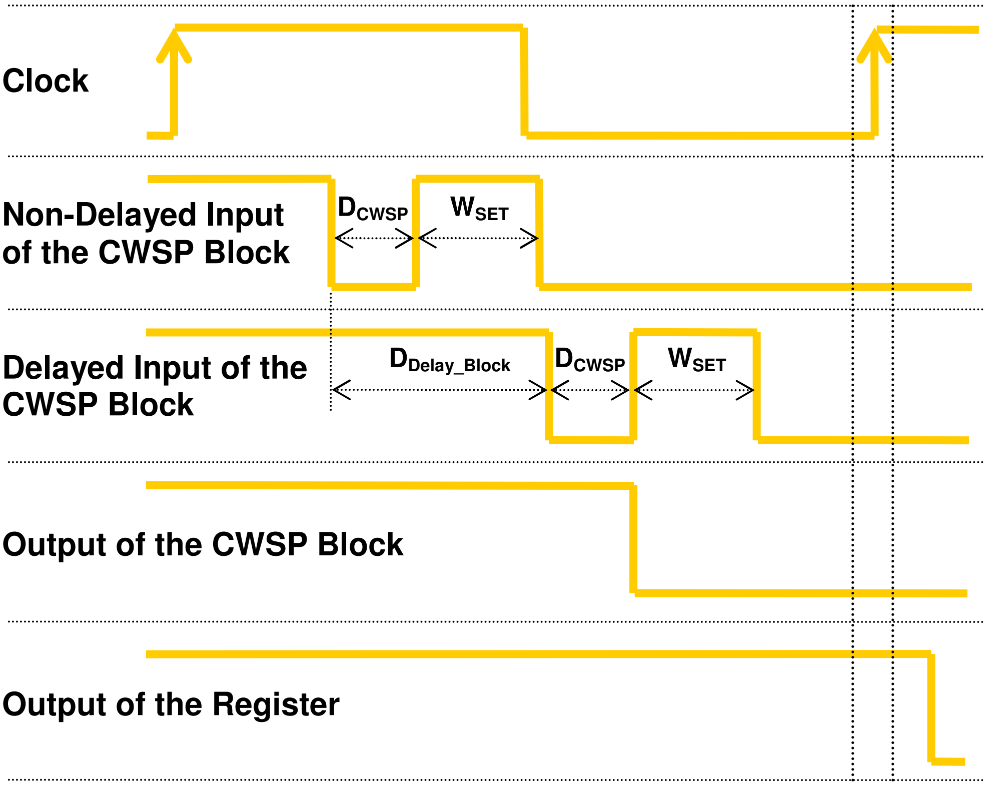


Figure 3.8: Functional characteristics of the TR+CWSP scheme

Such characteristics guarantee that any SET-pulse occurrence on the Combinational Block, in which the pulse reaches its output (i.e., the non-delayed input of the CWSP block) with a width lesser than $D_{Delay\_Block} - D_{CWSP}$, will be mitigated by the TR+CWSP scheme. Therefore, SET pulses will not arrive at the output of the CWSP block. Thus, the **W**idth of the **Max**imum **SET** pulse ($\mathbf{W}_{\mathbf{Max\_SET}}$) at the output of the Combinational Block that is able to be mitigated is defined by:

$$W_{Max\_SET} < D_{Delay\_Block} - D_{CWSP} \quad (3.2)$$

In accord to Figure 3.2 or Figure 3.7 and by not taking into account routing and parasitic issues, the **D**elay of the **C**ritical **P**ath of a **Non-Prot**ected **Circ**uit (**$D_{C\_P\_Non-Prot\_Circ}$**) is basically degraded by the delays of the components D_{Delay_Block}, D_{CWSP} and by an extra slack time W_{Max_SET} + D_{CWSP} required by the TR+CWSP scheme. It results in a **D**elay of the **C**ritical **P**ath of the **Rob**ust **Circ**uit (**$D_{C\_P\_Rob\_Circ}$**):

$$D_{C\_P\_Rob\_Circ} > D_{C\_P\_Non-Prot\_Circ} + D_{Delay\_Block} + D_{CWSP} + W_{Max\_SET} + D_{CWSP} \quad (3.3)$$

Reorganizing the $D_{C\_P\_Rob\_Circ}$ by using W_{Max_SET}:

$$D_{C\_P\_Rob\_Circ} > D_{C\_P\_Non-Prot\_Circ} + 2 \cdot D_{Delay\_Block} + D_{CWSP} \quad (3.4)$$

The extra slack D_{Delay_Block} required by such protection scheme can be explained by considering some issues. The non-delayed output of the Combinational Block reaches its steady state within a clock cycle after a stabilization period defined by $D_{C\_P\_Non-Prot\_Circ}$ – Tset-up, where **Tset-up** is the set-up time of the memory element or register. On the other hand, the delayed output of the Combinational Block reaches its steady state after $D_{C\_P\_Non-Prot\_Circ}$ – Tset-up + D_{Delay_Block}. In hypothesis of a SET arising at the non-delayed output after $D_{C\_P\_Non-Prot\_Circ}$ – Tset-up + D_{Delay_Block} + a time period slightly lesser than D_{CWSP}, in such way that both outputs of the Combinational Block do not achieve the same value by at least D_{CWSP}. Even so, there will be enough time, i.e., D_{Delay_Block} or around W_{Max_SET} + D_{CWSP}, to allow a correct updating of the CWSP output before the latching-window.

The TR+CWSP scheme allows also mitigating the called timing fault as a result of such extra slack established. Timing faults are due to fabrication process variations that can escape from the detection of production tests. They cause an enlargement of the delays of circuit paths (ANGHEL; NICOLAIDIS, 2000-a). By using the TR+CWSP scheme, timing faults will be mitigated whether they provoke enlargements of the delays of circuit paths at a maximum time variation up to D_{Delay_Block}. On the other hand, if there is a timing fault, the mechanisms to mitigate SETs may not work any more, since there will not be enough time slack to mitigate them.

4 DESIGN OF A ROBUST MICROPROCESSOR

As emphasized in chapter 3, commercial microprocessors typically use protection mechanisms such as parity and EDAC codes in order to mitigate SEs (IYER et al, 2005). These techniques are essentially focused on protecting memory arrays. As a result of this, some commercial systems can have their individual registers vulnerable to SEs. Former works (LIMA et al, 2000-a, 2000-b; COTA et al, 2001) dedicated special attention to implement an EDAC technique, the Hamming Code, on memory arrays and also on individual microprocessor registers of a microcontroller version from the Intel 8051 family. EDAC codes are relatively efficient for groups of memory elements or memory arrays like caches and perhaps register files. It is because the cost of the coding circuit can be amortized over the array. However, applying such codes to individual microprocessor registers could require a significant amount of overheads (HENTSCHKE et al, 2002; IYER et al, 2005) and thus other fault-tolerance mechanisms are necessary.

Another issue is that due to the current technology trends, protection mechanisms against indirect SEUs should be soon considered in IC designs (SHIVAKUMAR et al, 2002). The usual techniques like parity and EDAC codes are generally dedicated to mitigate direct SEUs, thus most commercial architectures result susceptible to indirect SEUs and other fault-tolerance techniques become mandatory.

Recently, two commercial microprocessors, MIPS and 8051, were protected in the work (LAZZARI; ANGHEL; REIS, 2005) with the aim of avoiding direct and indirect SEUs. The architectures were developed by using a commercial IC design flow through EDA tools. An extra special layout tool was used to implement non-standard gates similar to those CWSP gates from Figure 3.6 in section 3.2.1 for indirect SEU mitigation. In order to mitigate direct SEUs a TMR version that requires three clock signals was implemented. Such TMR version becomes the IC design flow more complex due to the extra networks of clock trees. The main goal of the work (LAZZARI; ANGHEL; REIS, 2005) is to evaluate the special tool of automatic layout generation. By this reason, few results in area and performance of the microprocessor implementations are presented. Furthermore, the fault-tolerant systems were not functionally verified.

The purpose of this present design is to follow activities and steps of an IC design that speed up the time-to-market and save development cost. In this way, some initial design constraints and final goals were established:

- Making robust to Soft SEEs or SEs, i.e., direct SEUs and also indirect SEUs, a commercial 8-bit microprocessor from the M68HC11 microcontroller family (FREESCALE, 2002);

- Using a IC design flow through conventional steps of commercial EDA tools;
- Starting from a high-level design language such as VHDL, creating functional blocks at the RT level, implementing through standard cells of any library and achieving a GDSII stream file for a future IC manufacture;
- Guaranteeing the functionality of such microprocessor by applying fault-tolerance techniques that ensure the reliability and reusability of their many system applications (hardware or software);
- The fault-tolerant mechanisms should be developed at high level as blocks at the RT level by using only standard gates, i.e., not adding in the design flow non-standard gates developed by full-custom layout tools;
- The implemented fault-tolerance techniques should require just one clock signal;
- For any application, the techniques should preserve the total number of clock cycles, even so under a fault occurrence;
- Previous cost evaluation of the robustness in the target microprocessor before the IC manufacture by means of estimated results in area, performance and also power.

An overview of the target microprocessor to be protected is shown in section 4.1. Design details of the developed fault-tolerant architecture are discussed in section 4.2. The design steps performed through a typical IC design flow are presented in section 4.3. The front-end design of microprocessor versions is detailed in section 4.4 and the back-end design is shown in 4.5. Finally, in section 4.6 some characteristics of the designed fault-tolerant architecture are emphasized.

4.1 The Target Microprocessor

Popular commercial microcontrollers are commonly mass-produced for electronic systems or embedded systems. Such systems have a wide range of applications in instrumentation, automation, control, telecommunication or even domestic appliances. Mass-produced ICs cost very little per unit due to the amortization of engineering costs over large number of volumes, high yields from many production runs and other economy-of-scale factors (VAHID; GORDON-ROSS, 2001). The most known mass-produced microcontrollers at the industry are from Freescale M68HC11, Intel 8051 and Microchip PIC families. These microcontrollers and their microprocessors are also largely used as cores or parts of SOCs.

In a microcontrolled system, as illustrated in Figure 4.1, there is a microprocessor or a Central Processor Unit (CPU), generally, an on-chip volatile memory just for data (stack, context or variables) and a non-volatile program memory accessed directly by the microprocessor, i.e., without another memory level. This kind of system typically has simple architectures and generally exclude features like multipliers, floating-point units, caches, deep pipelines and branch predictors (VAHID; GORDON-ROSS, 2001).

In the present work, the target is to make robust a Motorola or today Freescale M68HC11 microprocessor. It is a CISC architecture with 8 data bits and 16 address bits. All software instructions are executed in their programmed sequence, i.e., instructions are analyzed and data are processed in a sequential nature. The M68HC11 CPU can execute all M6800 and M6801 instructions (source and object-code compatible) and more than 90 new instruction opcodes. Since more than 256 instruction opcodes exist, a multiple-page opcode map is used in which some new instructions are specified by a page-select prebyte before the opcode byte (FREESCALE, 2002). Actually, this microprocessor can execute up to 308 different instructions.

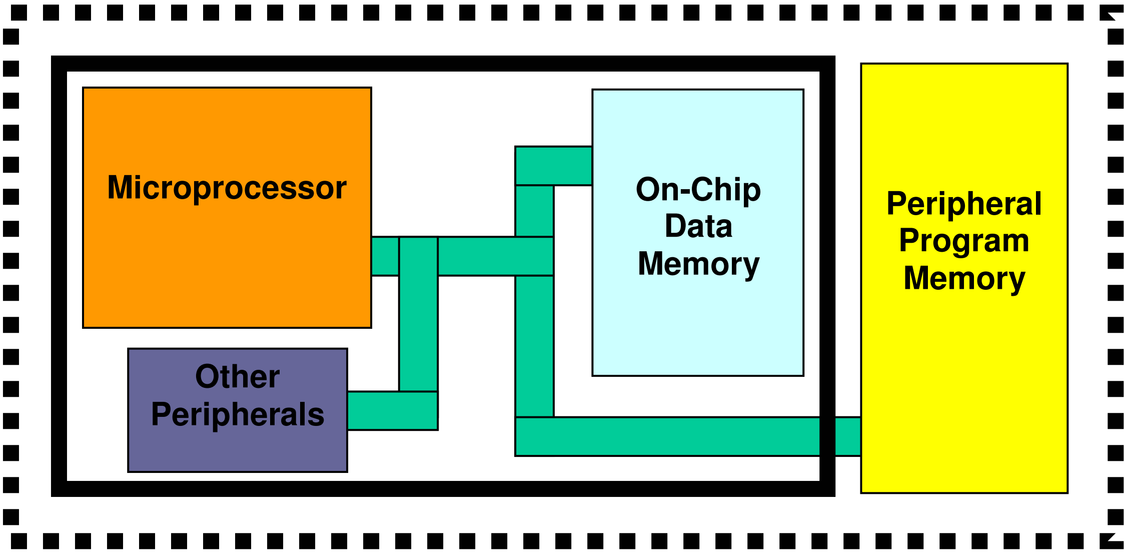


Figure 4.1: Illustration of a typical microcontrolled system

The architecture of the M68HC11 CPU considers all peripherals, on-chip devices, input/output (I/O) and memory locations to be treated identically as locations or addresses in the 64-Kbyte memory map (16-bit address bus). Thus, there are no special instructions for I/O that are separate from those used for memory. In addition, there is no execution-time penalty for accessing an operand from an external memory location comparing to a location within the microcontroller (FREESCALE, 2002). Such kind of CPU sometimes is called von Neumann architecture. The CPU can be either reading an instruction or reading/writing data from/to addresses of the memory map. Both operations cannot occur at the same time, since the instructions and data use the same signal pathways and memory map. It is different from Harvard architectures, in which the CPU can read both an instruction and data from the memory at the same time. In Harvard architectures, data and program memories are located separately by using different signal pathways and memory maps.

The main innovations of the M68HC11 CPU compared to the earlier M6801 and M6800 CPUs (FREESCALE, 2002) are:

- The inclusion of a second index register (Y);
- New instructions of bit manipulations that allow accessing bits in some memory localizations in the 64-Kbytes address space;
- Two new instructions that do a division 16 by 16 bits;
- Transfer instructions from the indexation register to the 16 bit double accumulator;
- Updated instructions for easier complete arithmetic operations.

The M68HC11 CPU support four data types: bit data; 8 bits and 16 bits signed and unsigned integers; 16 bits unsigned fractions and 16 bits addresses (FREESCALE, 2003). Six addressing modes can be used to access the memory: immediate, direct, extended, indexed, inherent and relative.

Seven CPU registers are visible for the programmer or software designer (FREESCALE, 2003). Figure 4.2 shows such registers and in the following paragraphs they are briefly described:

- **Accumulators A and B**: are general-purpose 8-bit accumulators used to hold operands and results of arithmetic calculations or data manipulations. Some instructions treat the combination of these two 8-bit accumulators as a 16-bit double accumulator (**accumulator D**);
- **Index Registers X and Y**: are 16-bit index registers used for indexed addressing mode. In the indexed addressing mode, the contents of a 16-bit index register are added to an 8-bit offset, which is included as part of the instruction, to form the effective address of the operand to be used in the instruction;
- **Stack Pointer SP**: is the pointer of a program stack supported automatically by the CPU. This stack may be located anywhere in the 64-Kbyte address space and may have any size up to the amount of data memory available in the system. At any given time, the stack pointer register holds the 16-bit address of the next free location on the stack;
- **Program Counter PC**: is a 16-bit register that holds the address of the next instruction to be executed;
- **Condition Code Register CCR**: contains five status indicators, two interrupt masking bits and a STOP disable bit. The five status flags reflect the results of arithmetic and other operations of the CPU as it performs instructions. The five flags are half carry (H), negative (N), zero (Z), overflow (V) and carry/borrow (C). The interrupt request (IRQ) mask (I bit) is a global mask that disables all maskable interrupt sources. The XIRQ mask (X bit) is used to disable interrupts from a certain pin. The STOP disable (S) bit is used to allow or disallow the STOP instruction.

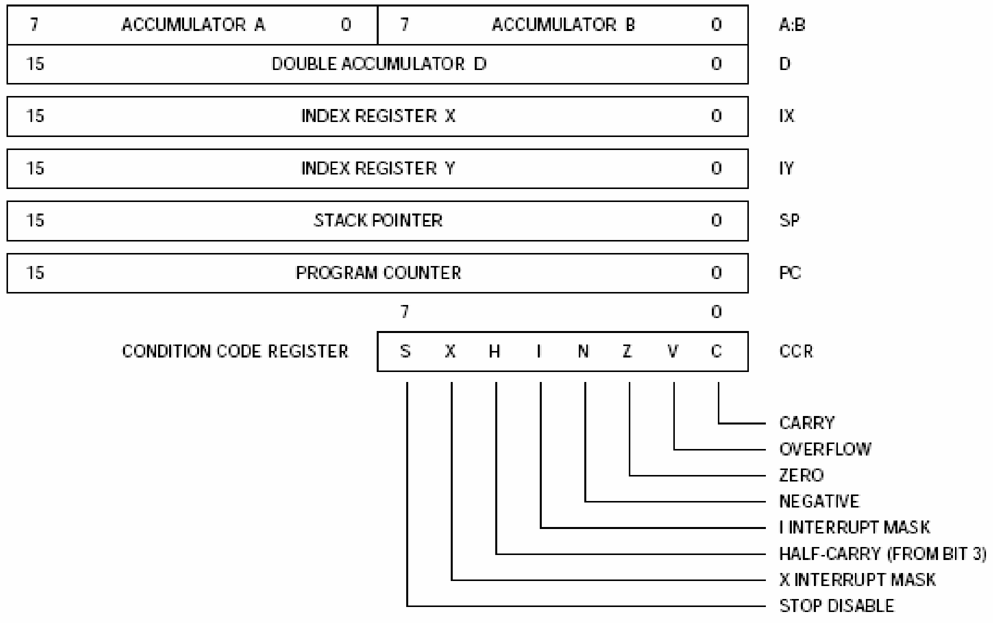


Figure 4.2: CPU registers visible to the programmer (FREESCALE, 2003)

Figure 4.3 shows a simplified diagram with the main functional blocks in a version of the M68HC11 architecture. Such CPU is basically organized into 6 blocks:

- **Branch Coder**: codes the next address for branch;
- **Control Unit**: generates the next values to State, Address and PC registers based on the current instruction code and state;
- **Interruption Decoder**: decodes an interruption;
- **Operation Coder**: codes an operation to Registers and ALU based on the current instruction code and state;
- **ALU** (the Arithmetic Logic Unit): executes the arithmetic and logic operations;
- **Registers**: include all 18 internal registers or sets of flip-flops dispersed on the CPU area. They totalize 187 1-bit flip-flops. The registers are: 8-bit A, 16-bit Address, 16-bit ALU, 8-bit B, 8-bit CCR, 4-bit Counter4, 1-bit D_Prefix, 8-bit Datain, 16-bit Load_Addr, 8-bit Opcode, 16-bit PC, 8-bit Prev_Data, 16-bit Reg_Addr, 16-bit SP, 5-bit State, 16-bit X, 1-bit Y_Prefix, 16-bit Y.

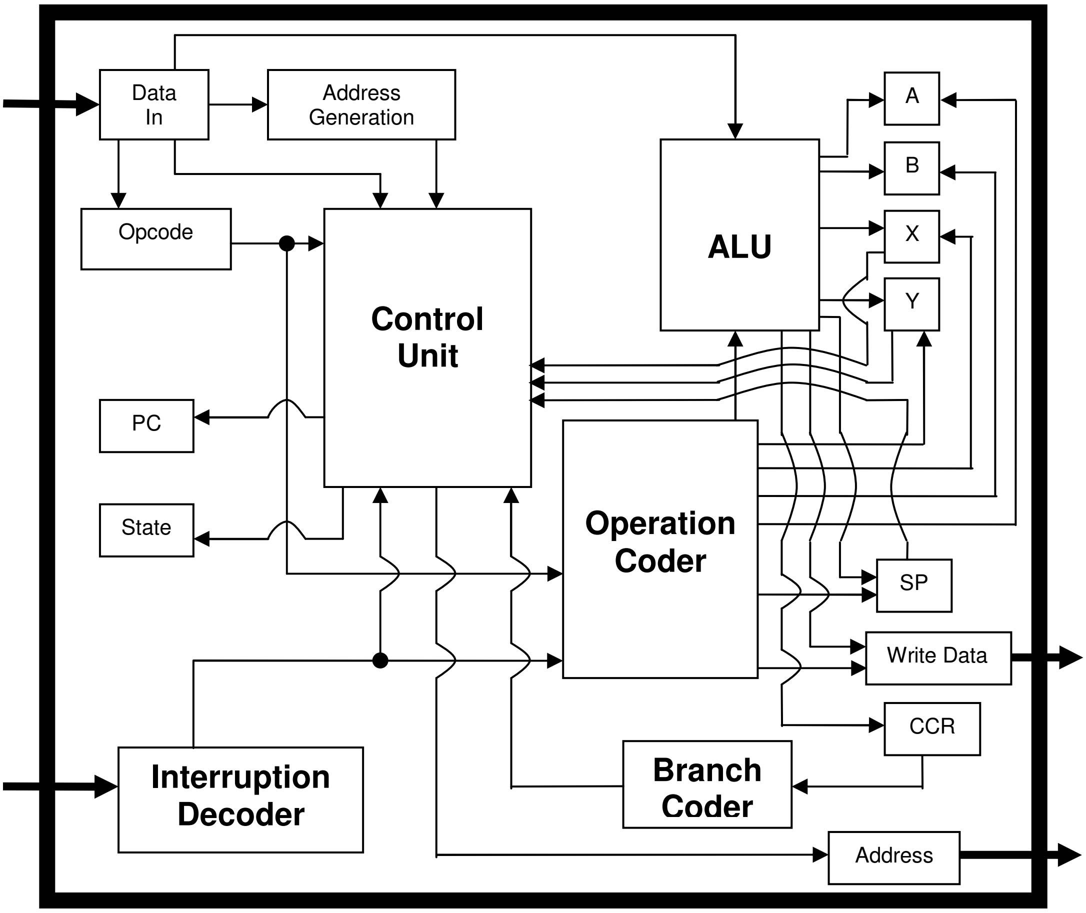


Figure 4.3: Main functional blocks of the M68HC11 architecture

4.2 Fault-Tolerant Circuit Design

In the architecture of a M68HC11 microprocessor core, the unique existing sequential or memory elements, which can potentially store wrong values due to a fault event, are the 18 individual registers dispersed on the core area. The remaining area corresponds to the combinational blocks.

The fault-tolerant version design of the M68HC11 microprocessor is based on applying fault-tolerance techniques on its vulnerable elements to Soft SEEs. Since the objective is the IC manufacture not a FPGA implementation, the overall microprocessor circuit would be potentially susceptible to direct SEUs on its sequential elements and to indirect SEUs through SETs on its combinational blocks, as discussed in 2.1.2. The implemented fault-tolerance techniques must be able to mitigate such faults.

There are fault-tolerance techniques based on detection circuits like those presented in (ANGHEL; NICOLAIDIS, 2000-a) that require retry procedures for correction. Applying on susceptible IC elements such techniques can be disadvantageous at the cycle-timing aspect of microprocessor software applications. Typically, in a clock cycle, a functional operation of the circuit is performed. The same clock cycle is generally also used to monitor an eventual fault and to process the detection hardware task. In the next clock cycle, based on the result of the detection, it is performed a decision hardware task. If a fault is detected, extra clock cycles are generated in order to execute the error correction task, or in other words to retry the functional procedure of the previous clock cycle. A fault-tolerance technique that maintains, even under an eventual fault, the number of cycles of a software execution is quite desirable. It is to avoid unexpected overheads in performance and to guarantee the reusability of the system design. Furthermore, the fault-tolerance technique implementation in the circuit would be simplified and less arduous due to the absence of retry procedures and unexpected extra cycles that need not be predicted.

Another issue is that some fault-tolerance techniques like those proposed in (NICOLAIDIS, 1999; ANGHEL; NICOLAIDIS, 2000-a; KRISHNAMOHAN; MAHAPATRA, 2004) require more than one clock to evaluate the data signal at different time instants. In this way, these TR versions also require building trees for multiple clocks in order to avoid eventual clock skew. Thus, an extra cost in area and power to allocate the additional clock networks is inherent. Moreover, the design complexity increases and the compatibility with standard systems can be affected as a result of the exigency of support and supply circuits for multiple clocks.

In order to avoid direct SEU, a traditional fault-tolerance technique due to its efficient error detection is the TMR approach presented in section 3.1. It might be costly in area and power. However, it can sometimes provide better results in area than EDAC techniques like Hamming Code. In the work (HENTSCHKE et al, 2002) both techniques were compared by using arithmetic circuits with pipeline and register files. Results indicate that TMR is more appropriated to protect single registers like those in pipelines, control and data-path circuits. On the other hand, as already emphasized at the beginning of this chapter, Hamming Code is more suitable to protect groups of storage cells like RAMs. Another valorous detail of the TMR technique is that it protects against errors on all bits of a register. Furthermore, the circuit critical path is only affected by the delay of the voter, no unexpected extra clock cycles can occur and just one clock is necessary. The TMR implementation at the RT level makes simple, since descriptions of systems at this level usually have their sequential elements in modular components.

Applying TMR is enough just to mitigate direct SEUs on sequential elements, nevertheless covering indirect SEUs due to SETs on combinational blocks is necessary another fault-tolerance technique. A TR version presented in section 3.2 can be an adequate alternative to mitigate indirect SEUs. It uses the special CWSP element proposed by (NICOLAIDIS, 1999) to tolerant SET pulses. Such approach has as main virtue to work with only one clock. Additionally, it does not require retry procedures that could result in unexpected extra clock cycles. This technique can also be developed at the RT level by using standard combinational logic gates to build the CWSP element as a component. Figure 3.2 illustrates this approach.

4.3 Integrated Circuit Design Flow

Typical integrated circuit design flows developed at semiconductor companies are discussed in (SMITH, 1997; DAVIS et al, 2000; BRÜNING, 2006). Such design flows show the sequence of usual steps to design a complex IC such as a microprocessor. They are based on circuit models at high-level of abstraction and standard cells of a technology to cope with the current high complexity of the circuits. Otherwise, low-level models and full-custom designs are more accurate but they are more onerous and a large number of engineers or a lot of time would be required to design an IC.

Two major design parts can be considered in the design flows presented in (SMITH, 1997; DAVIS et al, 2000; BRÜNING, 2006): front-end design which is the logical design; and back-end design which is the physical design. Normally, different engineers handle the front-end and back-end design, even so there is some overlap between these two design parts. The front-end and back-end parts can be well defined by Figure 4.4 published in (SMITH, 1997). The steps of this typical design flow are briefly presented below:

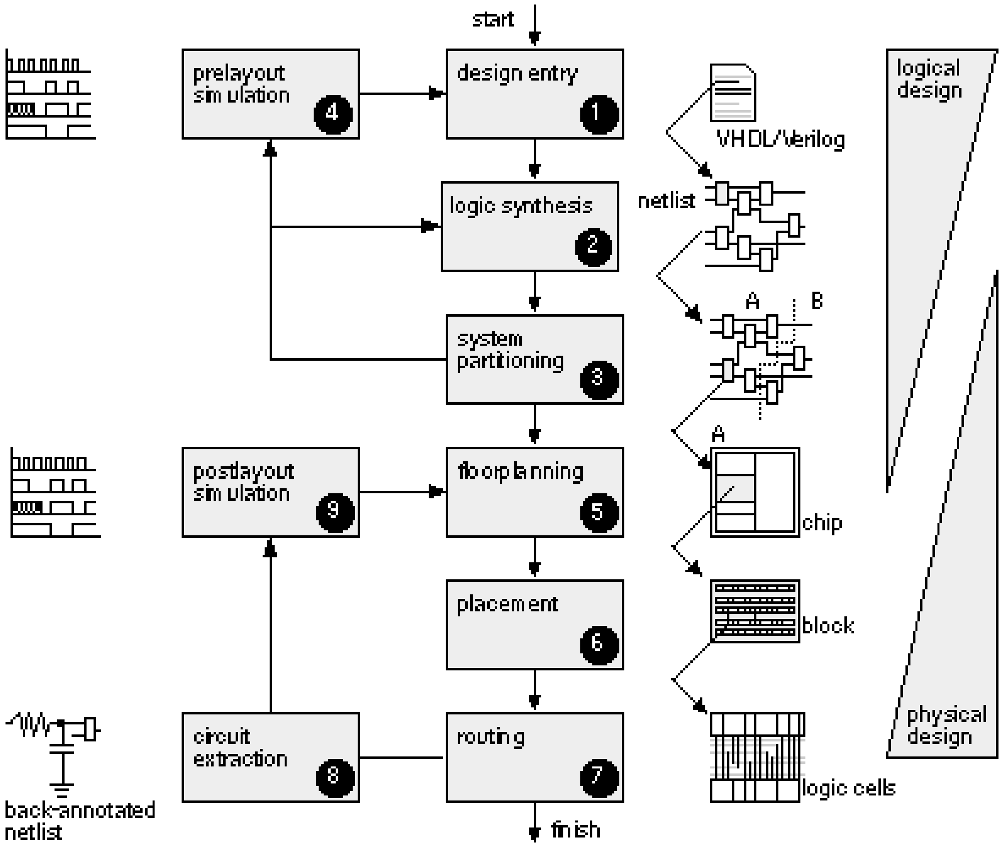


Figure 4.4: A typical IC design flow (SMITH, 1997)

- **Design entry**: the initial entry into the design flow, either using a hardware description language (HDL) or schematic entry;
- **Logic synthesis**: by using an HDL (**VHDL** or **Verilog**) and a logic synthesis tool to produce a description of the logic cells and their connections known as **netlist**;
- **System partitioning**: divide a large system into IC-sized pieces. This step is especially important for even more complex systems composed of many functional units or blocks. If a functional block is too large to fit in one piece, a partition of the function into pieces may have to be done. Common or standard parts are allocated into different IC-sized pieces;
- **Pre-layout simulation**: check to see if the design functions correctly;
- **Floorplannig**: arrange the blocks of the netlist on the chip;
- **Placement**: decide the locations of cells in a block or unit of the IC;
- **Routing**: make the connections among cells, blocks or units;
- **Extraction**: determine the resistance and capacitance of the interconnect;
- **Post-layout simulation**: check to see if the design still works with the added loads of interconnects.

Typically, the steps 1 to 4 in Figure 4.4 are tasks of the front-end design and steps 5 to 9 of the back-end design. However, there might be some overlaps. The system partitioning, for instance, is usually performed by considering both logical and physical factors.

4.3.1 The Developed Design Flow

Based on the design flows (SMITH, 1997; DAVIS et al, 2000; BRÜNING, 2006) and fundamentally on that suggested by the EDA tools (CADENCE, 2002), in the present work, a design flow illustrated briefly in Figure 4.5 was developed. Nowadays, these commercial EDA tools for simulation, synthesis, partitioning, floorplanning, placement, routing, extraction, verification and analysis are amply used by semiconductor industries essentially because they support engineers to faster design even more complex systems. In addition to EDA tools from (CADENCE, 2002), a logic simulator from (MENTOR, 2004) was also used in this work.

By means of several different steps, this IC design flow starts from a VHDL description of the target circuit at RT level and achieves an equivalent representation at the physical level. This physical representation used by foundries in an IC manufacture process is a stream file known as Graphical Design System II (GDSII). It contains the geometry information of the IC physical design. An equivalent file format known as the Caltech Intermediate Form (CIF) is also usual.

In the present work, the design flow in Figure 4.5 starts from a VHDL description of the target architecture (**step 1**) presented in section 4.1. As the front-end logical design discussed in section 4.4, this description was worked at the RT level in order to make robust the architecture.

At **step 2** of this design flow, an initial verification by a behavioral simulation, detailed in chapter 5, was performed based only on the VHDL code worked at the RT level without a logic synthesis. Thus, this code characterizes the behavior of a system with circuits logically non-simplified and with no physical information. It requires a single logic simulator tool able to check VHDL code syntax, compile VHDL code and simulate data streams on the system. As the circuit information evaluated by this tool is simple, it requires very little execution time. Furthermore, this step 2 occurs ahead of running EDA tools that demand a larger processing and design time. It allows speeding up the correction of eventual design errors detectable at the RT level.

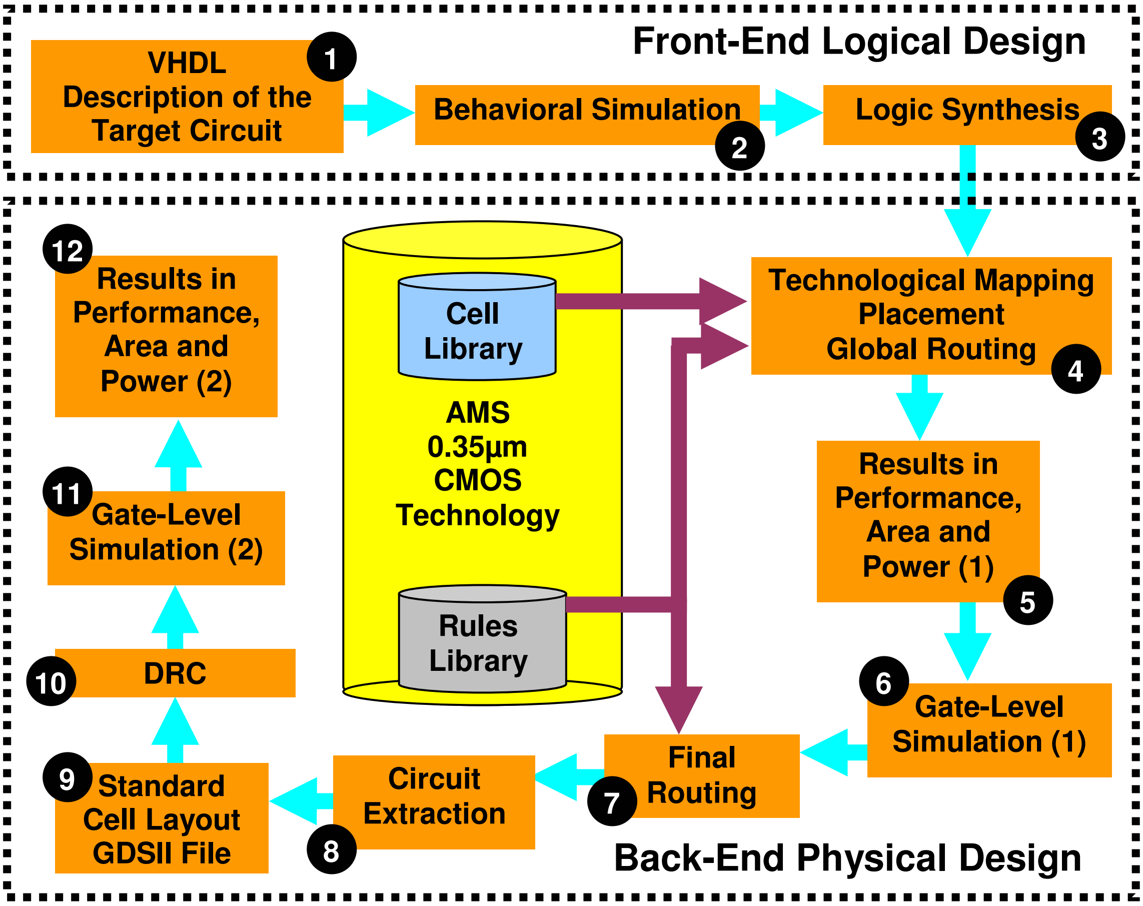

Front-End Logical Design
1
VHDL Description of the Target Circuit
Behavioral Simulation
2
Logic Synthesis
3
12
Results in Performance, Area and Power (2)
Cell Library
AMS 0.35µm CMOS Technology
Rules Library
Technological Mapping Placement Global Routing
4
Results in Performance, Area and Power (1)
5
11
Gate-Level Simulation (2)
10
DRC
6
Gate-Level Simulation (1)
9
Standard Cell Layout GDSII File
Circuit Extraction
8
Final Routing
7
Back-End Physical Design

Figure 4.5: The developed design flow

At **step 3**, it was done a logic synthesis of the circuits described in the worked VHDL code. Depending on the complexity of the target circuit, some design steps can be simplified. The step relating to the system partitioning from Figure 4.4 was not worth performing because the target system of this work is composed of few functional blocks that are not too large. By this reason and due to step 2 from Figure 4.5, unlike the design flow from Figure 4.4, another verification simulation was not performed thereupon the logic synthesis. On the other hand, a new verification simulation was worth performing after step 4 from Figure 4.5, since the available EDA tool platform allows easily arranging the logic synthesizer tool together with this step 4.

Then, at **step 4** from Figure 4.5, the back-end physical design discussed in section 4.5 starts by using standard cells from the AMS 0.35 µm CMOS technology (AUSTRIAMICROSYSTEM, 2003). This target technology uses 4 layers of metal and allows a power supply (vdd) of 3.3 V. At this step 4, various design steps could be assigned due to the practical software resources provided by the EDA tools. The step 4 arranges the technological mapping, floorplanning, timing analysis, placement, clock-tree generation and global routing. At **step 5**, a first set of preliminary estimated results in area, performance and power could be analyzed before the final routing. It can be useful to evaluate quickly but superficially the IC design viability for a manufacture.

The arrangement done at step 4 makes easy to run another verification experiment (**step 6**) as a pre-layout simulation, since it is before the detailed or final routing step (**step 7**). This verification allows the correction of eventual design errors occurred up to this step. Details about step 6 are presented in chapter 5.

Afterwards, at **step 8**, a circuit extraction by taking parasitic elements was performed. At **step 9**, the GDSII file was created from the generated final standard cell layout. At **step 10**, a Design-Rule Check (DRC) was performed. It is the major check that is typically used before a fabrication. Finally, at **steps 11 and 12**, a more accurate verification and analysis of the post-layout design could be done due to the extracted parasitic information.

During the development of an IC design, it is quite common the front-end and back-end designers do not succeed at the phases of verifications or checks. It usually occurs due to design errors such as logical errors, timing or geometry violations. By this reason, every design step can loop to every other step in order to fix design details or to adjust new sets of constraints.

4.4 Front-End Logical Design

In the IC design of this work, fault-tolerance mechanisms were implemented in the target microprocessor at the RT level. In accord to section 4.2, the TR+CWSP technique was used to protect the combinational blocks of the microprocessor. The TMR makes robust the microprocessor registers and elements of the TR+CWSP scheme.

In order to obtain the costs of the robustness in the target architecture, three microprocessor versions were developed:

- The **Non-Protected** version which is the reorganized architecture of the CPU without any fault-tolerance mechanism;
- The **TMR** version that is just protected by applying TMR on the registers and thus it mitigates only direct SEUs;
- The **TMR+TR+CWSP** version that is the robust version to direct and indirect SEUs by using TMR and the TR+CWSP scheme.

4.4.1 Non-Protected Version or Susceptible to Direct and Indirect SEUs

In this design, a VHDL description (THIBAULT, 2000) of the M68HC11 CPU was initially used. It differs from the standard CPU (FREESCALE, 2002) only by not implementing two instructions of division 16 by 16 bits (fractional and integer).

Such M68HC11 VHDL description (THIBAULT, 2000) was developed by using high-level resources of the VHDL. It presents a quite behavioral characteristic, i.e., there is a unique VHDL architecture with many concurrent processes. As Figure 4.6 illustrates, in which E_i is the clock signal of the system, combinational and sequential logics are not described in individual processes. Furthermore, the typical separation of a digital system between operative and control parts is not clearly organized.

```
process (E_i, tsc_i, state, address_i)
begin
   if (E_i'event and E_i = '1') then
      if (tsc_i = '0') then
         if (state = LOAD1) then
            load_addr <= address_i;
         end if;
      end if;
   end if;
end process;
```

Figure 4.6: A process from the original VHDL description

Before starting the implementation of the fault-tolerance techniques, the target system description need be adapted in a practical way. It must allow that designers know identifying combinational and sequential elements easier and thus applying uniformly the fault-tolerance techniques. Such practice in principle should be a usual rule in any description of digital system. However, digital systems can be developed by designers with different backgrounds. Occasionally, hardware descriptions are worked by using excessive high-level resources of the languages in which synthesis tools do not succeed in implementing. Furthermore, basic rules of structure and indentation of the description frequently are not considered.

In the present design, the microprocessor description (THIBAULT, 2000) was adequately reorganized in such way that each one of its sequential elements (18 registers) was individually separated from its combinational blocks in accord to Figure 4.7. The description was also structured by transforming the main functional blocks such as those from Figure 4.3 into VHDL components.

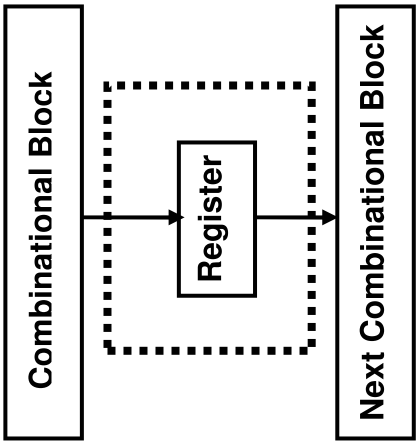


Figure 4.7: Combinational and Register blocks in the Non-Protected version

Some tips, which were developed in this present design with the intention of making easy the fault-tolerance implementation in the CPU, are presented below:

4.4.1.1 Analysis of Sequential Logics and Attached Combinational Logics

The identification of sequential logics and the separation of eventual attached combinational logics in VHDL processes as that from Figure 4.6 can be summarized into the following steps:

- Identifying all probable sequential logics in the original VHDL description by searching for sensitive processes to the clock signal(s) of the system. An example is shown in Figure 4.6, in which E_i is the clock signal of the system;
- Removing all combinational logic from identified original processes. It results for each one of the processes a VHDL process purely sequential (Figure 4.8);

```
process (E_i, address_i)
begin
   if (E_i'event and E_i = '1') then
      load_addr <= address_i;
   end if;
end process;
```

Figure 4.8: A VHDL process purely sequential

- Creating a VHDL signal for interconnection between the combinational logic and sequential process. It should be done in each one of those identified processes. In Figure 4.9, the new signal called “address_i_signal” replaces the former “address_i”;

```
process (E_i, address_i_signal)
begin
   if (E_i'event and E_i = '1') then
      load_addr <= address_i_signal;
   end if;
end process;
```

Figure 4.9: VHDL process purely sequential with the new interconnection signal

- Generating another VHDL process as illustrated in Figure 4.10 that characterizes only the combinational logic from the original process (Figure 4.6). Notice that as seen in Figure 4.10, the “else” construction is included in all “if” structures. It is because by means of the VHDL synthesis tools, any construction such as “if” or “case” can also implement sequential logics such as latches. Whenever not all options of the tested signals are evaluated, latches will be created. Since the intention is originally creating a pure combinational logic like a multiplexer, the “else” construction is included in order to avoid latches;

```
process (state, address_i, tsc_i, load_addr)
begin
   if (tsc_i = '0') then
      if (state = LOAD1) then
         load_addr_signal <= address_i;
      else
         load_addr_signal <= load_addr;
      end if;
   else
      load_addr_signal <= load_addr;
   end if;
end process;
```

Figure 4.10: A VHDL process purely combinational

- Analyzing reports of results provided by the synthesis tool with the purpose of verifying if all sequential logics were actually identified and if none extra latch was implemented due to the creation of the new VHDL processes for the combinational logics.

4.4.1.2 Modeling All Registers in a Unique Reusable Component

After the design steps from section 4.4.1.1, all sequential elements of the target microprocessor, which are 18 registers, can be arranged in a practical way. All 18 registers can be modeled by a unique reusable parameterized VHDL component (Figure 4.11), in which the unique required parameter is a VHDL generic that assign the number of bits of the target register to be instantiated. Reset and enable ports could be also implemented in this VHDL component, it was not included in the next figures just to simplify the illustrations.

Note that this approach requires a unique VHDL architecture and therefore a unique VHDL file to describe all 18 registers of the system. Such unique architecture was instantiated in 18 different points of the VHDL description (in the top VHDL architecture or inside functional blocks) where were described the 18 registers. It makes easy the implementation of the fault-tolerance mechanisms. For all registers, it is required to instantiate just once the VHDL components that model the fault-tolerance techniques (the TMR and TR+CWSP techniques, in this present design).

Furthermore, this practice of structuring the VHDL description also allows improving the visibility, manipulability and reusability of the VHDL code. All this reduces the susceptibility to designer's errors, since the individual protection of the registers would involve the creation of many VHDL processes in different files.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity RegisterComponent is
   generic
   (
      number_of_bits          : integer
   );
   port
   (
      clock                   : in std_logic;
      data_in                 : in std_logic_vector((number_of_bits - 1) downto 0);
      data_out                : out std_logic_vector((number_of_bits - 1) downto 0)
   );
end RegisterComponent;
architecture RTL of RegisterComponent is
begin
Register:
   process (clock, data_in)
   begin
      if (clock'event and clock = '0') then
         data_out <= data_in;
      end if;
   end process;
end RTL;
```

Figure 4.11: The unique reusable parameterized VHDL component for all registers

4.4.2 TMR Version or Robust to Direct SEUs

Based on reorganized Non-Protected version of the microprocessor previously presented in 4.4.1, the TMR version was structured as Figure 4.12. A VHDL component that characterizes the voter circuit illustrated in Figure 3.1 was created as Figure 4.13 and Figure 4.14.

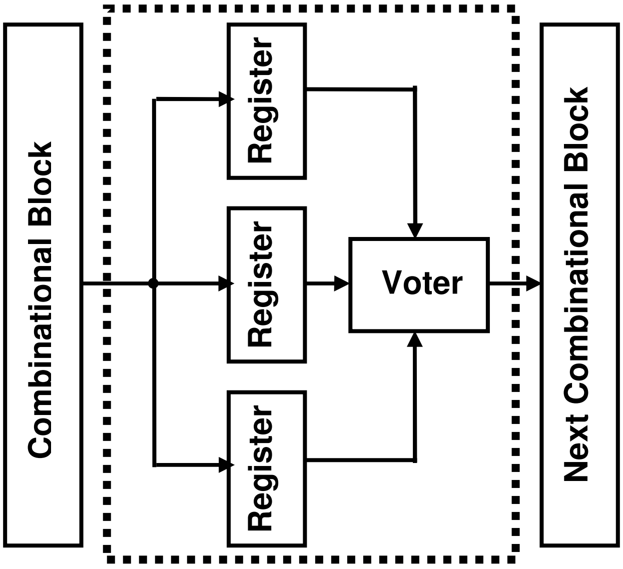


Figure 4.12: Combinational and Register blocks in the TMR version

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity VoterComponent is
    generic
    (
        number_of_bits          : integer
    );
    port
    (
        data_in_0               : in std_logic_vector((number_of_bits - 1) downto 0);
        data_in_1               : in std_logic_vector((number_of_bits - 1) downto 0);
        data_in_2               : in std_logic_vector((number_of_bits - 1) downto 0);
        data_out                : out std_logic_vector((number_of_bits - 1) downto 0)
    );
end VoterComponent;
architecture RTL of VoterComponent is
begin
Voter:
    process (data_in_0, data_in_1, data_in_2)
    begin
        data_out <= ((data_in_0 or data_in_1) and (data_in_1 or data_in_2) and
(data_in_0 or data_in_2));
    end process;
end RTL;
```

Figure 4.13: Voter component

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

package fault_tolerance_mechanisms is

   component VoterComponet
      generic
      (
         number_of_bits   : integer
      );
      port
      (
         data_in_0        : in std_logic_vector ((number_of_bits - 1) downto 0);
         data_in_1        : in std_logic_vector ((number_of_bits - 1) downto 0);
         data_in_2        : in std_logic_vector ((number_of_bits - 1) downto 0);
         data_out         : out std_logic_vector ((number_of_bits - 1) downto 0)
      );
   end component;

end package fault_tolerance_mechanisms;
```

Figure 4.14: Package of the fault-tolerance mechanisms (1)

The TMR version of the unique reusable component detailed in Figure 4.11 is presented in Figure 4.15 and Figure 4.16. The registers of the microprocessor were triplicated by including the “generate” construction of the VHDL in the unique reusable register component. The voter component was instantiated in the register component by connecting its three inputs to the three outputs of the triplicated register.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
library work;
use work. fault_tolerance_mechanisms.all;

entity RegisterComponent is
   generic
   (
      number_of_bits          : integer
   );
   port
   (
      clock                   : in std_logic;
      data_in                 : in std_logic_vector((number_of_bits - 1) downto 0);
      data_out                : out std_logic_vector((number_of_bits - 1) downto 0)
   );
end RegisterComponent;
```

Figure 4.15: TMR parameterized component for each one of the registers (part 1)

```
architecture RTL of RegisterComponent is
   type type_data_in is array (0 to 2) of std_logic_vector((number_of_bits - 1)
downto 0);
   signal signal_data_in       : type_data_in:=(others => (others=>'0'));
begin
Redundant_Registers:
   for i in 0 to 2 generate
   Register:
      process (clock, data_in)
      begin
         if (clock'event and clock = '0') then
            signal_data_in(i) <= data_in;
         end if;
      end process;
   end generate;
Voter_Block:
   VoterComponent
   generic map
   (
      number_of_bits
   )
   port map
   (
      data_in_0             => signal_data_in (0),
      data_in_1             => signal_data_in (1),
      data_in_2             => signal_data_in (2),
      data_out              => data_out
   );
end RTL;
```

Figure 4.16: TMR parameterized component for each one of the registers (part 2)

4.4.3 TMR+TR+CWSP Version or Robust to Direct and Indirect SEUs

The TMR+TR+CWSP version of the microprocessor was built following the scheme illustrated in Figure 4.17. The output of a combinational block, which originally is connected to the input of a unique register, is shared with three delay and CWSP blocks that have their outputs towards the inputs of the triplicated register. A voter block compares the three register outputs and results that of majority to the input of the next combinational block. Note that not only the registers are protected by the TMR but also the elements of the TR+CWSP scheme. Otherwise, these elements of the TR+CWSP scheme would be unprotected against SETs, as section 4.6.3 better explains.

Such TMR+TR+CWSP version was based on the Non-Protected and TMR versions detailed respectively in sections 4.4.1 and 4.4.2. Additionally, a CWSP VHDL component in accord to Figure 3.2 was created as Figure 4.18, Figure 4.20 and Figure 4.21 show. A dummy delay block was also modeled as a VHDL component. This component characterized in Figure 4.19, Figure 4.20 and Figure 4.21 was used by the synthesis tools to implement the target propagation time for the delay blocks. In other words, it is to define the place in the circuit for setting the timing constraints required by the synthesis tools to implement the delays. Details about these constraints are discussed in section 4.5 and 4.6.1.

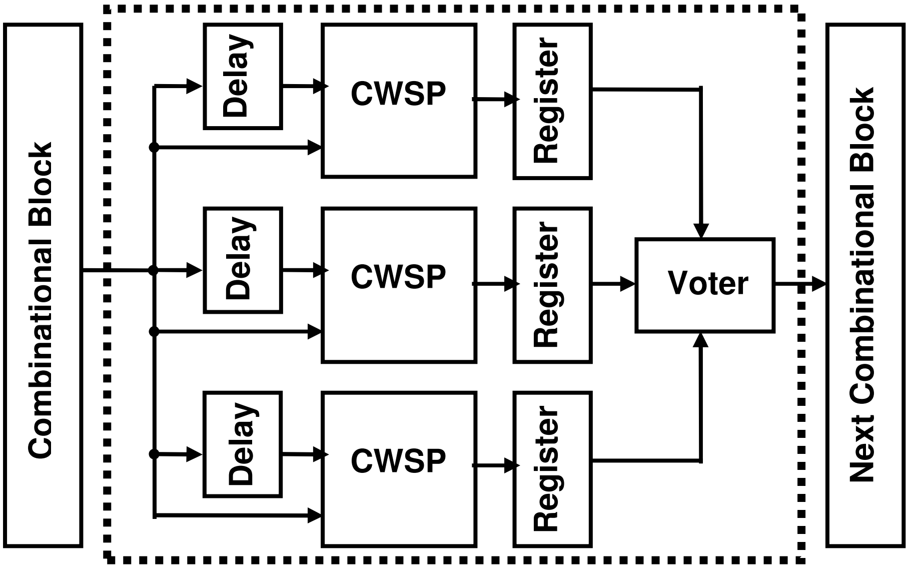


Figure 4.17: Combinational and Register blocks in the TMR+TR+CWSP version

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity CWSPcomponent is
   generic
   (
      number_of_bits          : integer
   );
   port
   (
      data_in                 : in std_logic_vector((number_of_bits - 1) downto 0);
      delayed_data_in         : in std_logic_vector((number_of_bits - 1) downto 0);
      data_out                : out std_logic_vector((number_of_bits - 1) downto 0)
   );
end CWSPcomponent;

architecture RTL of CWSPcomponent is
   signal signal_data_out     : std_logic_vector((number_of_bits - 1) downto 0);
begin
Register:
   process (data_in, delayed_data_in, signal_data_out)
   begin
      signal_data_out <= ((delayed_data_in and signal_data_out) or (data_in and
delayed_data_in) or (signal_data_out and data_in));
   end process;
   data_out <= signal_data_out;
end RTL;
```

Figure 4.18: CWSP block component

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity DelayComponent is
    generic
    (
        number_of_bits          : integer
    );
    port
    (
        data_in                 : in std_logic_vector((number_of_bits - 1) downto 0);
        data_out                : out std_logic_vector((number_of_bits - 1) downto 0)
    );
end DelayComponent;

architecture RTL of DelayComponent is
begin
DelayBlock:
    process (data_in)
    begin
        data_out <= data_in;
    end process;
end RTL;
```

Figure 4.19: Delay block component

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

package fault_tolerance_mechanisms is

    component VoterComponet
        generic
        (
            number_of_bits  : integer
        );
        port
        (
            data_in_0       : in std_logic_vector ((number_of_bits - 1) downto 0);
            data_in_1       : in std_logic_vector ((number_of_bits - 1) downto 0);
            data_in_2       : in std_logic_vector ((number_of_bits - 1) downto 0);
            data_out        : out std_logic_vector ((number_of_bits - 1) downto 0)
        );
    end component;
```

Figure 4.20: Package of the fault-tolerance mechanisms (2) (part 1)

```
   component DelayComponet
      generic
      (
         number_of_bits   : integer
      );
      port
      (
         data_in          : in std_logic_vector ((number_of_bits - 1) downto 0);
         data_out         : out std_logic_vector ((number_of_bits - 1) downto 0)
      );
   end component;
   component CWSPcomponent
      generic
      (
         number_of_bits   : integer
      );
      port
      (
         data_in          : in std_logic_vector((number_of_bits - 1) downto 0);
         delayed_ data _in: in std_logic_vector((number_of_bits - 1) downto 0);
         data_out         : out std_logic_vector((number_of_bits - 1) downto 0)
      );
   end component;
end package fault_tolerance_mechanisms;
```

Figure 4.21: Package of the fault-tolerance mechanisms (2) (part 2)

The TMR+TR+CWSP version of the unique reusable component detailed in Figure 4.11 is presented in Figure 4.22 and Figure 4.23. The CWSP and delay block components were instantiated in the register component from Figure 4.15 and Figure 4.16.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
library work;
use work.fault_tolerance_mechanisms.all;

entity RegisterComponent is
   generic
   (
      number_of_bits          : integer
   );
   port
   (
      clock                   : in std_logic;
      data_in                 : in std_logic_vector((number_of_bits - 1) downto 0);
      data_out                : out std_logic_vector((number_of_bits - 1) downto 0)
   );
end RegisterComponent;
```

Figure 4.22: TMR+TR+CWSP component for each one of the registers (part 1)

```
architecture RTL of RegisterComponent is
    type type_data_in is array (0 to 2) of std_logic_vector((number_of_bits - 1)
downto 0);
    signal delayed_signal       : type_data_in;
    signal tr_signal            : type_data_in;
    signal signal_data_in       : type_data_in:=(others => (others=>'0'));
begin
Redundant_Registers:
    for i in 0 to 2 generate
    Delay_Block:
        DelayComponent
        generic map (number_of_bits)
        port map (data_in, delayed_signal(i));
    CWSP_Block:
        CWSPcomponent
        generic map (number_of_bits)
        port map (data_in, delayed_signal(i), tr_signal(i));
    Register:
        process (clock, data_in)
        begin
            if (clock'event and clock = '0') then
                signal_data_in(i) <= tr_signal(i);
            end if;
        end process;
    end generate;
Voter_Block:
    VoterComponent
    generic map (number_of_bits)
    port map (signal_data_in(0), signal_data_in(1), signal_data_in(2), data_out);
end RTL;
```

Figure 4.23: TMR+TR+CWSP component for each one of the registers (part 2)

The kind of CWSP element implemented at all inputs of the CPU registers was the called Identity block suggested by (NICOLAIDIS, 1999) and shown in Figure 3.2. However, logic blocks of CWSP like those illustrated in Figure 3.5 could be used to replace the last logic gate of the target combinational block (the nearest gate from the output of the combinational block). It could decrease penalties in area, performance and power. Nevertheless, the implementation and adaptation in the RT-level description of the design would be onerous. The original combinational blocks from the Non-Protected version would be modified. Furthermore, there could be a different logic block of CWSP for each one of the original combinational blocks.

4.5 Back-End Physical Design

The physical design steps are well-defined problems with some complexity that typically require CAD resources in order to solve them. By this reason, the physical design flow depends so much on the available EDA tools. Table 4.1, Table 4.2 and Table 4.3 summarize all steps, including also logical design steps, of the design flow presented in Figure 4.5 by means of the tools used in each one. The tables show the CAD resources like platforms and commands utilized in the designs of this work.

Additionally, software scripts based on these tables were created with the aim of organizing and easily executing command sets of the design steps.

In the front-end logical design of this work, VHDL codes at the RT level were developed (**step 1** from Table 4.1), verified by simulation (**steps 2 and 3**) and afterwards logically synthesized (**step 6**). It was supported by a logic simulator from (MENTOR, 2004) and a logic synthesizer that provide, as output results, logic descriptions of the target circuits with no physical information.

Table 4.1: The developed design flow (part 1)

Design Step		**Tool**		**Platform (Command)**
		Kind	**Command**	
1	VHDL Description of the Target Circuit	Any Editor	-	-
2	Compilation of the Target VHDL Description	Compiler	vlib vcom	Mentor ModelSim (vsim)
3	Behavioral Simulation	Simulator	vsim run	
4	Technological Information Setup	Reader	read_tlf read_lef	Cadence PKS (pks_shell)
5	Importation of the Target VHDL Description and I/O Pad Cells	Reader	read_vhdl	
6	Logic Synthesis	Synthesizer	do_build_generic	
7	Constraint Setting	Reader	set_operating_condition set_wire_load_mode set_port_capacitance set_drive_cell set_clock ideal_clock set_clock_root set_path_delay_constraint set_floorplan_parameters set_clock_tree_constraints	
8	Technological Mapping	Synthesizer	do_optimize	
9	Floorplanning	Floorplanner		
10	Placement	Placer		
11	Clock-Tree Generation	Placer	do_build_clock_tree	
12	Global Routing	Router	do_route	
13	Area Analysis	Analyzer	report_area	
14	Timing Analysis	Analyzer	report_timing	
15	Power Analysis	Analyzer	report_power	
16	Preliminary Circuit Extraction	Analyzer	write_sdf	
17	Netlist Generation	Synthesizer	write_verilog write_vhdl	

The back-end physical design starts by including just physical information in these resulting logic descriptions. Such physical information comes from a technology library.

In this work, it was the AMS 0.35 µm CMOS technology that uses 4 layers of metal (AUSTRIAMICROSYSTEM, 2003).

By following the steps of the design flow detailed in Table 4.1, at **steps 4 and 5**, the technological information, the I/O pad cells and VHDL descriptions of the target circuits are loaded in a software platform. Such platform labeled as PKS (CADENCE, 2002) was utilized for the initial development of the physical design. Note, however, that **step 6** is a logical design step. It was arranged together with the physical design steps in order to make easy the development of the design. This approach was feasible because the PKS software platform has also integrated a logic synthesizer.

At **step 7**, initial design constraints are defined. The same initial constraints were used in the three microprocessor design versions mentioned in section 4.4. The operating conditions of the circuits were set in accord to the typical options of the technology library. These options consider a temperature of 25 °C and a power supply (vdd) of 3.3 V. All input ports of the circuit versions were connected to output buffers for modeling the drive capability of external drivers. These output buffers are BU24P cells from the target technology. It is a pad-limited output buffer cell with the strongest drive strength of the library (around 24 mA). Furthermore, capacitances based on the input loads from input buffers were specified at all output ports of the circuit versions. These input buffers are ICP cells that have input capacitance of 4.737090 pF. It is a pad-limited CMOS input buffer cell which can provide the typical input capacitance of a pad. An initial timing constraint at the clock period used for the timing analysis was 333 ns. This preliminary value comes from the typical clock period of a commercial system that: uses the target architecture of this work; and respects the lower speeds required by the compatible data and program memories habitually used in such applications (FREESCALE, 2002). The falling edge of the clock was set preceding the rising edge, since the architectures were designed by using the negative edge of the clock. As mentioned in section 4.4.3, delay constraints were also set in order to implement the buffers for the delay blocks. More details about the values used in these delay blocks are discussed in section 4.6.1. Other constraints were set at this step, such as the floorplan and clock-tree constraints that are detailed in the following paragraphs.

At **step 8**, the generic logic cells of the logic descriptions are mapped to standard cells of the target technology. At **step 9**, the floorplanning step is performed in order to estimate sizes and set the initial relative locations of the blocks in the IC (SMITH, 1997). It also allocates the space for the clock and power wiring and decides on the location of the I/O, power and ground pads. Concerning the initial floorplan parameters set at **step 7** and illustrated in Figure 4.24, the initial aspect ratio of 1 was set as a constraint. In other words, the chip area should initially have a square shape (y = x, in Figure 4.24). The left, right, top and bottom distances (x0, x1, y0 and y1 in Figure 4.24) from the I/Os to the core were defined all equally as 746.200 µm. This dark gray area detailed in Figure 4.24 between the I/Os and the core is the optimized space for placing and routing symmetrically the power and ground rings and 52 pad cells required by the three microprocessor versions. Values lower than 746.200 µm were tried but the tools could not attain their aims successfully. The initial utilization of the core rows (look at Figure 4.24 the core rows, where the standard logic cells are placed) was set to reach around 70 % of the row area. The remaining row area of 30 % is for the routing finishing successfully and it is just occupied with special cells known as filler cells. This initial utilization was the highest value which the floorplanner succeeds for the three microprocessor versions. The number of core rows is determined by the floorplanner in accord to the design. More details about the initial floorplan parameters are presented in section 6.1.1.

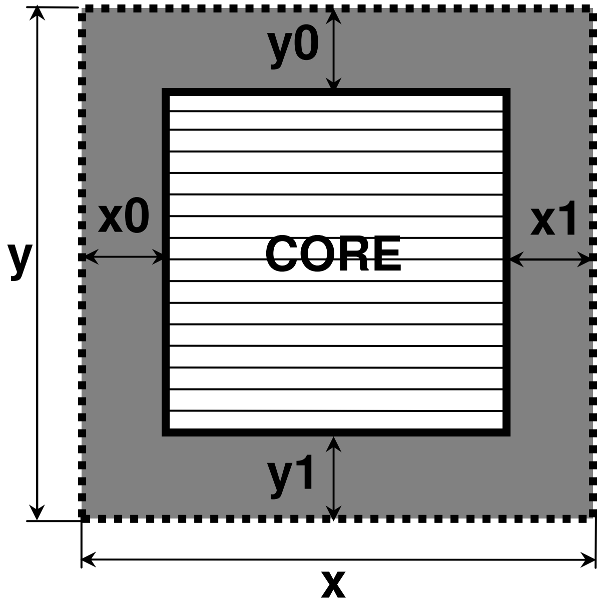


Figure 4.24: Initial floorplan parameters for the three microprocessor versions

At **step 10**, the placement tool defines the locations of the standard cells within the IC and sets aside the space for the interconnect to each standard cell (SMITH, 1997). The placement assigns each standard cell to a position in a row or core row as mentioned in previous paragraph. Note that steps 8, 9 and 10 are integrated at the same software command. In order to minimize the circuit path delays, this approach tries to optimize the choice and placement of the standard cells based on a circuit timing analysis.

At **step 11**, in order to avoid clock skew, a clock tree is built by an EDA tool based on some initial constraints set at **step 7**. In other words, buffers or inverters are added in the wires of the clock network for balancing the clock distribution. A minimum clock delay of 3.00 ns and a maximum of 3.50 ns were initially assigned. In addition, a maximum skew was set to 0.32 ns. Such values were deduced from some preliminary simulation experiments with the target robust architecture using higher values for these clock-tree constraints.

At **step 12**, the global routing tool determines where the interconnections between the placed standard cells and blocks will be situated (SMITH, 1997). Only the routes to be used by the interconnections are decided at this step, not the actual locations of the interconnections within the wiring areas. At this step 12, a physical design information file known as Design Exchange Format (DEF) is created to be used at following design steps.

At **steps 13, 14 and 15**, analysis tools provide preliminary estimated results in area, performance and power before the final layout adjustments. These design results can be seen through report files generated by the tools. An IC preliminary view can be also seen at these steps, as Figure 6.2, Figure 6.3 and Figure 6.4 in chapter 6 illustrate for the three microprocessor versions. Area results show the total circuit area but they do not detail about the wiring and routing issues. Performance results are estimated by a static timing analysis tool. Power results are based on default values defined by a power analysis tool at primary inputs and outputs of circuit sequential elements. As emphasized in section 4.3.1, these pre-layout design results are not as accurate as the post-layout design results, but they can allow an initial evaluation of the IC design.

At **step 16**, preliminary delay information of logic gates and interconnects based on the technology library was generated and stored in a Standard Delay Format (SDF) file. At **step 17**, post-synthesis structured descriptions (VHDL and Verilog netlists) were created. They are constituted of gates relating to the standard cells from the technology library.

In Table 4.2, **steps 18, 19 and 20** perform the pre-layout verification simulations of the circuits, as the design flow in Figure 4.5 shows. It uses the pre-layout SDF information from step 16.

Table 4.2: The developed design flow (part 2)

Design Step		**Tool**		**Platform (Command)**
		Kind	**Command**	
18	Technology Library Compilation	Compiler	vlib vcom vmap	Mentor ModelSim (vsim)
19	Netlist Compilation	Compiler	vlib vcom	
20	Pre-Layout Gate-Level Simulation	Simulator	vsim (with sdf file) run	
21	Technological Information Setup	Reader	INPUT LEF INPUT CTLF	Cadence SE P&R (seultra)
22	Netlist Importation	Reader	INPUT VERILOG INPUT DEF	
23	Insertion of Corner Cells, Power and Ground Pad Cells	Reader	INPUT DEF ADD ROW	
24	I/O Placement	Placer	IOPLACE	
25	Power Ring Planning	Planner	CONSTRUCT RING	
26	Insertion of Filler Cells	Placer	SROUTE ADDCELL	
27	Detailed or Final Routing	Router	CONNECT RING WROUTE	
28	Area Analysis	Analyzer	REPORT SUMMARY REPORT WIRES	
29	Circuit Extraction	Extractor, Analyzer	REPORT RC (rspf file) REPORT DELAY (sdf file)	
30	Netlist Generation	Synthesizer	OUTPUT VERILOG OUTPUT DEF	
31	GDSII-File Generation	Synthesizer	OUTPUT GDSII	

At **steps 21 and 22**, the technology information and the netlists, DEF and Verilog files, from step 12 and 17 are read by the physical design tool. In addition to the I/O pad cells, inserted at step 5 for the functional pins of the IC, other special pads are inserted at **step 23**. At **step 24**, these special cells defined as corner cells, power and ground pad cells are placed in the designs. The corner cells give continued to the power and ground interconnects (pad rings) among the pad cells that are placed around the core.

At **step 25**, two power rings are constructed around the core area of the circuit. One of them for the power supply (vdd) and the other one for the ground (gnd). Both rings were built with a width of 75.000 µm. At **step 26**, filler cells are added in order to fill gaps among the standard logic cells placed on the core of the design and among corner and pad cells. As emphasized at step 9, such filler cell areas are the spaces for the routing finishing successfully.

At **step 27**, the power rings are initially connected to the cells. After this, all wires are routed by using a router tool. In other words, the standard logic cells, corner and pad cells are joined by wires or interconnections. It includes also the routing of the clock, power and ground interconnections. The width, mask layer and exact location of the interconnections are defined by the router (SMITH, 1997). These interconnections can be built by 4 different layers of metal. As emphasized in previous paragraphs, it is due to the characteristics of the technology used in this work. The final layout illustrations of the target circuits can be also seen at this step, as Figure 6.2, Figure 6.3 and Figure 6.4 in chapter 6 show for the three microprocessor versions.

The length and position characteristics of each interconnect for each net is known after the detailed routing (SMITH, 1997). Thus, at **step 28**, results in area are generated with the wiring details. And at **step 29,** parasitic capacitance and resistance associated with each interconnect, via and contact can be calculated. It is generated by a circuit-extraction tool that provides a Reduced Standard Parasitic Format (RSPF) file. Additionally, a SDF file can be also generated based on the post-layout information. At **steps 30 and 31**, post-layout netlists (Verilog, DEF and GDSII files) are created for the next design steps.

At **step 32** from Table 4.3, the GDSII files from the previous step are imported to a software platform able to check the circuits. At **step 33**, a Design-Rule Check (DRC) is performed to ensure that nothing has gone wrong in the process of assembling the standard logic cells and routing (SMITH, 1997). It checks for shorts, spacing violations, or other layout design-rule problems between standard logic cells. Other check like the Layout Versus Schematic (LVS) could also be performed to ensure that the extracted electrical schematic from the physical layout is the same to the designed netlist or HDL code. Another usual check that could be used is a formal verification. It would extract a Boolean description of the function of the layout and would compare that to a known good HDL description.

At **steps 34 and 35**, a post-layout verification simulation of the circuits at the gate level is performed by using the post-layout SDF information. In order to estimate the dynamic power consumption, at **steps 36 and 37**, the switching activities of the circuits are analyzed by counting and collecting changes of state on all nodes of the circuits. In addition, the toggle coverage allows a view of the testbench effectiveness used in the verification experiments. After the simulations of verification based on benchmarks, Value Change Dump (VCD) files, which contain the switching activities of the circuits, are created for the next design steps. At **steps 38, 39, 40, 41 and 42**, final post-layout results in area, performance and static and dynamic power consumptions are generated based on the post-layout information represented by Verilog, DEF, RSPF, SDF and

VCD files. The VCD files were converted to Toggle Count Format (TCF) files due to the requirements to perform the power analysis through the used software platform. By using report files, the EDA tools arrange all final design results. They are presented in chapter 6 of this work.

Table 4.3: The developed design flow (part 3)

Design Step		Tool		Platform (Command)
		Kind	Command	
32	GDSII-File Importation	Reader	From a Virtuoso Command Interface Window (Diva Tool)	Cadence IC (icfb)
33	DRC	Checker		
34	Netlist Compilation	Compiler	vlib vcom	Mentor ModelSim (vsim)
35	Post-Layout Gate-Level Simulation	Simulator	vsim (with sdf file)	
36	Toggle-File Generation	Synthesizer	vcd file run	
37	Toggle Coverage	Analyzer	toggle report	
38	Technological Information Setup	Reader	read_tlf read_lef	Cadence PKS (pks_shell)
39	Netlist Importation	Reader	read_verilog read_def read_wdb read_spf read_sdf	
40	Timing Analysis	Analyzer	report_timing	
41	Toggle-File Importation	Reader	lpsvcd2tcf.exe read_tcf	
42	Power Analysis	Analyzer	report_power	

4.6 Some Fault-Tolerant Circuit Characteristics

Some characteristics of the robust microprocessor by using the TMR+TR+CWSP scheme are presented below:

4.6.1 The Maximum Width of SETs

As defined in section 3.2.2, there is a maximum width of SET pulse arisen at the output of the combinational block that is able to be mitigated by the TR+CWSP scheme. It is based on the propagation time of the CWSP and Delay blocks from Figure 4.17. The delay of the CWSP blocks are based on their logic circuits, otherwise the Delay blocks are defined by any sequence of buffers or inverters. By this reason, the Delay blocks are used in the design as adjustment elements to achieve the target maximum SET width. In this work, the delays of these blocks were implemented through timing constraints readable by the synthesis tools, as emphasized previously in section 4.4.3 and 4.5.

In section 2.1.2.3 was discussed that in micrometer technologies the duration of a SET pulse is typically a few hundreds of picoseconds. Note that in the Non-Protected microprocessor version, it might diffuse itself and make SEs. The target technology, AMS 0.35 µm (AUSTRIAMICROSYSTEM, 2003), establishes delays of basic standard logic gates typically around 10 ps and 2 ns. See that a typical SET (e.g. 100 ps) can propagate itself through a basic standard logic gate (e.g. 10 ps), since pulses of widths larger than the delay of a logic gate usually are not able to suffer electrical masking. Based on these issues, in the TMR+TR+CWSP microprocessor design, the Delay blocks from Figure 4.17 were defined in order to achieve a maximum width of SET able to be mitigated around 1 ns. Such value covers slackly the requirements of a typical SET occurrence on this robust circuit based on this target technology. However, adjusting this width for another desired value is easily made by using CAD resources. If a larger width of SET is required, larger overheads are attained. Otherwise, optimized width can be achieved by decreasing its value. Thus, lesser overheads are attained, since fewer buffers or inverters are used to implement the Delay blocks.

Furthermore, as discussed in section 3.2.2, the TR+CWSP approach also mitigates timing faults. Therefore, enlargements of the circuit paths up to around 1 ns will be mitigated. However, in case there are such faults, the robust circuit may not be able to mitigate SET pulses any more. It is because there will not be enough time slack to mitigate them.

There is another important related issue detailed also in section 2.1.2.3. A SET pulse that arises inside a combinational block can result at the output of this block with a width larger than the original one. It can occur due to the different delays of the propagation paths of the combinational circuit. In this case, the TR+CWSP scheme may not work because the allowed maximum width of SET will be overcome. However, it is a low-probability event. Experiments were done by (ANGHEL; ALEXANDRESCU; NICOLAIDIS, 2000-b) in order to evaluate such effects. The results showed that the scheme achieves a high error correction efficiency (around 97 %) and it can be improved further by increasing the Delay blocks.

4.6.2 Multiple SEUs

A SET can start at any moment inside a clock cycle period of microprocessor software applications. The TR+CWSP technique mitigates SETs inside each clock cycle. In case of several SETs (more than one SET) occur inside a clock cycle, the technique may not work. It is because there could be some confusion in the comparison between the delayed and non-delayed outputs of the affected combinational circuit. In sections 2.1.2.3 and 2.1.2.4, a low-probability multiple event is detailed. A unique SET pulse on an internal node of a combinational circuit can sometimes create multiple pulses at an output bit of this combinational circuit as a result of the delay difference among the circuit paths.

Furthermore, in accord to the topology of the circuit, a unique SET can also achieve a unique or several bits of the registers (i.e., a unique or several potential indirect SEUs) or even not cause any consequence (i.e., an electrical, logical or latching-window masking). At any moment during the use of the microprocessor, several SETs can also occur at the same time on any bit of a combinational or sequential circuit (i.e., several potential indirect or direct SEUs). The TMR+TR+CWSP scheme protects all bits of the registers, hence MBUs or SEUs can occur at same time and on any data bit of the registers that they will be mitigated. However, SEUs cannot occur at the same time on the redundant parts of a triplicated register, as section 4.6.3 explains.

4.6.3 SETs on the Elements of the Fault-Tolerance Mechanisms

As affirmed in section 3.1, the fault-tolerant systems based only on the TMR protection are susceptible to indirect SEUs. It is because a SET can propagate itself up to the three registers of the triplicated one. Thus, in case of a triple indirect SEU, the voter block is not able to detect differences among the values stored in the triplicated register. As result of this, it is mandatory the use of dedicated fault-tolerance techniques, like the TR+CWSP scheme, in order to avoid indirect SEUs.

On the other hand, SET occurrences on the Delay or CWSP blocks are not mitigated for the TR+CWSP mechanisms as those SETs that can occur on combinational blocks. By this reason, note in Figure 4.17 that the branch formed by the Delay, CWSP and Register blocks is triplicated based on the TMR principles. Thus, a SET occurrence on one branch of the triplicated one could provoke an indirect SEU on one register of the triplicated one. However, this error would not propagate because the Voter block would proceed in the same way if a direct SEU had occurred on that register. In other words, it would elect the value of majority among the three stored in the triplicated register.

As defined in section 3.1, considering the three registers of a triplicated register, observe that the Voter block requires at least two registers without errors to elect a correct output. Therefore, for the mechanism working correctly, direct SEUs or SET pulses, which potentially would cause indirect SEUs on the registers, cannot occur at the same time on two or three branches of the triplicated one.

The Voter block is a combinational circuit that can be considered as a part of the Next Combinational Block illustrated in Figure 4.17. Thus, if a SET happened on the Voter block, it would be mitigated in the same way as those SETs that can occur on combinational blocks.

4.6.4 Other Remarks

Another issues related to design of the fault-tolerant circuit are emphasized below:

- There are combinational circuits that do not achieve sequential elements, i.e., those combinational circuits in which their outputs are directly connected at outputs of the core (output pads of the chip). Such combinational circuits are susceptible to SET effects. In the target microprocessor of this work, it is the case in two situations. The outputs of the registers called Address and ALU are connected to combinational circuits (voter blocks) that achieve directly output pads. Thus, SET effects on these combinational circuits may cause undesired transient results at their outputs. These circuits correspond to a small part of the core area, therefore such effects have a low-probability occurrence. On the other hand, a fault-tolerance approach like the TR+CWSP scheme could be applied on this circuits to reduce these effects;
- SET pulses arisen at input pins of the fault-tolerant system could be also taken account of. These pulses can be erroneously considered as good inputs to internal registers. The scheme TR+CWSP is able to mitigate these events, however the allowed maximum width of SET must be lesser than a typical pulse at input pins of the chip such as data pins or interruption pins. It is to prevent bad detection by part of the TR+CWSP scheme. Such condition is not difficult to meet for the target architecture of this work. The maximum SET width is around 1 ns as detailed in 4.6.1. It will be lesser than the usual minor widths of signals

required by the inputs of this CPU and provided by the outputs of its conventional external peripherals. For this architecture, the widths of these signals are usually larger than 10 ns. In future or advance architectures, this condition may be critical;

- The combinational circuit (buffers and inverters) that defines the clock tree of the system is susceptible to SETs. As this circuit does not achieve data inputs of registers but specific clock inputs of the registers, it requires another protection approach. The effects of a SET on this circuit can be a larger clock skew for a certain register. Thus, an unbalance on the clock network can be characterized and registers may be induced to store values wrongly;
- The target architecture in this work does not present latches. Note, however, that these components are memory elements too. Consequently, in robust IC designs composed also by latches, dedicated fault-tolerance techniques should be also used to protect such elements. These techniques can be similar to those used for flip-flops;
- In this work, the target system to protect was the microprocessor core (CPU). In a design scenario of a fault-tolerant microcontroller similar to Figure 4.1, the volatile memory resources vulnerable to SEUs are basically registers dispersed on the CPU and a data memory (RAM). The program memory usually is non-volatile and in principle it is not susceptible to SEUs because requires typically higher currents to modify its bits. About the microcontroller protection, the CPU registers would follow the same TMR+TR+CWSP approach implemented in this work. The data memory could be protected against direct SEUs by using parity or EDAC codes. As emphasized at the beginning of this chapter, it is because such codes are relatively cheaper for memory arrays similar to the data memory. Regarding possible indirect SEUs, it could be mitigated by using the TR+CWSP approach applied on the elements of the data memory or another specific technique for memory arrays such as that presented in (HENES-NETO; WIRTH; KASTENSMIDT, 2006).

5 DESIGN VERIFICATION SIMULATION OF A ROBUST MICROPROCESSOR

Incorrect operations or errors in a digital system can be detected by using a testing scheme. Some typical errors that may occur in this kind of system can be classified (ABRAMOVICI; BREUER; FRIEDMAN, 1990). **Design errors** are, for instance, incomplete or inconsistent specifications; incorrect mappings between different levels of design; or violations of design rules. **Fabrication errors** are those which occur during fabrication due to, for example, wrong components; incorrect wiring; or short caused by improper soldering. **Fabrication defects** are not directly attributable to a human error, rather, they result from an imperfect manufacturing process such as short, opens, improper doping profiles, mask alignment errors, poor encapsulation, etc. **Physical failures** occur during the lifetime of a system due to component wear-out or environment factors like temperature, humidity, vibrations, electrical noise and the radiation-induced effects. Fabrication errors, fabrication defects and physical failures are consequences of physical faults that can be permanent, intermittent or transient.

The initial testing of an IC is performed by simulation within a CAD environment. At this stage, the designer is verifying the functionality and the performance of the intended circuit (GROCHOWSKI et al, 1997). In fact, the designer is looking for eventual design errors. Many of the production tests, during the IC manufacture, are based on this initial testing.

In this work, the testing experiments are concerned about such design errors. It is because these errors precede the fabrication of the IC. As mentioned in chapter 1, the steps of the IC manufacturing stage will be performed on the future. Moreover, some kinds of physical failures (i.e., the Soft Errors (SEs) due to direct or indirect SEUs) need be also considered in the testing experiments. The effects of these physical faults on the IC can be represented by logical faults based on the fault model discussed in chapter 2.

The fault-tolerance techniques implemented in the target microprocessor as on-line testing mechanisms (self-checkers) need be also tested by a different testing experiment, such as a fault injection, in order to avoid eventual design errors in their circuits. The issue is that the absence of faults in the circuit under test, i.e., in the target microprocessor for this work, might hide design errors in the self-checking circuits. It is because testing schemes like these self-checkers work different when the circuit under test is at presence of faults. Thus, a means of performing such tests is emulating the physical failures that these on-line testing circuits are able to mitigate through a fault injection experiment.

The design verification testing in order to detect design errors is typically performed by a testing experiment that uses a model of the designed system. Design verification simulation or logic simulation usually determines the time evolution of the signals in the model as responses to applied input sequences (ABRAMOVICI; BREUER; FRIEDMAN, 1990). The verification is done by comparing the results obtained by simulation with the expected results provided by the specified design behavior. In the present work, a model of the target microprocessor was designed in accord to sections 4.4 and 4.5. In sections 5.1, 5.2, 5.3 and 5.4 is explained details about the design verification simulations developed for this target circuit.

In addition to these experiments to detect eventual design errors, naturally, there are others such as those verification methods briefly discussed in section 4.5: DRC, LVS, formal verification, etc. Furthermore, there are some testing steps performed during or after an IC fabrication that are mandatory. The called production tests usually are based on test vectors applied at circuit inputs by tester equipments. They have the purpose of checking essentially whether there are permanent physical failures, fabrication defects or even fabrication errors (when the IC is part of a system on some kind of board). By means of the difference between the behavior of the target circuit in the presence of a fault and the fault-free circuit behavior, one can derive a test for that fault (ABRAMOVICI; BREUER; FRIEDMAN, 1990). Basically, the outputs of the fault-free circuit are compared with outputs of the faulty circuit. Such test can be also performed by simulation as a further verification step to find design errors before an IC fabrication. Other post-fabrication testing approaches are redesigning circuit parts to improve the accessibility to hard-to-test elements as a design-for-testability (LUBASZEWSKI; HUERTAS, 2004). Another approach of design-for-testability such as Built-in Self-Tests (BISTs), tester equipments to apply test vectors are not required due to the on-chip test generation and evaluation. BISTs are performed through signals specifically created to test the circuit, unlike the on-line self-checkers that are also classified as a design-for-test approach but they are performed through functional signals of the circuit under test.

5.1 Types of Design Verification Simulation

By means of simulator tools, the circuits can be simulated to verify their characteristics at different design levels. Thus, the level of simulation corresponds to the level of modeling employed to represent the simulated system (ABRAMOVICI; BREUER; FRIEDMAN, 1990). From a high-level to a low-level simulation, the simulations become more accurate, but they also become progressively more complex and take longer to run (SMITH, 1997):

- **Behavioral simulation** considers circuits modeled with black boxes or components without delays. In each clock cycle, the signals of the circuit are updated at the clock event, not during the clock cycle. It occurs before the logic synthesis of the circuit without any technological or physical information;

- **Functional simulation** or unit-delay simulation ignores timing and includes a unit of delay. It sets delays in the circuit components to a fixed value like 1 ns. This simulation occurs after the logic synthesis and technological mapping;

- **Gate-level simulation** or logic simulation can also be used to check the timing performance of an IC. A logic gate or logic cell is treated as a black box usually modeled by a function that determines the delay through the cell. This simulation can be called as **pre-layout simulation** when includes logic-cell delays but no interconnect delays. Setting all delays to a unit value, it becomes as a functional simulation. On the other hand, this simulation can also be called as **post-layout simulation** when considers, after physical design, delays of logic cells and also interconnects;

- **Switch-level simulation** considers circuits modeled with transistors as switches (on or off). This simulation may use a large possible set of discrete voltage values or the value of a node may be allowed to vary continuously. It can provide more accurate timing predictions than the gate-level simulation, but takes longer time of execution;
- **Transistor-level simulation** or circuit level simulation is perhaps the most accurate, but also the most complex and time-consuming. It requires models of transistors, describing their non-linear voltage and current characteristics. This simulation usually is used to analyze the analog, rather than the digital, behavior of circuit voltages.

Different parts of the system can be simulated by different levels of simulation (SMITH, 1997). Critical blocks can be simulated by more accurate low-level simulations. On the other hand, as mentioned above, there is a cost in run time against accuracy. Low-level simulation like switch-level and transistor-level simulations take longer time and are almost impracticable for large circuits such as a microprocessor. Indeed, switch-level and transistor-level design simulations are often used in full-custom designs of specific small circuits.

5.1.1 The Developed Types of Design Verification Simulations

In this work, as detailed in section 4.3.1 by the design flow from Figure 4.5, three different stages of simulations for verification were developed for each one of the three microprocessor versions (Non-Protected, TMR and TMR+TR+CWSP versions designed in accord to sections 4.4 and 4.5):

- **Behavioral simulation** simulates the behavior of the VHDL codes that describe the microprocessor versions at the RT level;
- **Pre-layout gate-level simulation** simulates logically the resulting VHDL netlists of logic gates that characterizes the microprocessor versions after the synthesis steps. It uses the SDF information from before the final routing;
- **Post-layout gate-level simulation** simulates again logically the netlists of gates but uses the SDF file with the final routing information;

These three stages of simulation were performed by using a CAD logic simulator from (MENTOR, 2004). In order to run these design verification simulations, stimuli need be generated at the inputs or internal signals of the microprocessor versions. This task can be supported by a testbench, as section 5.2 discusses. Moreover, the resulting responses at the outputs or other signals of these circuits need be checked with the aim of detecting eventual design errors. By this reason, as section 5.3 presents, a check approach through a **functional testing** experiment was performed at these three stages of simulation. Additionally, only at the post-layout gate-level simulation, a **fault injection** experiment was done as section 5.4 details.

Since in this work the target circuit was a microprocessor that inherently denotes a complex circuit, a design flow based on high-level models and standard cells was mandatory as discussed. Thus, the switch-level and transistor-level simulations, cited at the beginning of this section 5.1, would be practically unfeasible due to the circuit complexity. Furthermore, the standard cells were in principle well designed by technology providers. Regarding the functional simulation, it would not be worth performing due to the characteristics of the available CAD platform. As explained in section 4.3.1, it was well replaced by a pre-layout simulation in order to develop a faster design flow.

5.1.1.1 Timing Verification by Static Timing Analysis

Timing verification targets at determining whether the timing constraints imposed to the design may be satisfied or not. It can be performed by using circuit simulation or by timing analysis (GÜNTZEL, 2000).

Vector-based simulations (or dynamic simulations), like the functional simulation and the gate-level simulation detailed at the beginning of this section 5.1, can check if the design functions correctly (SMITH, 1997). However, in order to find the longest delays of the circuits, they require test vectors that active the critical circuit paths. It is not usually a simple task by considering the current design complexity. To cope with this in an easier way, a static timing analysis is quite suitable.

A static timing analysis checks the timing performance based on circuit topology and technology information and timing analysis algorithms. It analyzes the logic in a static manner by computing the delay times for each path. It is static because does not require the creation of a set of test-vectors (stimuli) that, as mentioned above, would be onerous for a large circuit (SMITH, 1997).

In this work, in accord to section 4.3.1 by the design flow from Figure 4.5, two different stages of static timing analysis were developed for each one of the three microprocessor versions (defined in sections 4.4 and 4.5):

- **Pre-layout static timing analysis** estimates the performance through the critical path in the microprocessor models generated before the final routing;
- **Post-layout static timing analysis** estimates again the critical path delay of the circuits but after the final routing and circuit extraction.

Such analyses were performed through a CAD analyzer from (CADENCE, 2002). This timing-analysis tool basically identifies the longest delays among the paths of the circuit. It also checks the set-up and hold time requirements in accord to timing constraints. The tool reports all this information in text files.

5.2 Modeling of System for Simulation

In order to perform the testing experiments in the microprocessor versions of this work, whose models are characterized by VHDL codes or netlists detailed in chapter 4, a system like Figure 5.1 composed of the target **CPU** and other peripherals was modeled as a testbench in VHDL.

Based on this modeling of system, the CPU can execute machine codes of instructions compiled from a software application or benchmark. These program codes are allocated in another instance defined as the **program memory**. In accord to function of the executed instructions, the CPU requires the temporary use of memory resources to manipulate the instruction operations. These resources are defined in another instance as the **data memory**. The CPU accesses such program and data memories essentially by using an address bus and a data bus. The address bus indicates the desired position in the memory to be accessed by the CPU. On the other hand, based on these address bus, the data bus is used by the CPU to read/write codes from/to that position in the memory.

In addition, other peripherals support the design verification simulation: a **clock generator** provides the clock signal to the system; and an **interface device** that offers information about the signals of the system to a logic simulator. This information is treated by the simulator terminal that prints it in an interface window with the designer.

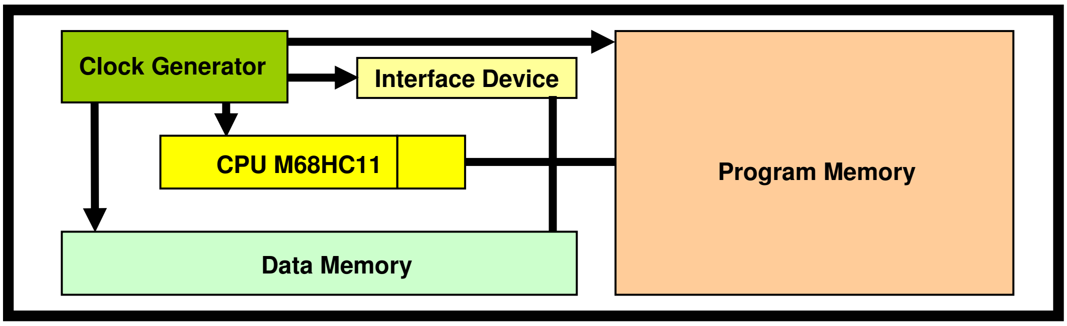


Figure 5.1: Diagram of the system modeled as a testbench in VHDL

5.2.1 Functional Behavior of the Modeled System

The testbench emulates the functional behavior of an actual system based on this CPU. All execution steps of a benchmark can be seen by means of the logic simulator. The benchmarks must be described in assembly language of the M68HC11 family (FREESCALE, 2002). Thus, they can be compiled by means of the standard assembler tool provided by Freescale. As a result, the assembler generates an output file that contains the program codes of the compiled benchmark. By using this output file, the VHDL testbench can access the program codes as whether it was the program memory of the system illustrated in Figure 5.1.

Figure 5.2 illustrates, through the main signals of the system, an instruction of a certain benchmark being executed. The illustrated instruction is the JSR that makes the current execution jumping to a subroutine. Its operation code (opcode) is 0xBD at the extended addressing mode. It uses 2 operands and 6 clock cycles. In this example, the operands are 0xF8 and 0x44 that represent, as 0xF844, the initial address of the subroutine.

The execution of this instruction starts at state 0x01 when the CPU requests to the program memory the address 0xF851. This address is based on the previous executed instruction and indicates the position in the program memory where opcode 0xBD is stored. This opcode is read by the CPU through the data bus. In the two following states (0x03 and 0x0F), the operands (0xF8 and 0x44) are read from the program memory in the same way that opcode 0xBD.

During state 0x1E, the CPU builds address 0xF844 to jump. Furthermore, the address to return (0xF854) after the subroutine execution is calculated and stored in the program counter (PC). At state 0x13, the address to return (0xF854) is saved in the stack. As the stack is allocated in the data memory, 0xF8 and 0x54 are written in the data memory by means of the write_data bus and the rw signal. The location in the data memory where 0xF8 and 0x54 are positioned depends on the current address (0x00FB) pointed by the stack pointer (SP). The SP always points to the next empty position in the stack. Thus, 0xF8 and 0x54 are written in the address 0x00FB and 0x00FA of the data memory.

At its last state (0x01), the JSR instruction finishes by putting the address 0xF844 on the address bus. It is to request to the program memory the opcode of the next instruction (0x36), i.e., the opcode of the first instruction of the subroutine.

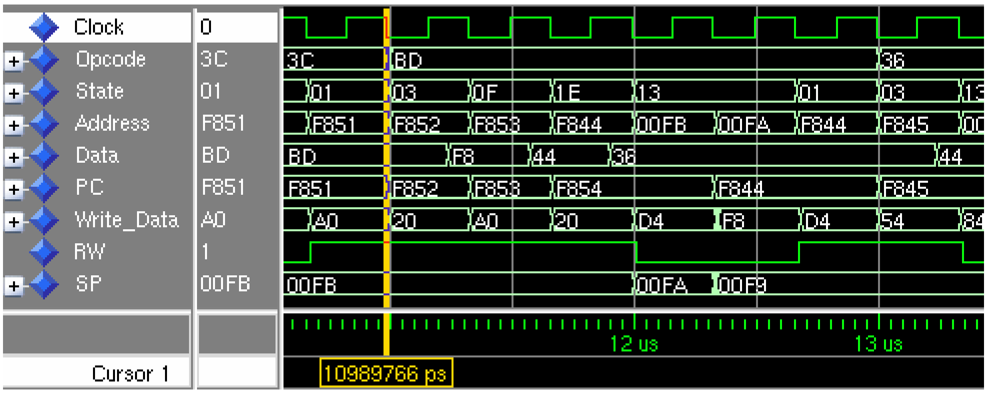


Figure 5.2: Execution of the JSR instruction by simulation

5.3 Functional Testing by Simulation

A functional testing experiment was performed at each one of the three simulation stages discussed in 5.1.1: behavioral, pre-layout gate-level and post-layout gate-level. The goal of this simulation experiment is testing functionally each one of the three designed microprocessor versions (Non-Protected, TMR and TMR+TR+CWSP) detailed in section 4.4.

The target circuits can be exercised through stimuli made by executions of benchmarks. It is possible by using the testbench presented in section 5.2. New benchmarks were created based on two software applications from (THIBAULT, 2000; FREESCALE, 2002). One of these applications converts hexadecimal codes to their ASCII characters and prints the output results in the interface device from Figure 5.1 to be seen by the designer. The other one application is a program for automatic control of a motor. It generates a PWM signal of constant frequency and variable duty cycle and it also prints output results like the current speed and direction in the interface device.

5.3.1 Benchmark Design

As these two base benchmarks detailed above do not contain all microprocessor instructions, the stimuli would not achieve the whole of the circuit. In this manner, the microprocessor versions would not be completely tested. In order to check the functions specified by all microprocessor instructions, five new benchmarks were created by including instructions that are not present in the two base benchmarks.

The benchmarks should use all 306 different microprocessor instructions through the six addressing modes. Note that thus all 18 existing microprocessor registers would be used too. It was not 308 instructions because two instructions of division are not implemented, as emphasized in sections 4.1 and 4.4.1.

The additional instructions in the new benchmarks must follow the functional coherence of the two base benchmarks. Any value manipulated by an instruction in one of these base benchmarks must be used by the first instruction added in one of the new benchmarks. As the target CPU employs a sequential instruction processing without any type of pipeline, a bundle of new instructions in sequence can be inserted to be tested. Logic links through the manipulation of values must be established among the first added instruction, its former and latter instructions and so on. The last instruction added in the new benchmark must be able to give continuity to the remaining processing without altering the functional execution of the base benchmark. It is to preserve the same output results calculated by this base benchmark. Indeed, no new features are inserted in the five new benchmarks but whether an added instruction not working correctly, the benchmark execution shall be broken.

All added instructions to be tested were arranged in five groups. Thus, each one of the five benchmarks tests a different set of instructions. It emulates a habitual characteristic of the target software applications that almost in totality do not use all microprocessor instructions. In addition, it divides the simulation time in parts facilitating the management of the design tasks.

Another benchmark issue is about the values of inputs and operands used by the instructions. It depends quite on the design specification of the software applications. Therefore, there can be many combinations due to different specifications of software applications. The option by values that have all their bits used by the instructions can be good. The coverage of all value possibilities would be an arduous design task.

5.3.2 Benchmark Analysis

A statistics resource to estimate the quality of the stimuli or test vectors induced by the created benchmarks is to use a **toggle test**. It checks which circuit nodes toggle as a result of applying input stimuli. There is a strong correlation between high-quality test vectors and high **toggle coverage** (SMITH, 1997). Table 5.1 shows the toggle coverage for each one of the five benchmarks labeled as HNO, HLC, HLB, AOR and ALO on each one of the three microprocessor versions.

Table 5.1: Toggle coverage of the benchmarks on the microprocessor versions

Circuits:	Non-Protected		TMR		TMR+TR+CWSP	
Total Node Count:	10988		12480		17155	
Benchmarks	Toggled Node Count	Toggle Coverage	Toggled Node Count	Toggle Coverage	Toggled Node Count	Toggle Coverage
HNO	5430	49.42 %	6619	53.04 %	9468	55.19 %
HLC	5756	52.38 %	6839	54.80 %	9644	56.22 %
HLB	5776	52.57 %	7017	56.23 %	10064	58.67 %
AOR	6228	56.68 %	7794	62.45 %	10938	63.76 %
ALO	6288	57.23 %	7942	63.64 %	11261	65.64 %

Note in this Table 5.1 that the toggle coverage does not achieve 100 % especially because not all microprocessor instructions are present in each one of the benchmarks. However, as all these microprocessor instructions were arranged in the five benchmarks, there is a trend for the overall toggle coverage of this functional testing. By using the five benchmarks in sequence, as a unique program, the toggle coverage shall approach to 100 %. It is an excellent sign of which these testing experiments achieve almost the whole of the target circuits. It will not attain 100 % due to the values of the inputs and operands or even due to unreachable parts of the circuits.

5.3.3 Benchmark Simulation Characteristics

About the clock cycles used by the five benchmarks, Table 5.2 shows the total number of clock cycles required to run once the complete benchmark execution. This table shows also the number of clock cycles to execute the program main loop by considering a set of fixed inputs. It is to be used by the power analysis that requires the periodic part of the benchmark. In addition, by using a typical clock period of 333 ns as initially defined for the timing analysis in section 4.5, Table 5.2 illustrates the equivalent times to the clock cycles (run time).

Table 5.2: Clock cycles of the benchmarks

	Simulation of the Program Main Loop		Total Simulation	
Benchmarks	Clock Cycles	Run Time	Clock Cycles	Run Time
HNO	393	130.869 µs	404	134.532 µs
HLC	393	130.869 µs	613	204.129 µs
HLB	393	130.869 µs	1448	482.184 µs
AOR	721215	240.164595 ms	729632	242.967456 ms
ALO	721215	240.164595 ms	731648	243.638784 ms

5.3.4 Required Processing Time

Regarding the processing time required by these testing experiments, the highest one among those simulations performed through a machine Sun Blade 2000 from Sun Microsystems was around 3 hours. Such processing time correspond to the simulation tasks that execute the post-layout gate-level simulations. It includes also the processing time to convert the generated VCD file to a TCF file. As discussed in section 4.5, this TCF file is required at the power analysis. In contrast, the processing time at the pre-layout gate-level simulations was around 1 hour and 30 minutes.

The time resolutions used at the post-layout and pre-layout gate-level simulations due to the technology-library requirements were 1 ps. On the other hand, at the behavioral simulations, the time resolution was 1 ns. It is the default of the simulator. As the circuit component models do not have delays at these behavioral simulations, a higher value for the time resolution can be used to speed up the simulation processing times. At the behavioral simulations, such processing times were lower than 5 minutes.

5.3.5 Verification of Functional Testing Results

In order to verify whether the benchmark executions attain the functional goals specified by their instructions, the output values generated by such functional testing were monitored. If any instruction was not working correctly, these resulting output values would be corrupted.

These output values are results due to the functional characteristics programmed in the benchmarks. In this way, they were previously specified to be attained by means of instructions. Therefore, they were known before the benchmark executions. Thus, golden values (i.e., values that are considered the correct output values) could be defined for each benchmark. An illustrative example can be given by the HNO benchmark that was designed to obtain a golden value 0x04d2, consequently its instructions must be processed to result such output value 0x04d2. As all benchmark instructions are executed in sequence, whether an instruction fails, this golden value will not be obtained or even the execution will be interrupted.

The output values monitored in benchmark executions (i.e., the benchmark output results) were stored by the testbench in text files. In this way, they could be compared with the defined golden values (i.e., the specified benchmark output results) in order to detect eventual differences and thus eventual design errors. Such verification process of design simulation results can be illustrated by Figure 5.3. It was performed until the design simulation results matched the golden results.

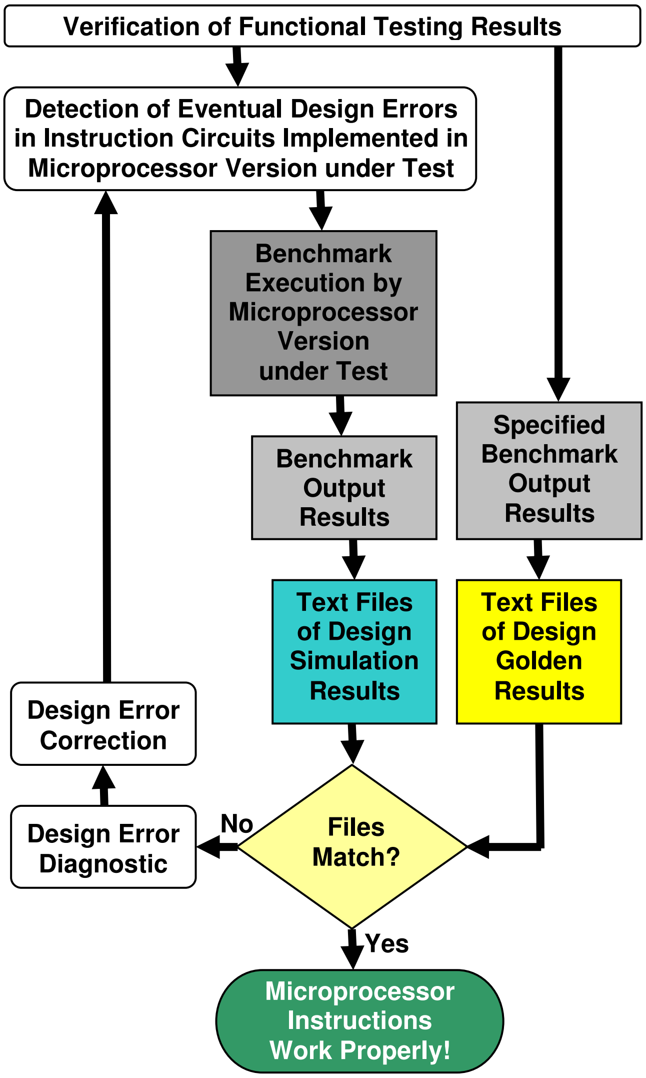

Verification of Functional Testing Results
Detection of Eventual Design Errors in Instruction Circuits Implemented in Microprocessor Version under Test
Benchmark Execution by Microprocessor Version under Test
Benchmark Output Results
Specified Benchmark Output Results
Text Files of Design Simulation Results
Text Files of Design Golden Results
Design Error Correction
Design Error Diagnostic
No
Files Match?
Yes
Microprocessor Instructions Work Properly!

Figure 5.3: Verification process of functional testing simulation results

Table 5.3 summarizes all functional testing simulations performed in order to detect eventual design errors in the implemented microprocessor versions.

Table 5.3: All functional testing simulation approaches

Functional Testing Approach	Simulation Stage	Microprocessor Version Under Test	Executed Benchmark
1	Behavioral simulation	Non-Protected	HNO
2			HLC
3			HLB
4			AOR
5			ALO
6		TMR	HNO
7			HLC
8			HLB
9			AOR
10			ALO
11		TMR+TR+CWSP	HNO
12			HLC
13			HLB
14			AOR
15			ALO
16	Pre-layout gate-level simulation	Non-Protected	HNO
17			HLC
18			HLB
19			AOR
20			ALO
21		TMR	HNO
22			HLC
23			HLB
24			AOR
25			ALO
26		TMR+TR+CWSP	HNO
27			HLC
28			HLB
29			AOR
30			ALO
31	Post-layout gate-level simulation	Non-Protected	HNO
32			HLC
33			HLB
34			AOR
35			ALO
36		TMR	HNO
37			HLC
38			HLB
39			AOR
40			ALO
41		TMR+TR+CWSP	HNO
42			HLC
43			HLB
44			AOR
45			ALO

5.4 Fault Injection by Simulation

Fault injection experiments were performed through the post-layout gate-level simulation discussed in section 5.1.1. The goal of this simulation experiment is to verify by testing functionally the fault-tolerance mechanisms implemented in each one of the designed robust microprocessor versions (TMR and TMR+TR+CWSP) detailed in section 4.4. The target physical faults (SETs) were represented by rectangular transient logic pulses injected at certain circuit nodes. In fact, the target faults are direct and indirect SEUs. However, these two types of faults can be induced by SET pulses, as discussed in chapter 2.

As the target faults have a timing nature, these fault injection experiments require a type of simulation like the post-layout gate-level simulation that considers circuit models based on delays of logic gates and interconnects. The resulting circuit models of the three microprocessor versions are represented by VHDL netlists. On the other hand, the delays of logic gates and interconnects are defined by SDF files.

Furthermore, the target faults by nature occur during the use of the circuit. In this way, they need be injected during the simulation of the circuit. By using the same framework of simulation utilized by the functional testing detailed in section 5.3, SET pulses were injected during the execution of a benchmark.

In order to inject the SET pulses on the target circuit models, the VHDL netlists of the microprocessor versions were internally modified by insertion of specific VHDL components able to inject such faults. Indeed, a unique reusable parameterized VHDL component labeled as SET Injector was developed to inject SET pulses of adjustable width on any circuit node or signal and at any instant during each clock cycle of any benchmark. Such injected SET characteristics can be adjusted by parameters (VHDL generic) in the SET Injector component. In this way, this component can inject SETs of distinct characteristics on different parts where it is instantiated in the VHDL netlists.

5.4.1 Instants of the SET Injection

By using the SET Injector, one SET pulse was programmed to occur in each clock cycle of a benchmark. Since in this fault injection simulation, the goal was detecting eventual design errors in the circuits of the fault-tolerance mechanisms (on-line self-checkers) implemented in the target microprocessor. In other words, it was to verify whether these self-checkers mitigate appropriately SETs and SEUs in accord to the constraints of their features. Only those SETs, which respect the maximum width tolerated by the robust microprocessor and occur at instants at which they potentially would provoke SEs, need be injected.

Note that, as all injected SETs were fitted in these conditions, the self-checkers were able to cover all injected faults. In this way, these self-checkers were functionally tested through a fault injection simulation, but a measure of quality by the fault coverage of them was not able to be evaluated. To obtain this fault coverage, a more complex fault simulation based on probabilistic methods is required, as section 5.4.5 emphasizes. On the other hand, such self-checkers are typically considered efficient, as discussed in section 3.1 and 4.6.

5.4.2 Widths of Injected SETs

The SET Injector models SETs like rectangular pulses. Regarding the widths of injected SETs, some issues were considered in accord to section 4.6.1:

- The width of SETs is typically around hundreds of picoseconds;

- The TMR+TR+CWSP microprocessor version mitigates SETs of widths up to around 1 ns as a result of the timing constraints implemented in the Delay blocks from Figure 4.17;
- Moreover, there is the electrical masking effect of SETs by logic gates. In the target technology (AUSTRIAMICROSYSTEM, 2003), the delays of basic standard logic gates are typically around 10 ps and 2 ns.

Thus, in this simulation experiment, pulses of 100 ps and 1 ns were injected to evaluate extremities of widths tolerated by the robust microprocessor.

5.4.3 Target Circuit Nodes for the SET Injection

Ten SET injection simulation approaches were performed by considering different circuit nodes. In each approach, the HNO benchmark detailed in 5.3, which uses all 18 existing microprocessor registers, was executed under SET injections in order to functionally test all implemented fault-tolerance mechanisms.

- **SET Injection Approach 1**: it was performed on the Non-Protected microprocessor version just to verify whether the shapes of the injected SETs effectively provoke SEs. SET pulses were injected at the microprocessor register inputs (node A in Figure 5.4). Since all 18 microprocessor registers totalize 187 flip-flops, 187 SET pulses of 1 ns were injected in each clock cycle of the benchmark. Figure 5.5 illustrates a SET injected on this Non-Protected version in order to cause a SE on a bit of a microprocessor register. In this way, the SET meets the set-up and hold time requirements to avoid the latching-window masking discussed in section 2.1.2.3. Note that this fault injected on node A effectively results in a SE at the register output. The value 0 should be stored and not the value 1 from the faulty output of the combinational block;
- **SET Injection Approach 2**: it was the same that SET Injection Approach 1 except SET pulses of 100 ps were injected;

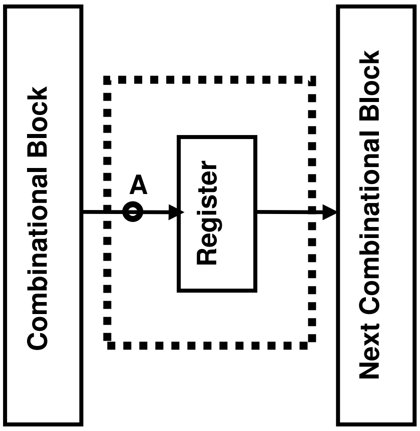


Figure 5.4: Target circuit nodes in the Non-Protected version

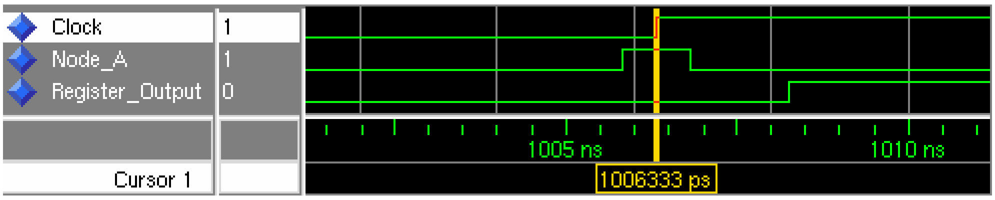


Figure 5.5: SET injected on the Non-Protected microprocessor version

- **SET Injection Approach 3**: it was performed on the TMR microprocessor version. SET pulses were injected only at one input of a triplicated flip-flop (node B in Figure 5.6) in such way that they were stored by one flip-flop. It was to emulate the occurrence of a direct SEU on only one flip-flop. Thus, the voter block functionality could be tested. In the same way that SET Injection Approach 1, 187 SET pulses of 1 ns were injected in each clock cycle of the benchmark in order to test all 187 voters. Figure 5.7 illustrates a SET injected on this TMR microprocessor version that provokes a direct SEU on a flip-flop. By reason of the TMR protection implemented in this circuit, the faulty output of this triplicated flip-flop (value 1 in register 1) is not propagated to the voter output (value 0). Thus, a SE does not occur due to this injected fault;

- **SET Injection Approach 4**: it was the same that SET Injection Approach 3, but the SET pulse was injected only on the node C from Figure 5.6;

- **SET Injection Approach 5**: it was also the same that SET Injection Approach 3, but the SET pulse was injected only on the node D from Figure 5.6;

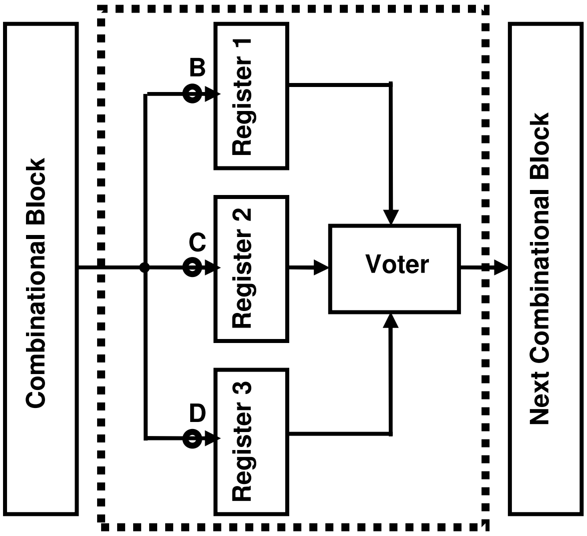
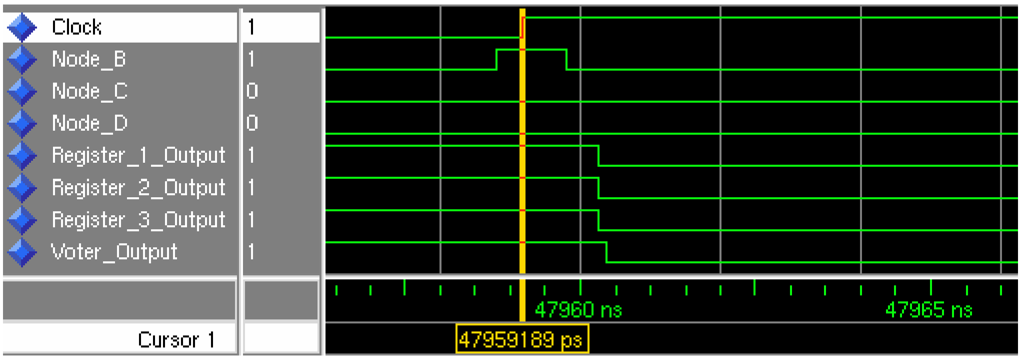


Figure 5.6: Target circuit nodes in the TMR version



Clock 1
Node_B 1
Node_C 0
Node_D 0
Register_1_Output 1
Register_2_Output 1
Register_3_Output 1
Voter_Output 1
Cursor 1 47959189 ps

Figure 5.7: SET injected on the TMR microprocessor version

- **SET Injection Approaches 6, 7 and 8**: these approaches were respectively the same that SET Injection Approaches 3, 4 and 5, but they were performed on the TMR+TR+CWSP microprocessor version and in accord to Figure 5.8;

- **SET Injection Approach 9**: it was performed on the TMR+TR+CWSP microprocessor version. SET pulses were injected at the outputs of the combinational blocks (node A in Figure 5.8). It emulated SETs that potentially would provoke indirect SEUs on the flip-flops. Thus, the TR+CWSP mitigation scheme could be tested. In the same way that SET Injection Approach 1, 187 SET pulses of 1 ns were injected in each clock cycle of the benchmark. Figure 5.9 illustrates a SET injected on the TMR+TR+CWSP microprocessor version that potentially would cause an indirect SEU on a bit of a microprocessor register. Observe that, as there is the TR+CWSP scheme implemented in this circuit, the faulty output of the combinational block (transient value 1 on node A) is not propagated to the three inputs of the triplicated register (value 0 on nodes B, C and D). Consequently, the three outputs of this triplicated register and the voter output are not reached by the fault. Therefore, a SE does not occur due to this injected fault;

- **SET Injection Approach 10**: it was the same that SET Injection Approach 9 except SET pulses of 100 ps were injected.

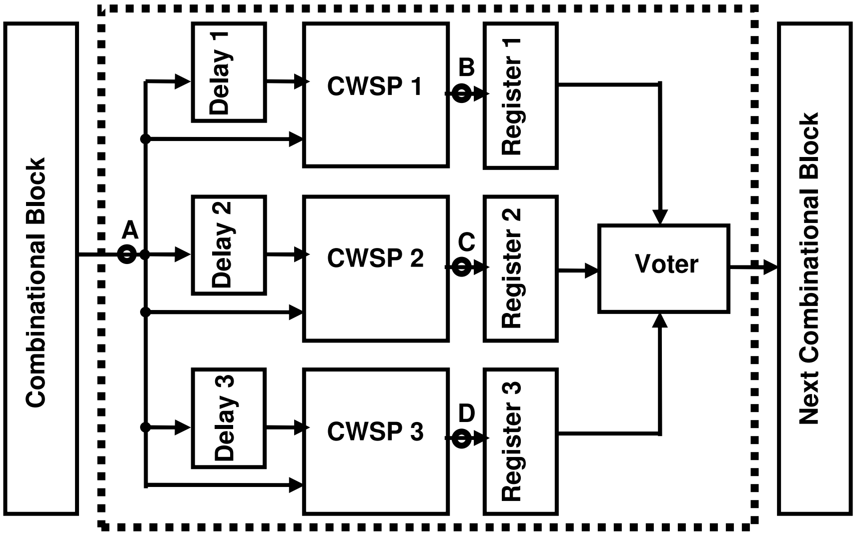


Figure 5.8: Target circuit nodes in the TMR+TR+CWSP version

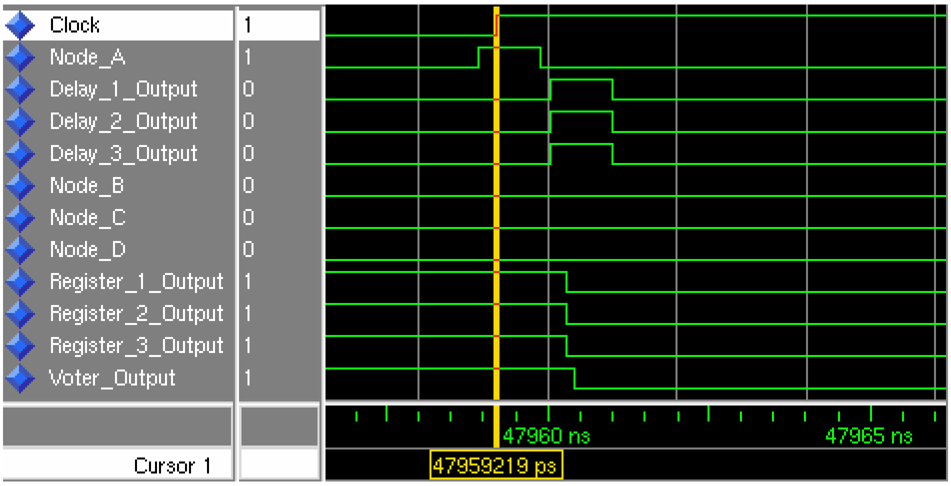


Figure 5.9: SET injected on the TMR+TR+CWSP microprocessor version

5.4.4 Verification of Fault Injection Results

In order to verify whether the implemented self-checkers work properly at benchmark executions under faults, the fault injection simulations were performed by using the same verification process detailed in section 5.3.5. However, just the HNO benchmark was executed through the post-layout gate-level simulation discussed in 5.1.1.

In addition, the values stored in all 18 microprocessor registers and all outputs of the voter blocks were also monitored in each clock cycle of the benchmark. By using a fault-free functional testing simulation of the Non-Protected microprocessor (i.e., a simulation without SET injection), the golden values of the microprocessor registers in each clock cycle were stored by the testbench in text files. In this way, the monitored values in the microprocessor versions under SETs could be compared with ideal values from a correct benchmark execution. Such verification process of design simulation results can be illustrated by Figure 5.10.

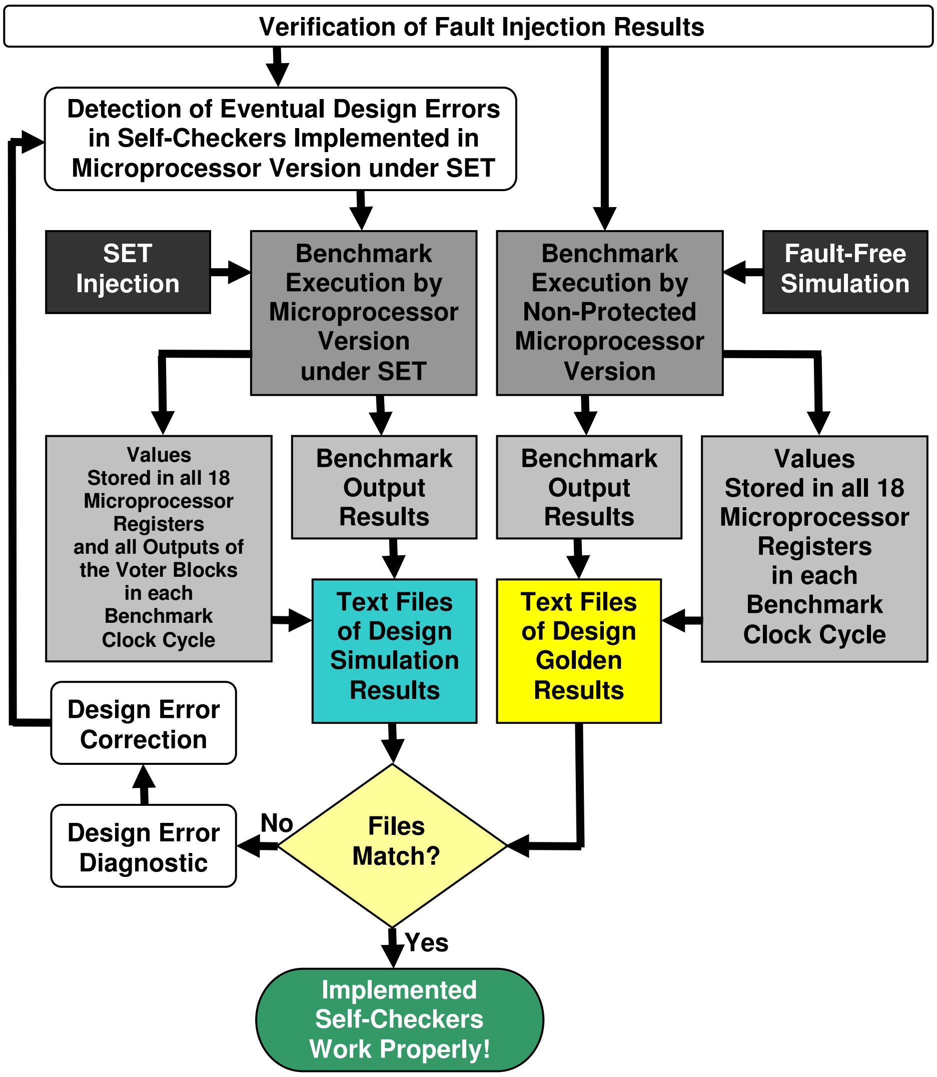

Verification of Fault Injection Results
Detection of Eventual Design Errors in Self-Checkers Implemented in Microprocessor Version under SET
SET Injection
Benchmark Execution by Microprocessor Version under SET
Benchmark Execution by Non-Protected Microprocessor Version
Fault-Free Simulation
Values Stored in all 18 Microprocessor Registers and all Outputs of the Voter Blocks in each Benchmark Clock Cycle
Benchmark Output Results
Benchmark Output Results
Values Stored in all 18 Microprocessor Registers in each Benchmark Clock Cycle
Text Files of Design Simulation Results
Text Files of Design Golden Results
Design Error Correction
Design Error Diagnostic
No
Files Match?
Yes
Implemented Self-Checkers Work Properly!

Figure 5.10: Verification process of fault injection simulation results

In each SET Injection Approach mentioned in section 5.4.3 was injected 187 SET pulses by clock cycle. Since a HNO benchmark execution requires 404 clock cycles (as detailed in Table 5.2), 75548 faults were injected in each one of the ten approaches. In section 5.4.1 was discussed that all these injected faults were covered by the implemented self-checkers. Thus, whether there were not design errors in these self-checkers, all injected faults would be mitigated and therefore the monitored values would match the golden values.

Table 5.4 summarizes all fault injection simulations performed in order to detect eventual design errors. The target circuit-node labels are in accord to Figure 5.4, Figure 5.6 and Figure 5.8.

Table 5.4: All fault injection simulation approaches

SET Injection Approach	Simulation Stage	Microprocessor Version Under Test	Executed Benchmark	Injected SET Width	Target Circuit Node
Fault-Free	Post-layout gate-level simulation	Non-Protected	HNO	-	-
1			HNO	1 ns	A
2			HNO	100 ps	A
3		TMR	HNO	1 ns	B
4			HNO	1 ns	C
5			HNO	1 ns	D
6		TMR+TR+CWSP	HNO	1 ns	B
7			HNO	1 ns	C
8			HNO	1 ns	D
9			HNO	1 ns	A
10			HNO	100 ps	A

5.4.5 Some Remarks about Fault Coverage of On-Line Self-Checkers

Fault simulation is also widely used for evaluation of the test by checking the fault coverage or the percentage of faults detected by a set of input stimuli (LUBASZEWSKI; HUERTAS, 2004). It can be used to measure qualitatively the effectiveness of tests such as the on-line detectors or self-checkers.

The fault simulation developed in this work is not able to obtain such fault coverage, as explained in section 5.4.1. Even so the fault-tolerance mechanisms implemented in the target microprocessor through the TMR and TR+CWSP on-line testing schemes are typically considered efficient, a more accurate estimation of efficiency would be useful. On the other hand, it would incorporate an inherent complexity in the fault simulation, since the target faults by nature have characteristics quite peculiar. Note that faults like the SETs can have different transient shapes. Furthermore, they can occur on any circuit part and at any instant during the use of the circuit. Even though such physical faults can be represented through logical faults based on fault models as that discussed in section 2.1.2, the simulation process would be complex too. See that the logical faults would be modeled like rectangular transient pulses. Therefore, the injected faults would be not just logical levels, they could have different widths, occur on any circuit node and at any instant.

Due to these different possibilities of SET behaviors on the circuits, methods more advanced based on non-deterministic or probabilistic fault simulation are required in order to evaluate the fault coverage of a robust circuit. Evaluation resources like (MASSENGILL et al, 2000; LIMA et al, 2001-a; ALEXANDRESCU; ANGHEL; NICOLAIDIS, 2002; NEVES et al, 2006-a, 2006-b) might offer some answers concerning the effectiveness of mitigation techniques against indirect and direct SEUs on the circuits. However, these works and other many ideas about this issue even lack advances to become in consolidated and practical analysis tools for large circuits. This issue could be progress faster whether commercial tools become further easily available, even so they need improvements. A usual better means of evaluation can be performed through radiation ground test experiments on prototypes of the target circuits.

6 DESIGN RESULTS OF A ROBUST MICROPROCESSOR

A preliminary assessment of the designed circuit characteristics is mandatory ahead of the IC fabrication. Furthermore, the design of the robust architectures always entails extra costs that need be considered. Accurate estimated design results of the IC features can be attained by using advanced CAD resources.

In this work, design results especially in area, performance and power were generated in accord to the IC design flow detailed in chapter 4. It is based on the standard cells from the AMS 0.35 µm CMOS technology (AUSTRIAMICROSYSTEM, 2003). In accord to section 4.5, the same initial constraints were adjusted for the three designed microprocessor versions.

Such design results are shown in the following sections. The costs of the robustness added in the **Non-Protected** microprocessor version are evaluated through the **TMR** and **TMR+TR+CWSP** microprocessor versions. In fact, as the TMR version mitigates only direct SEUs and TMR+TR+CWSP version mitigates direct and indirect SEUs, the overheads to protect the target CPU against such faults can be analyzed.

6.1 Area Analysis

Regarding IC design results in area, there are many issues which can be analyzed such as the final floorplan characteristics after the tool decisions; the number of standard logic cells, filler cells, corner cells, pad cells (I/O, power and ground); the clock-tree elements; the lengths of the routed wires; and obviously the area dimensions in each one of these former issues. In the following sections, results about these design issues for the three microprocessor versions are presented and compared.

6.1.1 Floorplan Characteristics

As detailed in section 4.5, initial floorplan parameters are set as constraints in order to the floorplanner tool achieves them. However, the final achieved floorplan parameters may not be exactly the same of those initially set. It is because the floorplanning task depends also on some designed circuit characteristics like the total number of required standard cells. This and other characteristics are not known when the initial floorplan parameters are set. Thus, the floorplanning algorithms can meet values that differ from those initial decided parameters, but they approach to them.

Table 6.1 presents the floorplan characteristics achieved for the three designed microprocessor versions. As defined in section 4.5, the initial core row utilizations were set to reach final core utilizations around 70 %. The initial aspect ratio was set to attain a square IC area, i.e., the value 1. Furthermore, initially, the left, right, top and bottom

I/O to core distances (x0, x1, y0 and y1 in Figure 6.1) were all equally set to 746.200 µm in order to define the optimized area required for the routing.

Table 6.1: Floorplan characteristics

	Non-Protected	TMR	TMR+TR+CWSP
Initial Core Row Utilization	75.30 %	73.30 %	64.00 %
Final Core Row Utilization	70.48 %	70.85 %	70.97 %
Number of Core Rows	48	58	69
w Side Length of the Core	635.600 µm	754.600 µm	896.000 µm
h Side Length of the Core	624.000 µm	754.000 µm	897.000 µm
Initial Core Aspect Ratio	1.0000	1.0000	1.0000
Final Core Aspect Ratio	1.0186	1.0008	0.9989
Final Core Area	0.39661440 mm^2	0.56896840 mm^2	0.80371200 mm^2
Left, Right, Top and Bottom I/O to Core Distances	746.200 µm	746.200 µm	746.200 µm
x Side Length of the Chip	2128.000 µm	2247.000 µm	2388.400 µm
y Side Length of the Chip	2116.400 µm	2246.400 µm	2389.400 µm
Final Chip Aspect Ratio	1.0055	1.0003	0.9996
Final Chip Area	4.50369920 mm^2	5.04766080 mm^2	5.70684296 mm^2

Based on these initial constraints, the following final design results were obtained (Table 6.1): the final core utilization; the number of rows; and the final core and chip dimensions in accord to Figure 6.1. These final results were adjusted with the aim of placing and routing successfully all cells of the designs, as explained in section 4.5. The final I/O to core distances (x0, x1, y0 and y1) are the same values initially defined as 746.200 µm. Note that the Non-Protected chip has around 4.504 mm^2 and the TMR+TR+CWSP chip around 5.707 mm^2. An area overhead evaluation among the three microprocessor versions is presented in section 6.1.3.

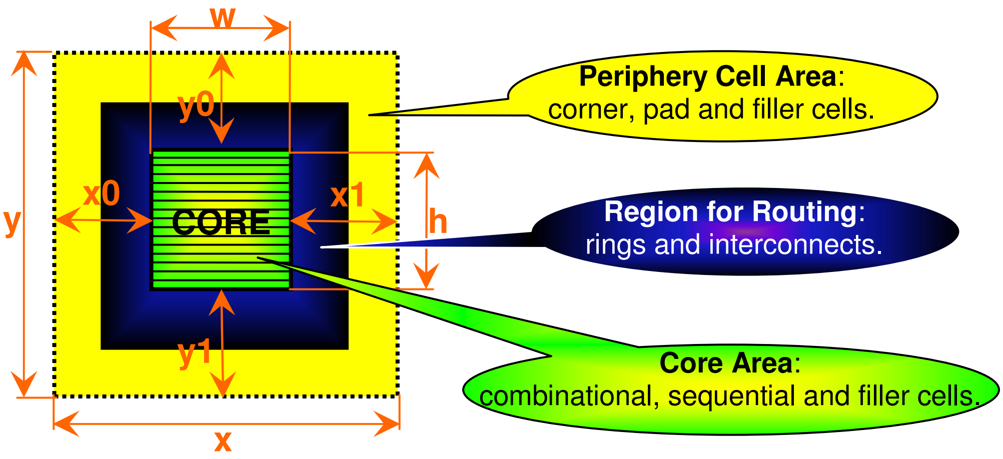


Figure 6.1: Core and chip areas

Microprocessor-version illustrations as an IC preliminary view (before the final layout adjustments, i.e., before step 7 from the design flow in Figure 4.5) and the final IC layout are presented in Figure 6.2, Figure 6.3 and Figure 6.4 by a scale of approximately 28:1. Note that, in the preliminary view, the final routing and pad cells of power and ground were not defined yet. The axes h and y are in relation to Figure 6.1.

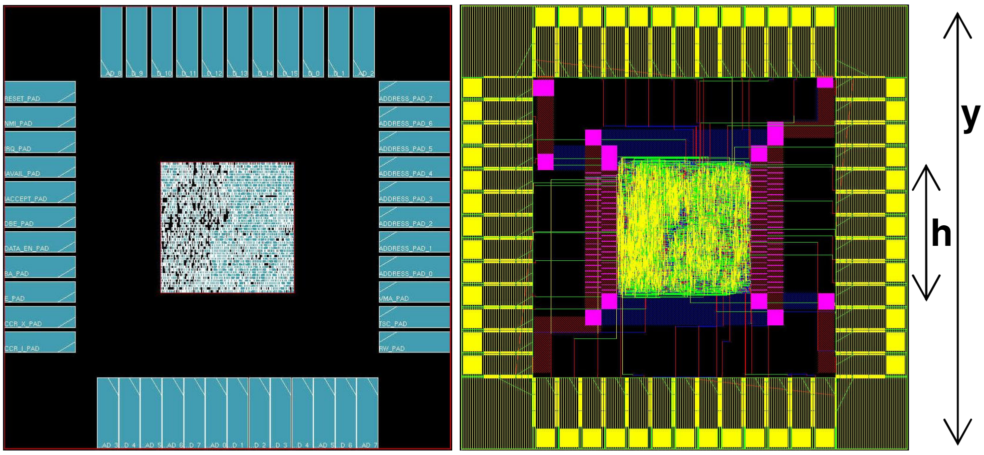


Figure 6.2: Non-Protected IC version: preliminary view (left) and final layout (right)

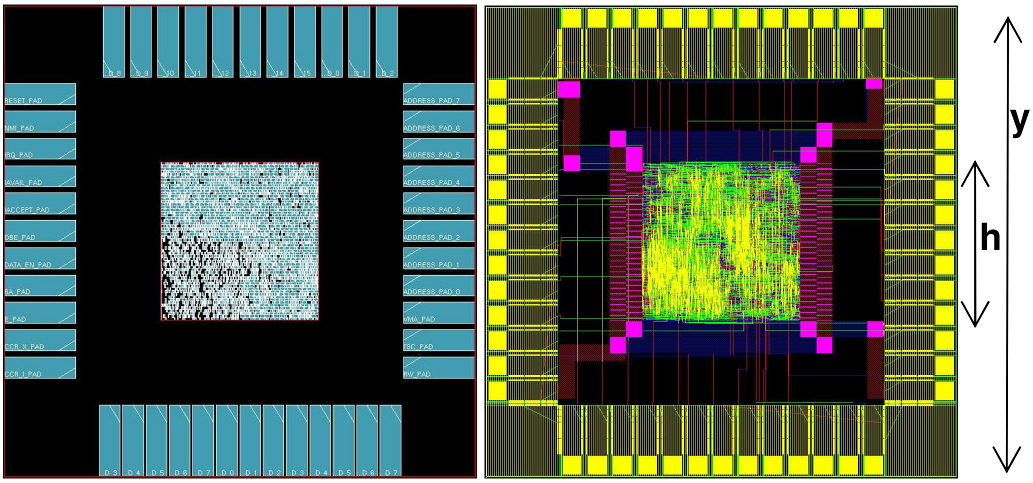


Figure 6.3: TMR IC version: preliminary view (left) and final layout (right)

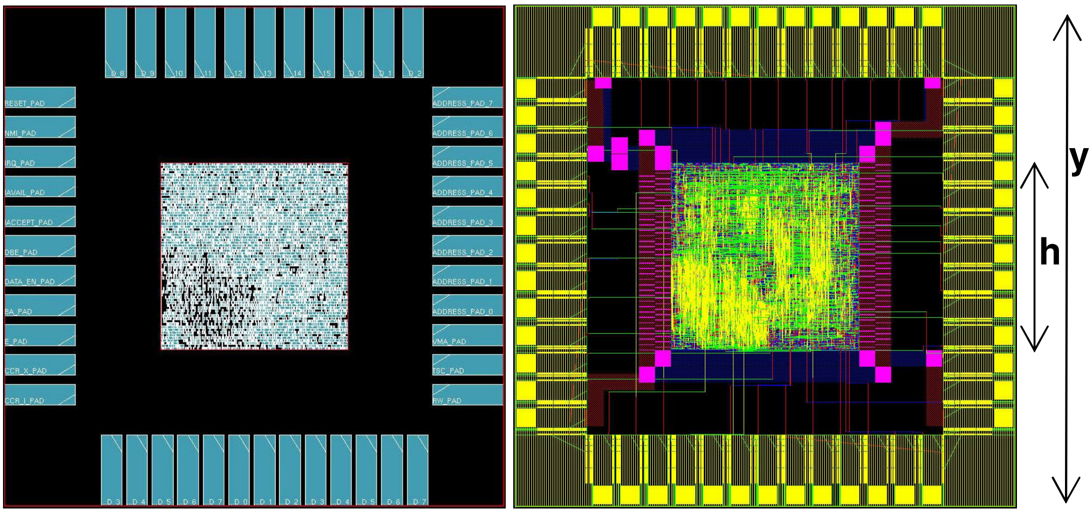


Figure 6.4: TMR+TR+CWSP IC version: preliminary view (left) and final layout (right)

At the final layouts from Figure 6.2, Figure 6.3 and Figure 6.4, the wires routed in four metal layers are represented by the following colors: blue (MET1), red (MET2), green (MET3) and yellow (MET4).

6.1.2 About Standard Logic Cells and Other Types of Cells

In Table 6.2 all types of cells used by the designed circuits are listed. The number of cells (columns "Count") and the area totalized by each type of cell are presented.

The sequential and combinational logic cells indicated in Table 6.2 are classified as parts of the standard logic cells. In the target technology of this work, the sequential logic cells are memory cells like flip-flops and latches. However, in the designed circuits, such cells are only flip-flops. The combinational logic cells are logic gates like NOT, NOR, NAND, XNOR, buffers or circuits of logic gates like half adders, full adders, majority (AB+AC+BC), multiplexers, AND+OR+NOT blocks, OR+AND+NOT blocks.

Table 6.2 shows also the gaps among the standard logic cells which are basically occupied with core-filler cells. The filler cell areas are the spaces for the routing finishing successfully. Information about corner cells is also shown. They are placed at each corner of the chips. As explained in section 4.5, it is to continue the pad ring sequence among I/O, power and ground pad cells that are placed around the core at the yellow area detailed in Figure 6.1. Periphery-filler cells are placed at the vacant spaces among corner and pad cells. Observe that the total number of cells in the Non-Protected chip is 6412 and in the TMR+TR+CWSP chip is 11994. Furthermore, Table 6.2 shows the area between the core and pad cells (row "Region for Routing"). It is the dark blue area detailed in Figure 6.1. This region is used for routing the power and ground rings and the interconnects among the core and the pad cells.

Table 6.2: About standard logic cells and other types of cells (1)

	Non-Protected		TMR		TMR+TR+CWSP	
Cells	Count	Area (μm^2)	Count	Area (μm^2)	Count	Area (μm^2)
Combinational Logic	3211	228501.00	3363	249958.80	5772	417271.40
Sequential Logic	187	51051.00	561	153153.00	561	153153.00
Core Filler	2754	117062.40	3690	165856.60	5329	233287.60
Corner	4	463488.64	4	463488.64	4	463488.64
I/O Pad	46	1565840.00	46	1565840.00	46	1565840.00
Power Pad	3	102120.00	3	102120.00	3	102120.00
Ground Pad	3	102120.00	3	102120.00	3	102120.00
Periphery Filler	204	188717.76	232	358236.96	276	551856.48
Region for Routing	-	1684798.40	-	1886886.80	-	2117705.84
Total Chip	6412	4503699.20	7902	5047660.80	11994	5706842.96

Based on Table 6.2, Table 6.3 details the number of cells (columns "Count") and the area about some sets of chip regions. The row "Logic" represents the sum of combinational and sequential logic cells. As this sum can be obtained at initial design steps, preliminary estimated results in area, as emphasized in sections 4.3.1 and 4.5, can be evaluated based on this information. Results about the core are shown in row "Core". The row "Filler" sums the core and periphery cells. The row "Pad" totalizes the pad cells. The row "Periphery" shows information for the "Periphery Cell Area" from Figure 6.1 that is occupied by corner, pad and filler cells. Moreover, the row "I/O to Core Region" indicates the region between I/Os and core. It is the yellow area plus the dark blue area from Figure 6.1. The existing cells in these yellow and dark blue areas are the periphery cells (corner, pad and filler cells) that do not determine the total area of this "I/O to Core Region". It is because there is a part of this region used for routing where there are not cells. As discussed in the former paragraph, this part is that represented by the "Region for Routing" from Figure 6.1.

Table 6.3: About standard logic cells and other types of cells (2)

	Non-Protected		TMR		TMR+TR+CWSP	
Cells	Count	Area (µm^2)	Count	Area (µm^2)	Count	Area (µm^2)
Logic	3398	279552.00	3924	403111.80	6333	570424.40
Core	6152	396614.40	7614	568968.40	11662	803712.00
Filler	2958	305780.16	3922	524093.56	5605	785144.08
Pad	52	1770080.00	52	1770080.00	52	1770080.00
Periphery	260	2422286.40	288	2591805.60	332	2785425.12
I/O to Core Region	260	4107084.80	288	4478692.40	332	4903130.96

Based on the information from Table 6.2, Figure 6.5 illustrates the percentages of the core utilization by filler, sequential and combinational cells.

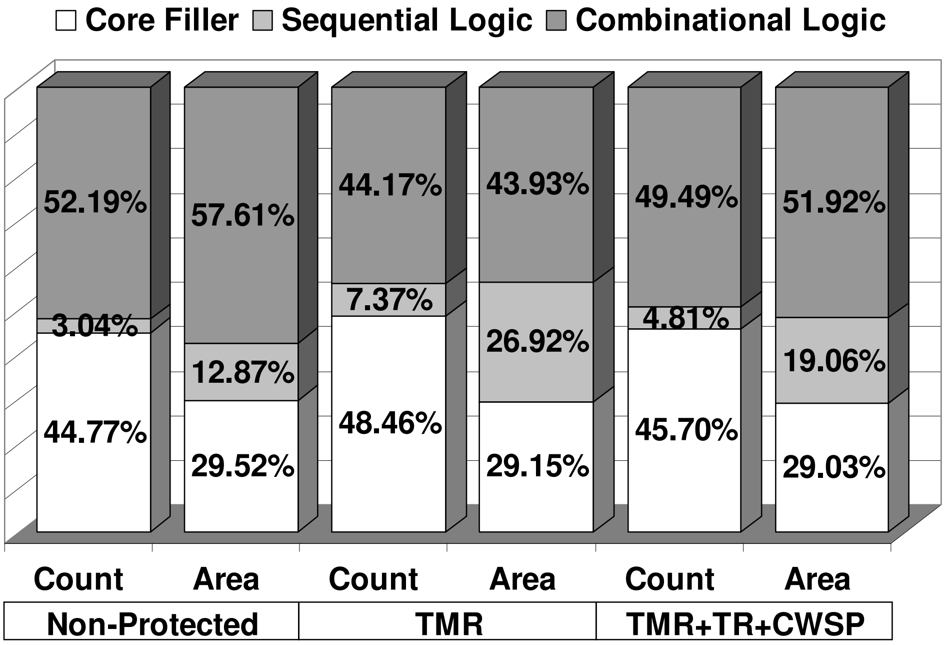


Figure 6.5: Core utilization

Observe in Figure 6.5 that the combinational logic cells prevail on the three microprocessor versions as in number of cells as in area. The count of standard logic cells (sequential and combinational logic) is a little higher than 50 % of the total core cells on the three architectures. The final core utilization indicated in Table 6.1 can be seen through this figure. It is represented by the area of the standard logic cells that use around 71 % of the core area. In consequence, around 29 % of the core area is required for the routing finishing successfully (the core-filler cell areas).

See also that the core area on the Non-Protected version is defined around 0.397 mm^2 in Table 6.1 or Table 6.3. Note in Figure 6.5 that a part of 12.87 % of this area corresponds to 3.04 % of the total core cell count. This part is the sequential logic area composed of storage components (flip-flops) that are inherently susceptible to SEU. It is protected by the TMR on the robust microprocessor versions, i.e., this part is triplicated on these robust versions. Observe that the percentage of SEU susceptible cell area in the TMR+TR+CWSP version (19.06 %) is not so higher than in the Non-Protected version (12.87 %).

Figure 6.6 is also based on Table 6.2 and analyzes the chip utilization by means of the typical circuit regions. Note that the biggest region on the three versions is around 37 % of the chip area. It is that required for routing (dark blue area detailed in Figure 6.1). Another big region is that related to the I/O pad cells that are big cells by nature. See also as the core area is small compared to the chip area. It is around 9 %, 11 % and 14 % of the overall chip area respectively on the three architectures. In the robust microprocessor versions, note that the fault-tolerance mechanisms are implemented in these small chip parts (core areas).

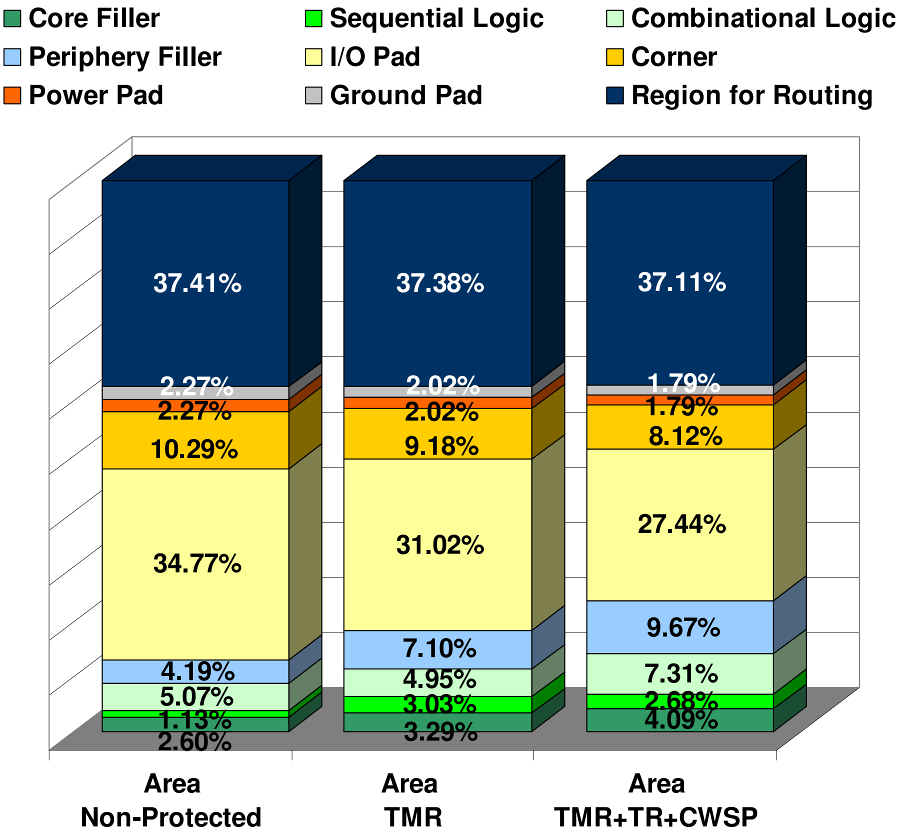


Figure 6.6: Chip area utilization

6.1.3 Costs in Area against Robustness

The TMR and TMR+TR+CWSP microprocessor versions use additional areas to obtain robustness. By using the area results presented in Table 6.2 and Table 6.3, the costs in area of these robust microprocessor versions can be evaluated based on the results of the Non-Protected version. Figure 6.7 and Figure 6.9 illustrate the percent overheads in number of cells to implement the fault-tolerance mechanisms in the Non-Protected version. Figure 6.8 and Figure 6.10 show such overheads in area.

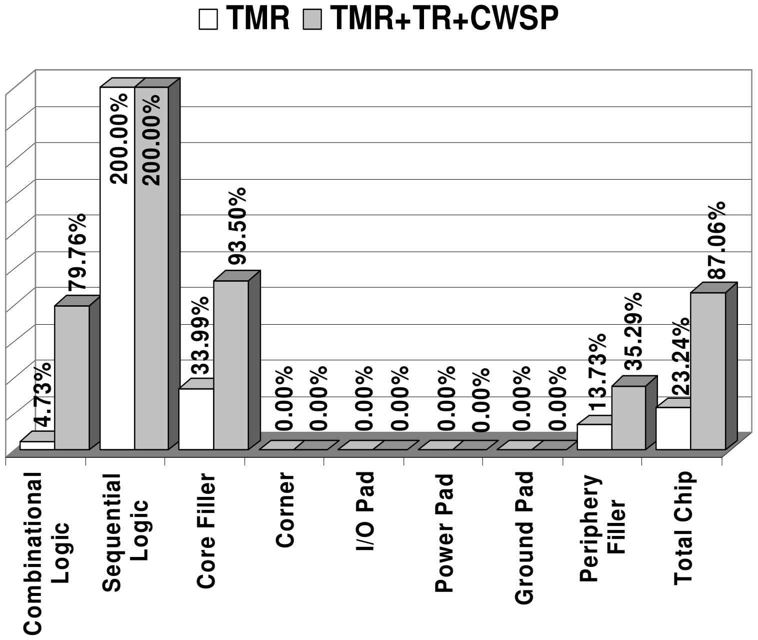


Figure 6.7: Percent increase in number of cells at chip areas (1)

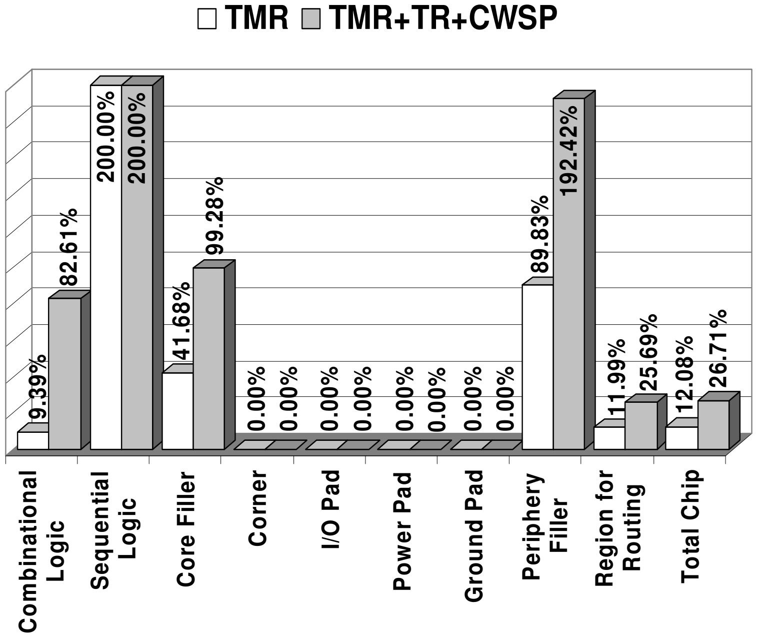


Figure 6.8: Percent increase in chip areas (1)

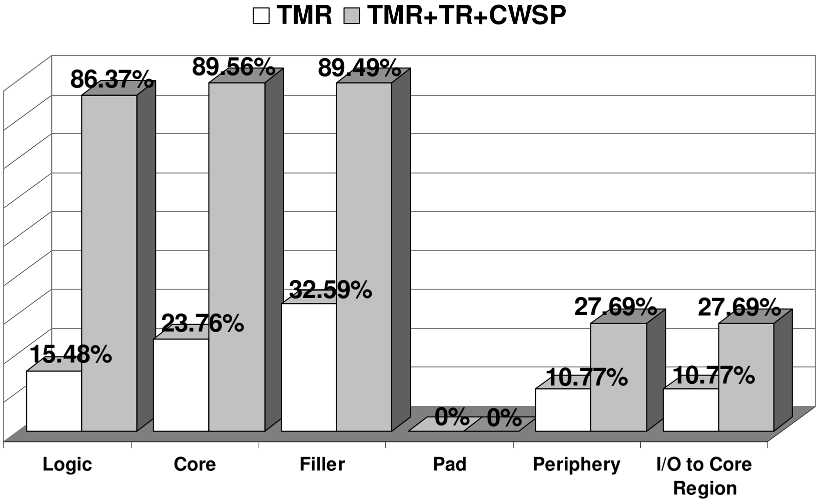


Figure 6.9: Percent increase in number of cells at chip areas (2)

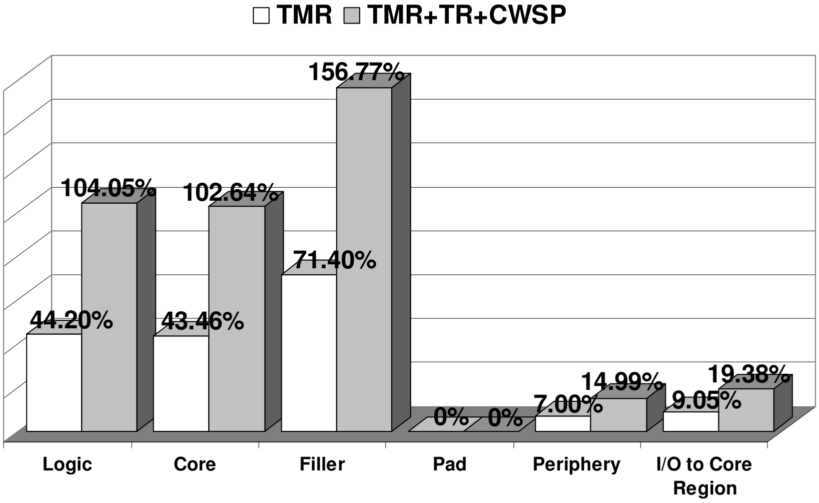


Figure 6.10: Percent increase in chip areas (2)

Observe in Figure 6.7 and Figure 6.8 that the number of sequential logic cells and the area increase by 200 % in the TMR and TMR+TR+CWSP microprocessor versions. It is justified because the TMR technique is applied to protect all sequential logic cells in both versions. Thus, as discussed in section 3.1.1, it inherently triplicates the target components to be protected. In these TMR-based designs detailed in section 4.4.2 and 4.4.3, the 187 sequential logic cells (flip-flops) from the Non-Protected version become 3 × 187 = 561 flip-flops as emphasized in Table 6.2.

On the other hand, note that the number of combinational logic cells increases by 4.73 % and their area by 9.39 % in the TMR version. It is a result of the voter implementations that are circuits purely combinational. Furthermore, it is also due to the additional clock-tree elements (buffers and inverters) that are discussed in section 6.1.4. Since there are the 187 flip-flops in the Non-Protected microprocessor version, there are 187 1-bit voter circuits in the TMR and TMR+TR+CWSP versions. In addition, there are the CWSP and delay blocks in the TMR+TR+CWSP version which are built by using combinational logic cells in order to mitigate SETs of widths up to around 1 ns. Therefore, due to the voter, CWSP and delay circuits for each one of the microprocessor registers and due to the additional clock-tree elements, the number of combinational logic cells rises in 79.76 % and their area in 82.61 % in the TMR+TR+CWSP version.

Figure 6.7 and Figure 6.8 also show increases in the core- and periphery-filler cells. The number of core fillers rises because the final core row utilization is almost the same in the three microprocessor versions. It is around 71 % as detailed in Table 6.1. As the area of combinational and sequential logic cells expands in the TMR and TMR+TR+CWSP versions, the 71 % of row utilization in these versions is bigger in area than the 71 % of row utilization in the Non-Protected version. Thus, as Figure 6.5, Figure 6.7 and Figure 6.8 explain, the 29 % of core-filler area in the robust versions are bigger than the 29 % in the Non-Protected version.

See in Figure 6.7, Figure 6.8, Figure 6.9 and Figure 6.10 that as the I/O pins required by the robust versions are the same used in the Non-Protected version, there are not increases (0 %) in the corner and pad cells. By this reason and since the I/O to core distances (Table 6.1) are maintained in the three microprocessor versions but the core areas increase respectively by 43.46 % and 102.64 % in the TMR and TMR+TR+CWSP version (Figure 6.10). The consequence is that the periphery-filler area (Figure 6.8), the region of routing (Figure 6.8) and the I/O to core region (Figure 6.10) grow too. Note, however, that due to these issues, the total chip area increases are about four times lesser than those in core areas. In the TMR version, it grows by 12.08 % and in the TMR+TR+CWSP version by 26.71 %, as Figure 6.8 illustrates. Figure 6.9 and Figure 6.10 also detail the increases in logic cells (combinational and sequential) and in filler cells (core and periphery).

6.1.4 Clock-Tree Elements

To avoid clock skew, a clock tree was created by an EDA tool to meet the same initial constraints in each microprocessor version (detailed in section 4.5, step 7). The clock-tree elements that build balanced clock networks in the target circuits are presented in Table 6.4. Observe that a total number of 38 combinational logic cells are added in the Non-Protected version to build the clock tree. In the TMR+TR+CWSP version, 107 combinational logic cells are added.

Table 6.4: Clock-tree elements

	Non-Protected		TMR		TMR+TR+CWSP	
Cells	Count	Area (μm^2)	Count	Area (μm^2)	Count	Area (μm^2)
Buffers	25	3494.40	49	7007.00	96	13904.80
Inverters	13	582.40	11	509.60	11	509.60
Total	38	4076.80	60	7516.60	107	14414.40

Figure 6.11 illustrates the contributions of the clock-tree elements in relation to the total number of combinational logic cells, core cells and chip cells (columns “Count”). Furthermore, it shows the contributions in relation to the total combinational, core and chip areas. See that in the Non-Protected version 1.03 % of the core area is due to the clock-tree elements. In the TMR+TR+CWSP version these clock-tree elements represent 0.25 % of the chip area. Observe, however, that the actual areas used by the clock trees in the microprocessor versions are bigger than those shown in Figure 6.11 in

relation to core and chip areas. It is because this figure does not consider the area due to the interconnects among buffers and inverters of the clock tree.

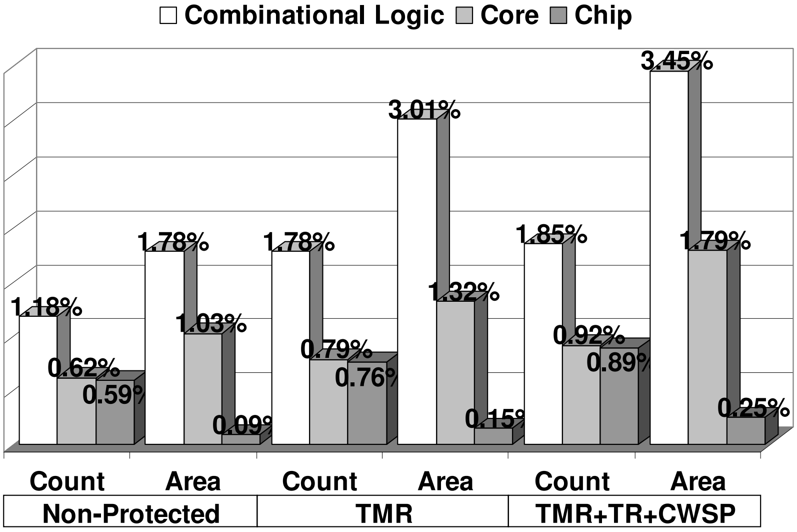


Figure 6.11: Clock-tree element ratios to total combinational, core or chip elements

6.1.4.1 Costs in Clock-Tree Elements against Robustness

Based on the Non-Protected version, the extra costs in combinational logic cells to build the clock tree in the TMR and TMR+TR+CWSP versions are those detailed in Figure 6.12. Note that the number of clock-tree elements (combinational logic cells) increase by 57.89 % in the TMR version. It is a consequence of the flip-flop triplication from the TMR technique. As there are more flip-flops, i.e., more circuit nodes requiring the clock signal, there is a bigger clock tree. In the TMR+TR+CWSP version, the area due to the clock-tree cells rises in 253.57 %. Such growth is higher than those 84.38 % in the TMR version because the sequential and combinational logic area in the TMR+TR+CWSP version is higher than the logic area in the TMR version. Thus, due to the higher circuit complexity and larger core area in the TMR+TR+CWSP version, the distances of its circuit paths are larger. Therefore, more clock-tree elements are required to the clock signal reach successfully all nodes of the circuit.

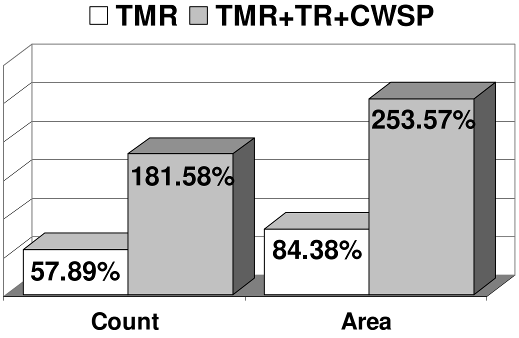


Figure 6.12: Percent increase in clock-tree elements

6.1.5 Routing Issues

The designed circuits can be routed by using wires in four metal layers in accord to the target technology (AUSTRIAMICROSYSTEM, 2003). MET1 and MET3 layers are used preferentially in horizontal routing. Otherwise, MET2 and MET4 layers are preferential in vertical routing. The wires used by the routing can be classified into special and regular wires. Special wires are those for power and ground interconnections. Regular wires are for interconnections among the cells. In Table 6.5, some information about the circuit routing can be seen such as the number of vias (connections between metal layers) and segments in the nets, besides the total length of wires and wire length per layer. See that the total length of all wires generated by the routing for the Non-Protected version is about 45.049 cm and for the TMR+TR+CWSP version about 66.864 cm.

Table 6.5: Routing issues

	Non-Protected			TMR			TMR+TR+CWSP		
Wires	Regular	Special	Total	Regular	Special	Total	Regular	Special	Total
MET1 Wire Length (µm)	7903.22	43600.80	51504.02	9338.90	59002.20	68341.10	13277.77	79386.00	92663.77
MET2 Wire Length (µm)	101967.13	4886.00	106853.13	127935.37	5396.50	133331.87	165546.25	5963.80	171510.05
MET3 Wire Length (µm)	165302.77	0.00	165302.77	201012.10	0.00	201012.10	246689.02	0.00	246689.02
MET4 Wire Length (µm)	126479.83	354.00	126833.83	124689.35	344.00	125033.35	157189.48	588.00	157777.48
Total Wire Length (µm)	401652.95	48840.80	450493.75	462975.72	64742.70	527718.42	582702.52	85937.80	668640.32
Number of Vias	29287	110	29397	32155	130	32285	43603	154	43757
Number of Segments	27034	491	27525	29438	521	29959	41220	558	41778

Based on Table 6.5, Figure 6.13 shows that in the three microprocessor versions the special wires correspond to around 12 % of the total wire length and the regular wires around 88 %. On the other hand, Figure 6.14 emphasizes that MET3 predominates in the regular wire length (around 42 %) and in the total wire length (around 37 %). In the special wire length, MET1 is predominant (around 91 %).

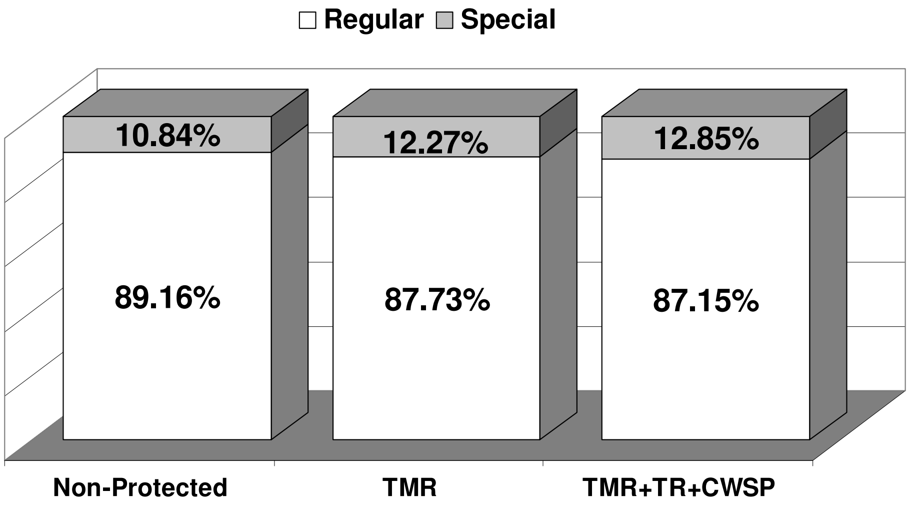


Figure 6.13: Total wire length through the regular and special wires

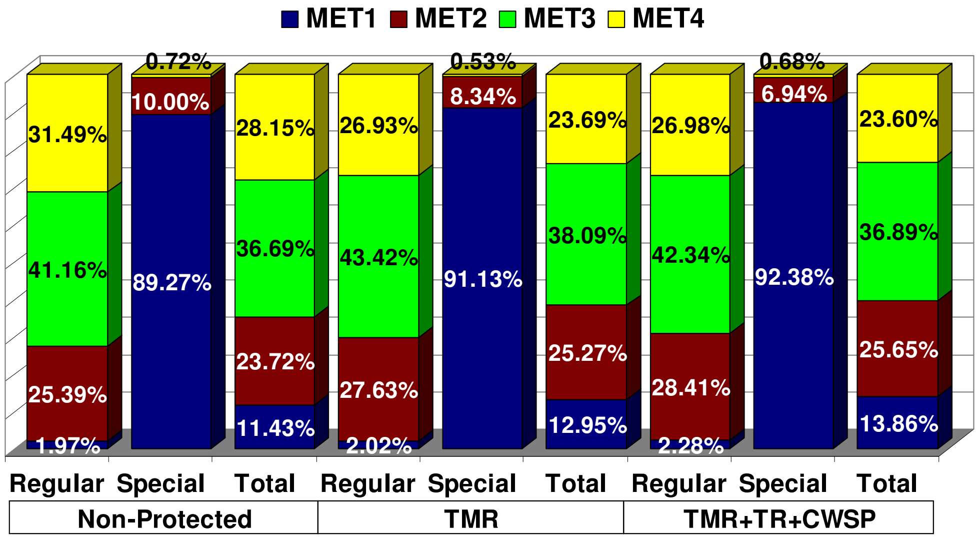


Figure 6.14: Total wire length through the layers

6.1.5.1 Costs in Routing Issues against Robustness

In relation to Non-Protected version, the extra costs in wire length can be seen in Figure 6.15 for the TMR and TMR+TR+CWSP versions. Observe that the total length of all wires generated by the routing is increased by 17.14 % for the TMR version and in 48.42 % for the TMR+TR+CWSP version. Wires in MET1 have the highest increases in length in the TMR+TR+CWSP version. In Figure 6.16, the extra costs in vias and segments are illustrated. Note that in the TMR and TMR+TR+CWSP versions the total number of vias increases respectively by 9.82 % and 48.85 %. The total number of segments of interconnections is grown by 8.84 % in the TMR version and by 51.78 % in the TMR+TR+CWSP version.

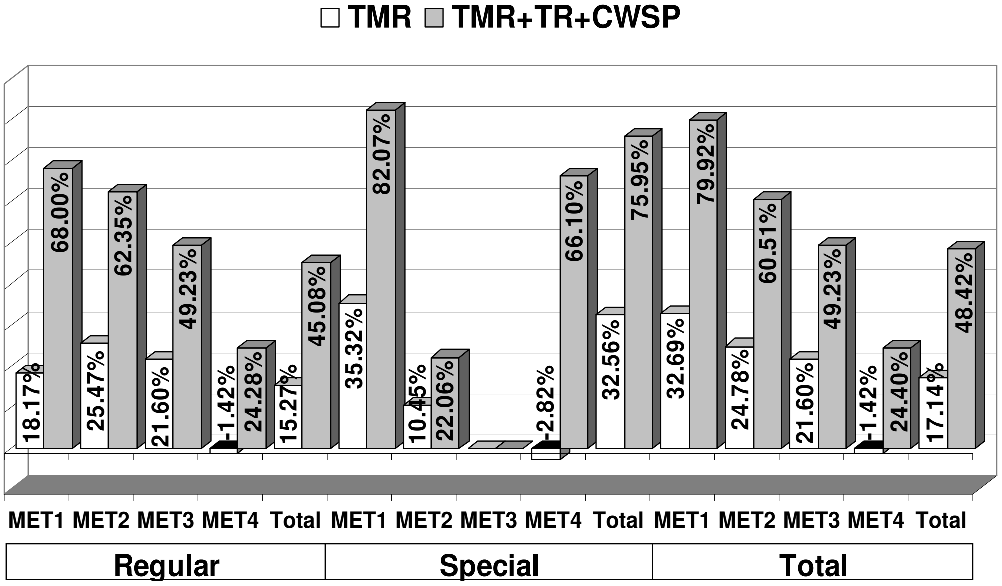


Figure 6.15: Percent increase in wire lengths

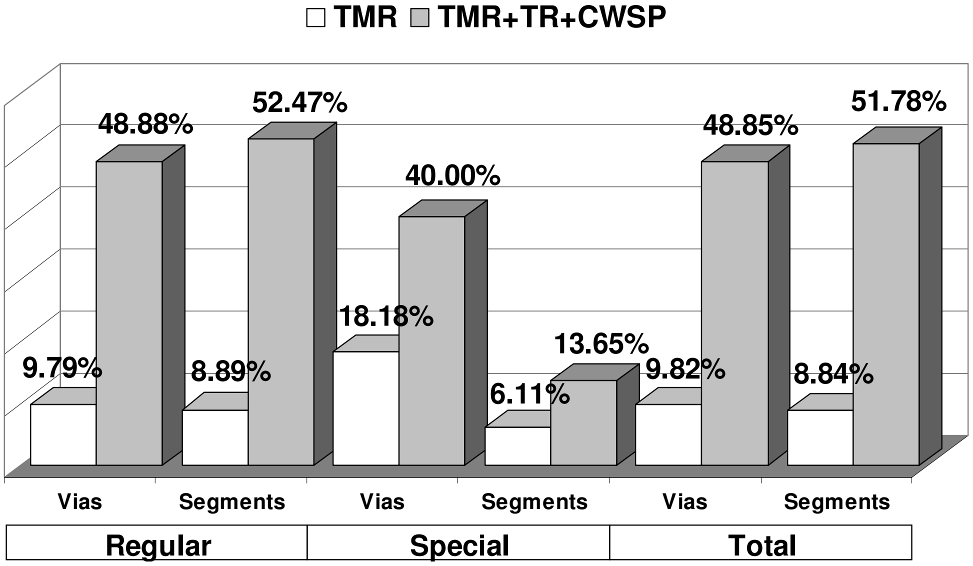


Figure 6.16: Percent increase in vias and segments

6.2 Performance Analysis

Results about performance of the three microprocessor versions are discussed in this section. Performance generally refers to the maximum clock frequency at which the designed circuit can operate. By using a static timing analysis through an EDA tool, the critical path delay of the circuits can be obtained. As detailed in sections 4.5 (step 7) and 5.1.1.1, based on the same initial constraints in each microprocessor version, the static timing analysis was performed at two steps: pre-layout and post-layout.

In Table 6.6, results from the pre-layout and post-layout static timing analysis are presented respectively by the preliminary and final estimations of the worst arrival times in the microprocessor versions. Worst arrival time is defined as the time at which the signal arrives at the other end of the worst circuit path from where this path starts. The maximum frequency is obviously the inverse ratio of this time.

Observe that the maximum frequency achieved by the Non-Protected version is around 14.40 MHz and by the TMR+TR+CWSP version around 12.77 MHz. Otherwise, a CPU from the M68HC11E family designed by Freescale (FREESCALE, 2002) achieves a nominal speed of 3 MHz. However, this comparison might be unfair, since this Freescale's design is implemented through a different fabrication process. It uses a HCMOS technology probably from a generation older than the AMS 0.35 μm CMOS technology (AUSTRIAMICROSYSTEM, 2003) used in the microprocessor versions of this present work. Furthermore, this Freescale's implementation considers a power supply (vdd) of 5.0 V unlike the 3.3 V used in the circuits of this work. The nominal speed of 3 MHz recommend by Freescale respects also the lower speeds typically required by the compatible data and program memories that are habitually used in commercial applications based on this CPU.

Table 6.6: Timing analysis results

	Non-Protected		TMR		TMR+TR+CWSP	
	Worst Arrival Time (ns)	Maximum Frequency (MHz)	Worst Arrival Time (ns)	Maximum Frequency (MHz)	Worst Arrival Time (ns)	Maximum Frequency (MHz)
Preliminary Estimation	36.01	27.77	38.86	25.73	39.74	25.16
Final Estimation	69.45	14.40	75.93	13.17	78.29	12.77

Figure 6.17 shows the contributions from the circuit-extraction information that are added in the preliminary estimations. Note that the worst arrival times at the final estimation are around 95 % higher than the preliminary estimation and the maximum frequencies are around 49 % higher. It occurs because at the preliminary estimation there are not details about the final routing. See that a performance analysis based only on preliminary estimation can be critical due to such differences.

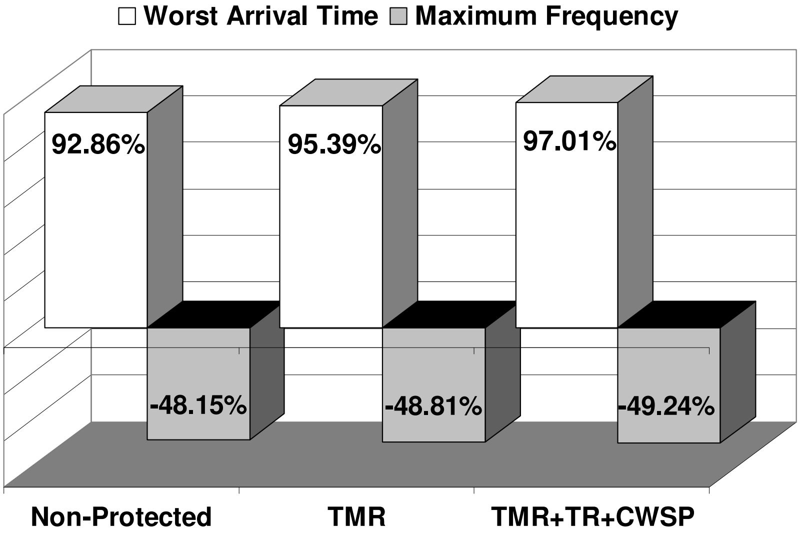


Figure 6.17: Circuit-extraction information contribution at the worst arrival time

6.2.1 Costs in Performance against Robustness

Figure 6.18 illustrates the costs in performance to implement the robustness in the Non-Protected microprocessor version. Observe that in TMR version, the worst arrival time rises in 9.33 % basically due to the voter block discussed in section 3.1.1. In the TMR+TR+CWSP version, the critical path delay is affected in 12.73 %. In accord to sections 3.1.1, 3.2.2 and 4.6.1, it is a result of the defined maximum SET width constraint (around 1 ns), the buffers for the delay blocks and the combinational logic gates for the CWSP elements and voter blocks. Furthermore, these timing degradations in the TMR and TMR+TR+CWSP versions are also due to the extra interconnects (wires) generated by reason of the additional components.

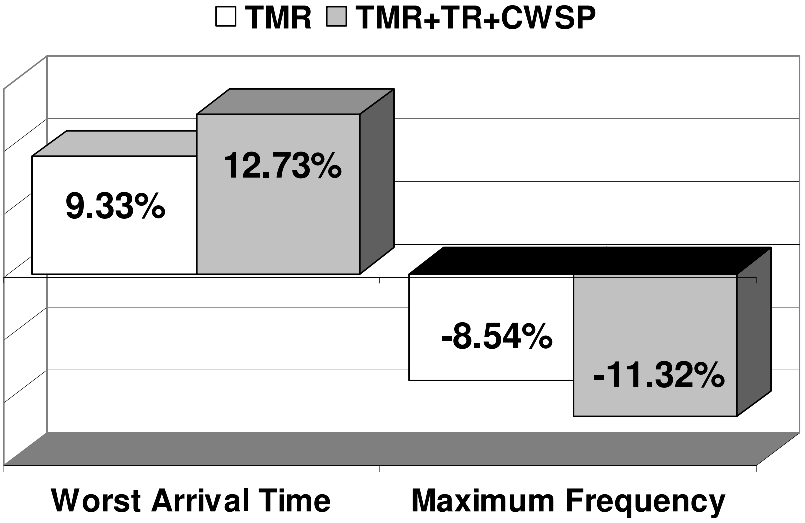


Figure 6.18: Timing degradation

6.3 Power Analysis

Estimations about the power consumption or in other words the power dissipation in the designed circuits are presented in this section. By using the circuit frames generated for the gate-level netlists of the three microprocessor versions, a power analysis through an EDA tool is able to estimate the power consumption in the circuits. The estimations consider the same initial constraints in each microprocessor version to specify input drivers and output loads, as detailed in sections 4.5 (step 7).

In order to calculate the power dissipation in the cells, the power analysis tool takes into account the internal load capacitive power dissipation and the short circuit power dissipation, which are obtained from a look-up table in the technology library. Furthermore, the leakage power dissipation is also considered. It is obtained from a leakage power annotation in the technology library. About the power consumed by nets of wires or interconnects, the tools calculates the power dissipation due to the net capacitance and the capacitances of the pins driven by the nets (CADENCE, 2002).

The leakage power consumption is defined as the static power estimation of the circuit. Regarding the power consumption inside the cells and in the nets, it is defined as the dynamic power estimation. Such estimation inherently requires computing the switching activity of circuit internal nodes unlike the static power estimation. The power analysis tool uses a probabilistic technique to propagate the switching activities from the nodes containing the asserted values. If none of the logic nodes in the circuit contain assertion values, the tool assumes default values at the primary inputs and outputs of sequential elements (CADENCE, 2002).

As mentioned in sections 4.3.1 and 4.5, a preliminary pre-layout power analysis and a more accurate final post-layout power analysis were performed. The preliminary pre-layout evaluation is based on PKS analysis (CADENCE, 2002), a power analysis without stimulating dynamically the circuit inputs is considered. Only default values defined by the power analysis tool itself are fixed at the primary inputs and outputs of the sequential cells. On the other hand, in the final post-layout evaluation of power, the switching activity created by stimuli of benchmarks at circuit inputs is considered. More accurate power results can be obtained because actual toggle count values from the benchmark switching activity are used by the power analysis tool.

Five benchmarks detailed in section 5.3 were considered in this final evaluation. Observe that the dynamic power estimation due to each one of these benchmarks indicates more accurately the dynamic power consumed by a part of the circuit (i.e., by a certain amount of nodes achieved by the stimuli of the benchmark). Habitual software applications normally do not use all parts of the circuit. It can be seen by the toggle coverage shown in Table 5.1 for each one of the benchmarks. These benchmarks use all microprocessor instructions and each one has a toggle coverage around 49 % and 66 %. However, note that the power analysis tool propagates probabilistically the switching activity to other nodes in order to achieve the whole of the circuit. On the other hand, in accord to section 5.3, a more accurate estimation could be obtained by using the five benchmarks in sequence, as a unique program. Thus, practically all circuit nodes would be attained. Nevertheless, the simulation and analysis tools would require memory resources and processing time that would be quite onerous to execute this task. In this way, a probabilistic approach is a sufficient analysis.

Table 6.7 shows the static power estimation through the leakage power consumption, moreover, the dynamic power estimation by means of the power consumption inside the cells and in the nets. The leakage power in the Non-Protected microprocessor version corresponds roughly 0.24 µW and in the TMR+TR+CWSP version approximately 0.50 µW. See that the preliminary pre-layout power analysis (the PKS estimation discussed at the former paragraphs) indicates a total power consumed by the Non-Protected version about 0.301 mW and around 0.506 mW by the TMR+TR+CWSP version. The dynamic power due to the five benchamrks considers the periods of the program main loops detailed in Table 5.2. Observe that the Non-Protected version consumes around 3.4 mW by using each one of the five benchmarks and the TMR+TR+CWSP version around 5.1 mW.

In Figure 6.19, the percent distribution of the power components from Table 6.7 is shown for each benchmark and the PKS estimation. Note that the component related to the power consumption of cells corresponds to around 55 % of the overall consumption and the component due to wires or interconnects among cells approximately 45 %. The component related to the leakage power is a quite small part of the overall power. It is around 0.09 % for the PKS estimation and approximately 0.009 % for the benchmarks. In these results, the dynamic power consumption predominance is justified as a typical characteristic of technologies like that used in the designed circuits (CMOS, channel length of 0.35 µm). The static power component can constitute a significant portion of the total power consumption in more recent technologies based on lesser channel lengths (KIM et al, 2003).

Table 6.7: Power results

	Non-Protected				TMR				TMR+TR+CWSP			
Power (µW)	Static Leakage			Total Power (Dynamic + Static) (mW)	Static Leakage			Total Power (Dynamic + Static) (mW)	Static Leakage			Total Power (Dynamic + Static) (mW)
	0.2440				0.3645				0.5024			
Power (mW)	Dynamic				Dynamic				Dynamic			
	Internal Cells	Nets	Total		Internal Cells	Nets	Total		Internal Cells	Nets	Total	
PKS Estimation	0.1515	0.1494	0.3009	0.3011440	0.2134	0.2009	0.4143	0.4146645	0.2637	0.2420	0.5057	0.5062024
HNO Benchmark	1.7985	1.5307	3.3292	3.3294440	2.3382	1.7762	4.1144	4.1147645	2.8725	2.1673	5.0398	5.0403024
HLC Benchmark	1.8265	1.5661	3.3926	3.3928440	2.3591	1.8062	4.1653	4.1656645	2.8491	2.1664	5.0155	5.0160024
HLB Benchmark	1.8270	1.5672	3.3942	3.3944440	2.3596	1.8073	4.1669	4.1672645	2.8491	2.1673	5.0164	5.0169024
AOR Benchmark	1.9740	1.6294	3.6034	3.6036440	2.5081	1.8846	4.3927	4.3930645	3.0822	2.2893	5.3715	5.3720024
AOL Benchmark	1.9176	1.6003	3.5179	3.5181440	2.4468	1.8524	4.2992	4.2995645	2.9912	2.2336	5.2248	5.2253024

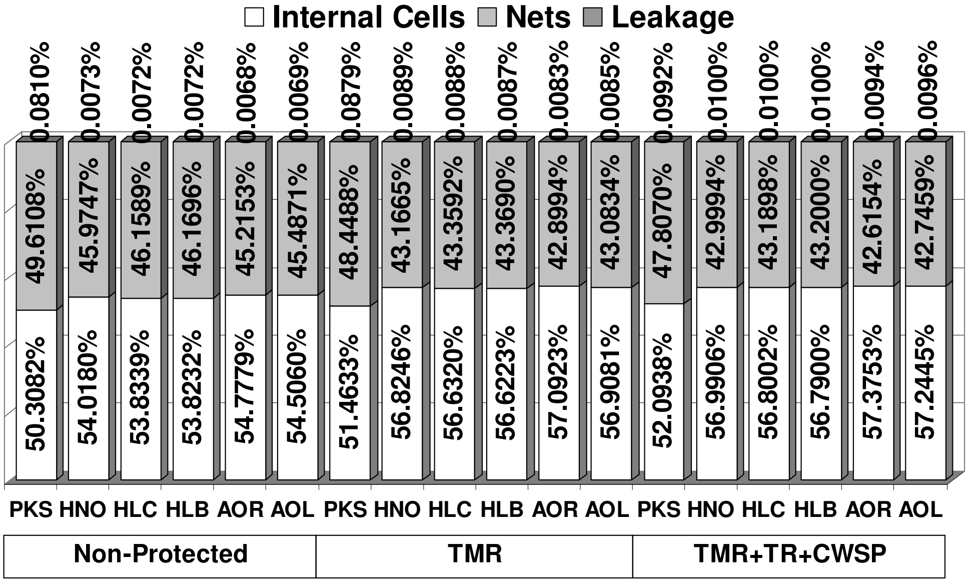


Figure 6.19: Total power through its components

6.3.1 Costs in Power against Robustness

By means of the TMR and TMR+TR+CWSP microprocessor versions, the extra costs in power due to the robustness applied in the Non-Protected microprocessor version can be analyzed based on the results from Table 6.7. In addition, observe that in the Non-Protected version, 88.54 % of the total leakage power and around 33 % of the total power consumption (dynamic power + static power) are due to the core area. Thus, the remaining 11.46 % of the total leakage power and the 67 % of the total power are consumed by the “Region for Routing” and “Periphery Cell Area” from Figure 6.1.

Figure 6.20 illustrates the increases in leakage power dissipation. It increases around 49 % in the TMR version and approximately 106 % in the TMR+TR+CWSP version.

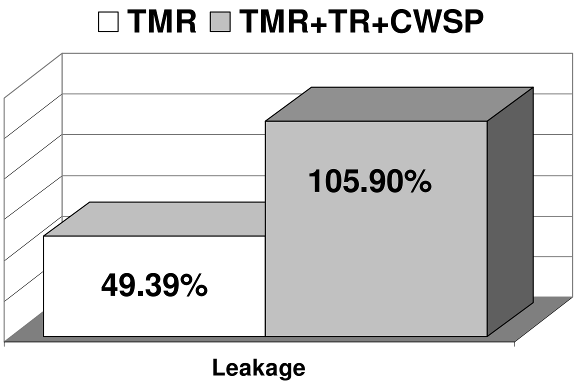


Figure 6.20: Percent increase in static power

On the other hand, Figure 6.21 shows that the total dynamic power consumption in the TMR version elevates around 23 % by using each one of the benchmarks and around 49 % in the TMR+TR+CWSP version. The less accurate PKS estimation indicates increases by roughly 38 % and 68 % respectively. In Figure 6.22 is denoted that such results are practically the same in respect to the increases in total power consumptions. It is explained by the dynamic power consumption predominance seen in Figure 6.19 and emphasized at the beginning of this section 6.3. Note that, as discussed in section 4.4, such increases in the TMR and TMR+TR+CWSP versions are basically due to the triplications of registers, inclusions of voters, additional clock-tree elements and required interconnects. Additionally, in the TMR+TR+CWSP version, the increases are also a result of the insertions of CWSP elements, delay blocks, additional clock-tree elements and their required interconnects.

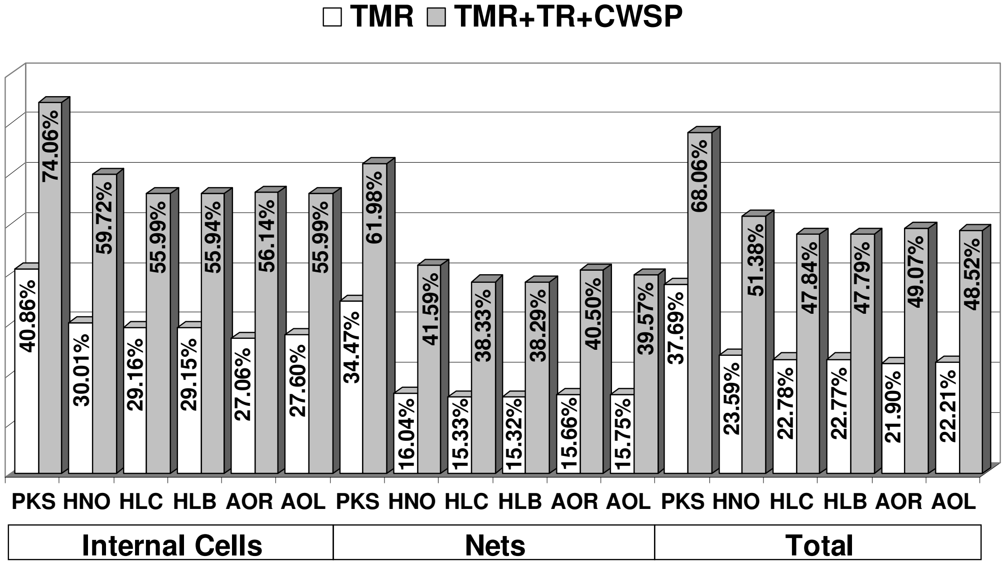


Figure 6.21: Percent increase in dynamic power

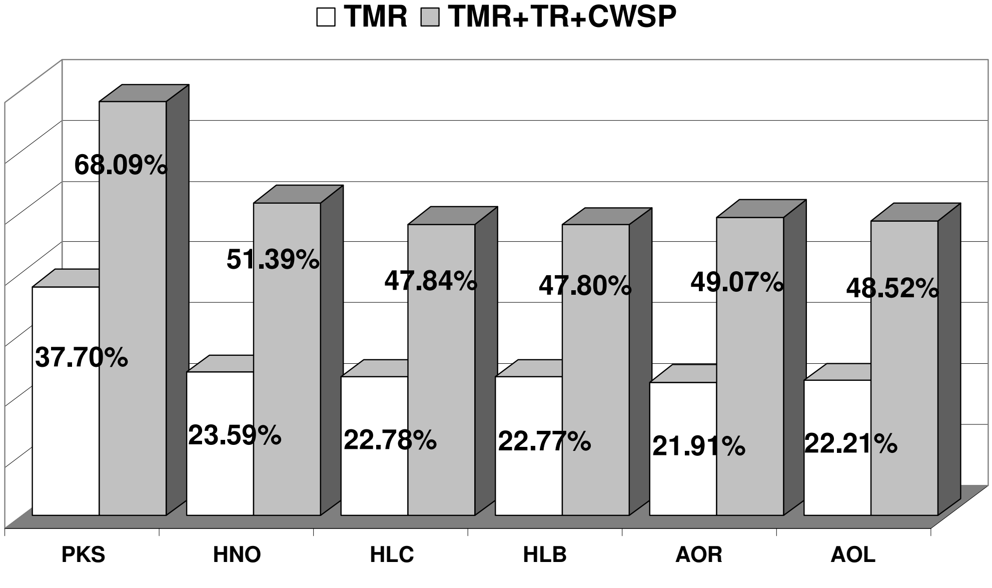


Figure 6.22: Percent increase in total power

7 CONCLUSIONS AND FINAL REMARKS

Inherently the implementation of any fault-tolerance mechanism involves additional overheads that claim a preliminary analysis before the IC manufacture. In the present work, fault-tolerance techniques were explored in such way that the most adequate ones for the target microprocessor were employed. Those techniques that use multiple clock networks and do not preserve the total number of clock cycles were avoided in order to conserve the standard microprocessor characteristics as detailed in section 4.2. The main goal was to evaluate the extra costs in area, performance and power due to such robustness implemented in this target circuit.

The work also explained the basic design steps to develop at the RT level a robust 8-bit microprocessor to SEs. By using a typical IC design flow, the front-end logical design and back-end physical design were developed. The design started from a VHDL description and achieved a GDSII stream file for manufacture. In addition, functional testing and fault injection simulations based on benchmark executions were performed in order to detect eventual design errors. Two robust microprocessor versions based on fault-tolerance techniques were designed. A microprocessor version protected by the TMR technique mitigates only direct SEUs. Another version mitigates direct and indirect SEUs as a result of the TMR and TR+CWSP protection techniques.

Furthermore, the present work showed the viability of the CWSP element implementation by using a design flow based only on standard logic gates of any library. It determines that a robust circuit by means of CWSP elements is also able to be developed at the RT level without requiring specific non-standard gates or even full-custom layout tools for the CWSP element design. In this way, the TMR and TR+CWSP protection schemes could be modeled as a unique reusable VHDL component which allows saving design time and development cost.

Results detailed in chapter 6 and published in (BASTOS; KASTENSMIDT; REIS, 2006-a, 2006-c, 2006-d) show the cost in area, performance and power to make robust the target microprocessor. The main design results are correlated in Figure 7.1 by means of their percent increases.

To protect such target microprocessor only against direct SEUs (TMR version), the core area increases by 43.46 % (9.39 % in combinational cell area and 200 % in sequential cell area). It results in a performance degradation of 9.33 % and a power consumption growth around 23 % for the target benchmarks. The static power grows by 49.39 %. The total length of all wires generated by the routing is increased by 17.14 %. Moreover, the extra cost in area due to the insertions of buffers and inverters to build the clock tree is + 84.38 %.

On the other hand, to protect the target microprocessor against direct and indirect SEUs (TMR+TR+CWSP version that mitigates SETs of widths up to around 1 ns), the core area practically doubles or it increases by 102.64 % (82.61 % in combinational cell area and 200 % in sequential cell area). It results in a performance degradation of 12.73 % and a power consumption growth around 49 % for the target benchmarks. The static power grows by 105.90 %. The total length of all routed wires is increased by 48.42 %. Furthermore, the extra cost in area due to the clock-tree elements is + 253.57 %.

Note also in Figure 7.1 that the increases in the chip areas are around four times lesser than in the core areas (TMR version: 12.08 %; and TMR+TR+CWSP version: 26.71 %), since the required I/O pins and the I/O to core distances are the same in the three microprocessor versions, as explained in section 6.1.3. The total chip area of the TMR+TR+CWSP version is around 5.707 mm^2. Note that such area is equivalent to approximately the following square: ■.

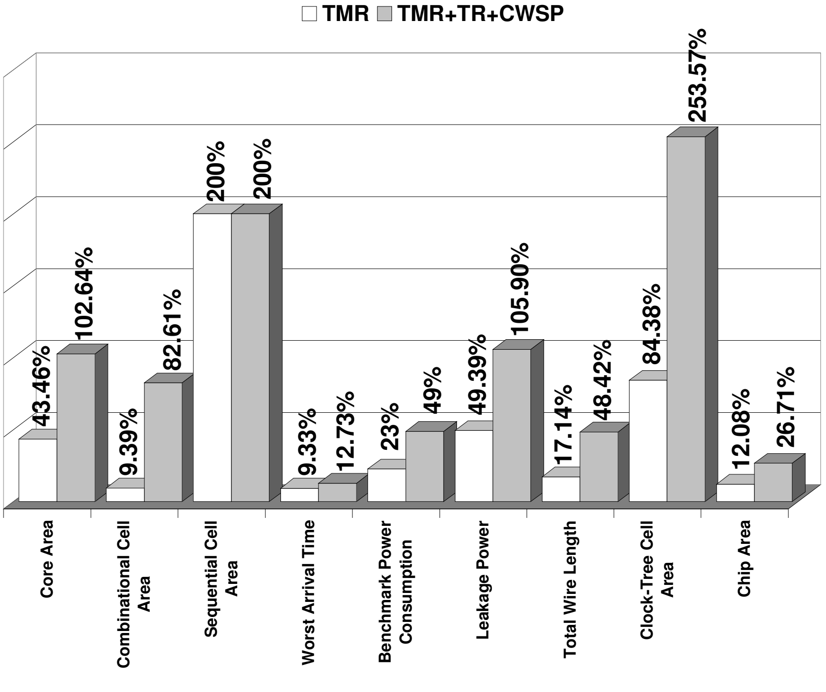


Figure 7.1: Correlation among design results

As illustrated by these results, the implemented fault-tolerance techniques induce considerable overheads in area, performance and power. However, it is the required cost to protect the target microprocessor at the RT level without modifying its standard characteristics. In this way, due to the maintenance of compatibility between the standard non-protected architecture and the robust versions, the reliability and reusability of their existing hardware and software applications are guaranteed. Furthermore, these fault-tolerance techniques are simple and fast to be implemented by using the RT level. All these design issues trend to reduce the time-to-market and development cost. On the other hand, other protection technique solutions might improve these results but they cannot maintain such characteristics of simplicity, compatibility and robustness. Alternative solutions could be aggregating compensation techniques that maintain these characteristics and decrease these costs due to the robustness. An alternative solution to reduce the dynamic power consumption was proposed in (BASTOS; KASTENSMIDT; REIS, 2005-b). By using just some bytes of the data memory and simple adjustments in the software applications, this low-power technique decreases the dynamic power consumption without penalties in area and performance.

There are many future works that can be done based on the present work. The developed front-end logical design could be implemented by using a nanometer technology. In addition, other microprocessor versions able to mitigate smaller and larger SET widths than that maximum SET width allowed in the present design could be also implemented. The design results of these new microprocessor versions could be compared with the present results that are based on a micrometer technology. Thus, the differences of costs due to the robustness in different technologies and SET widths could be evaluated.

To detect eventual design error, usual verification approaches such as timing analysis, functional testing and fault injection by simulation, and DRC were performed in the three designed microprocessor versions. Before the IC fabrication, other traditional checks like LVS and formal verification could be also performed to certify further these designs. Moreover, the suitable test vectors to detect eventual permanent faults based on the stuck-at fault model could be generated by simulation. Such faults could be injected in the designed circuit models, one at each simulation. Thus, by comparing the circuit outputs in simulations under fault with fault-free simulations, the test vectors that detect faults could be found. During the IC manufacture, such test vectors could be used at the circuit inputs by tester equipments.

As emphasized in section 4.6, the redundant parts of a triplicated register being attacked at the same time are susceptible elements of the designed robust microprocessor. Furthermore, the combinational circuit that defines the clock tree and the voter circuits that achieve directly output pads of the chip are also vulnerable. In order to improve the fault coverage, such parts of the robust circuit even susceptible to faults could be protected by using other fault-tolerance techniques, even so nowadays such circuit debilities have typically low probabilities of inducing errors.

Even though the implemented fault-tolerance techniques are typically considered efficient, an evaluation of those mentioned circuit debilities through fault injection experiments able to obtain the fault coverage could be performed. As discussed in section 5.4.5, it would require complex non-deterministic methods due to the peculiar characteristics of SETs. Since the main future goal of this work is manufacturing the three designed microprocessor versions, radiation ground test experiments could be performed on the prototypes of these target circuits to evaluate the effectiveness of the implemented fault-tolerance mechanisms.

REFERENCES

ABRAMOVICI, M.; BREUER, M. A.; FRIEDMAN, A. D. **Digital Systems Testing and Testable Design**. New York: IEEE, 1990.

ALEXANDRESCU, D.; ANGHEL, L.; NICOLAIDIS, M. New Methods for Evaluating the Impact of Single Event Transients in VDSM ICs. In: IEEE INTERNATIONAL SYMPOSIUM ON DEFECT AND FAULT TOLERANCE IN VLSI SYSTEMS WORKSHOP, DFT, 17., 2002. **Proceedings...** [S.l.]: IEEE Computer Society, 2002. p. 99-107.

ANGHEL, L.; NICOLAIDIS, M. Cost Reduction and Evaluation of a Temporary Faults Detecting Technique. In: DESIGN, AUTOMATION AND TEST IN EUROPE CONFERENCE AND EXHIBITION, DATE, 2000, Paris. **Proceedings...** Los Alamitos: IEEE Computer Society, 2000-a. p. 591-598.

ANGHEL, L.; ALEXANDRESCU, D.; NICOLAIDIS, M. Evaluation of a Soft Error Tolerance Technique Based on Time and/or Space Redundancy. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, SBCCI, 13., 2000, Manaus. **Proceedings...** Los Alamitos: IEEE Computer Society, 2000-b. p. 237-242.

AUSTRIAMICROSYSTEM. **0.35µm CMOS Digital Standard Cell Databook**. Austria, January 2003. Available at: <http://asic.austriamicrosystems.com/databooks/index_c35.html>. Visited on March 2006.

BARTH, J. Applying Computer Simulation Tools to Radiation Effects Problems. In: IEEE NUCLEAR SPACE RADIATION EFFECTS CONFERENCE, NSREC, 1997. **Proceedings...** [S.l.]: IEEE Computer Society, 1997. p. 1-83.

BASTOS, R. P.; KASTENSMIDT, F. L.; REIS, R. Designing Low Power Embedded Software for Mass-Produced Microprocessor by Using a Loop Table in On-Chip Memory. In: SOUTH SYMPOSIUM ON MICROELECTRONICS, SIM, 20., May 6-7, 2005, Santa Cruz do Sul, RS, Brazil. **Proceedings...** Porto Alegre, RS, Brazil: Universidade de Santa Cruz do Sul, UNISC, 2005-a. p. 137-140.

BASTOS, R. P.; KASTENSMIDT, F. L.; REIS, R. Designing Low Power Embedded Software for Mass-Produced Microprocessor by Using a Loop Table in On-Chip Memory. In: INTERNATIONAL WORKSHOP ON POWER AND TIMING MODELING, OPTIMIZATION AND SIMULATION, PATMOS, 15., September 20-23, 2005, Leuven, Belgium. **Proceedings...** Berlin, Germany: Springer, 2005-b. p. 59-68. (Lecture Notes in Computer Science, LNCS, v.3728).

BASTOS, R. P.; KASTENSMIDT, F. L.; REIS, R. Design of a Robust 8-Bit Microprocessor to Soft Single Event Effects. In: LATIN AMERICAN TEST WORKSHOP, LATW, 7., March 26-29, 2006, Buenos Aires, Argentina. **Proceedings...** [S.l.]: IEEE Computer Society, 2006-a. p. 137-142.

BASTOS, R. P.; KASTENSMIDT, F. L.; REIS, R. Design of a Robust 8-Bit Microprocessor to Soft Single Event Effects. In: SOUTH SYMPOSIUM ON MICROELECTRONICS, SIM, 21., May 8, 2006, Porto Alegre, RS, Brazil. **Proceedings…** Porto Alegre, RS, Brazil: Universidade de Federal do Rio Grande do Sul, UFRGS, 2006-b. p. 151-155.

BASTOS, R. P.; KASTENSMIDT, F. L.; REIS, R. Design of a Robust 8-Bit Microprocessor to Soft Errors. In: INTERNATIONAL ON-LINE TESTING SYMPOSIUM, IOLTS, 12., July 10-12, 2006, Lake of Como, Italy. **Proceedings...** [S.l.]: IEEE Computer Society, 2006-c. p. 195-196.

BASTOS, R. P.; KASTENSMIDT, F. L.; REIS, R. Design at High Level of a Robust 8-Bit Microprocessor to Soft Errors by Using Only Standard Gates. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, SBCCI, 19., August 28 – September 1, 2006, Ouro Preto, Brazil. **Proceedings...** [S.l.]: IEEE Computer Society, 2006-d. p. 196-201.

BAUMANN, R.; SMITH, E. Neutron-Induced Boron Fission as a Major Source of Soft Errors in Deep Submicron SRAM Devices**.** In: IEEE INTERNATIONAL ELIABILITY PHYSICS SYMPOSIUM, 38., 2000. **Proceedings…** [S.l.]: IEEE Computer Society, 2000.

BAUMANN, R. C. Soft Errors in Advanced Semiconductor Devices—Part I: The Three Radiation Sources. **IEEE Transactions on Device and Materials Reliability**, [S.l.], v.1, n.1, p. 17-22, Mar. 2001.

BORKAR, S. Designing Reliable Systems from Unreliable Components: The Challenges of Transistor Variability and Degradation. **IEEE Micro**, [S.l.], v.25, n.6, p. 10-16, Nov.-Dec. 2005.

BRÜNING, U. **Hardware Design and Simulation**. April 2006, Computer Architecture Group, Department of Computer Engineering, University of Mannheim, Germany. Available at: < http://mufasa.informatik.uni-mannheim.de/pages/lectures/ss06/hwe/script_pdf/vl_hwd_A4.pdf >. Visited on May 2006.

CADENCE DESIGN SYSTEMS, INC. **Tool Manuals**. USA, May 2002.

CALIN, T.; VARGAS, F.L.; NICOLAIDIS, M. Upset-Tolerant CMOS SRAM Using Current Monitoring: Prototype And Test Experiments. In: INTERNATIONAL TEST CONFERENCE, ITC, 1995. **Proceedings...** [S.l.]: IEEE, 1995. p. 45-53.

CALIN, T.; NICOLAIDIS, M.; VELAZCO, R. Upset Hardened Memory Design for Submicron CMOS Technology. **IEEE Transactions on Nuclear Science**, New York, v.43, n.6, p. 2874 -2878, Dec. 1996.

COCHRAN, D. J. et al. Recent Total Ionizing Dose Results and Displacement Damage Results for Candidate Spacecraft Electronics for NASA. In: RADIATION EFFECTS DATA WORKSHOP, 2005. **Proceedings...** [S.l.]: IEEE, 2005. p. 149-155.

CONSTANTINESCU, C. Neutron SER Characterization of Microprocessors. In: INTERNATIONAL CONFERENCE ON DEPENDABLE SYSTEMS AND NETWORKS, DSN, 2005. **Proceedings...** [S.l.]: IEEE Computer Society, 2005. p. 754-759.

COTA, E.; LIMA, F.; REZGUI, S.; CARRO, L.; VELAZCO, R.; LUBASZEWSKI, M.; REIS, R. Synthesis of an 8051-like Micro-Controller Tolerant to Transient Faults. **Journal of Electronic Testing Theory and Applications,** JETTA, MA, USA, v.17, n.2, p. 149-161, 2001.

DAVIS, R. et al. **BWRC IC Design Flow**. January 2000, Berkeley Wireless Research Center, USA. Available at: < http://bwrc.eecs.berkeley.edu/Presentations/Retreats/Winter_Retreat_Jan_2000/Tuesday%20AM/Microsoft%20PowerPoint%20-%20rhett_ICDFtalk_BWRCwinter2000retreat.pdf >. Visited on May 2006.

DODD, P. E. et al. Neutron-Induced Soft Errors, Latchup, and Comparison of SER Test Methods for SRAM Technologies. In: INTERNATIONAL ELECTRON DEVICES MEETING, IEDM, 2002. **Technical Digest...** [S.l.]: IEEE, 2002. p. 333-336.

FREESCALE SEMICONDUCTOR, INC. **M68HC11 Reference Manual**. USA, April 2002. Available at: <http://www.freescale.com>. Visited on July 2005.

FREESCALE SEMICONDUCTOR, INC. **M68HC11E Family Data Sheet**. USA, June 2003. Available at: <http://www.freescale.com>. Visited on July 2005.

GAISLER, J. Evaluation of a 32-Bit Microprocessor with Built-in Concurrent Error-Detection. In: INTERNATIONAL SYMPOSIUM ON FAULT-TOLERANT COMPUTING, FTCS, 27., 1997. **Digest of Papers...** [S.l.]: IEEE, 1997. p. 42-46.

GRANLUND, T.; GRANBOM, B.; OLSSON, N. Soft Error Rate Increase for New Generations of SRAMs. **IEEE Transactions on Nuclear Science**, [S.l.], v.50, n.6, p. 2065-2068, Dec. 2003.

GROCHOWSKI, A. et al. Integrated Circuit Testing for Quality Assurance in Manufacturing: History, Current Status, and Future Trends. **IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing**, [S.l.], v.44, n.8, p. 610-633, Aug. 1997.

GÜNTZEL, J. L. A. **Functional Timing Analysis of VLSI Circuits Containing Complex Gates**. 2000. 182 f. Thesis (Ph.D) – PPGC, Instituto de Informática, UFRGS, Porto Alegre.

IYER, R. K. et al. Recent Advances and New Avenues in Hardware-Level Reliability Support. **IEEE Micro**, [S.l.], v.25, n.6, p. 18-29, Nov.-Dec. 2005.

HARBOE-SORENSEN, R.; SUND, A. T. Radiation Pre-Screening of R3000/R3000A Microprocessors. In: RADIATION EFFECTS DATA WORKSHOP, 1992. **Workshop Record...** [S.l.]: IEEE, 1992. p. 34-41.

HASS, J. et al. Mitigating Single Event Upsets From Combinational Logic. In: NASA SYMPOSIUM ON VLSI DESIGN, 7., 1998. **Proceedings...** [S.l.: s.n.], 1998.

HASS, J. Probabilistic Estimates of Upset Caused by Single Event Transients. In: NASA SYMPOSIUM ON VLSI DESIGN, 8., 1999. **Proceedings...** [S.l.: s.n.], 1999.

HAZUCHA, P. et al. Neutron Soft Error Rate Measurements in a 90-nm CMOS Process and Scaling Trends in SRAM from 0.25-µm to 90-nm Generation. In: INTERNATIONAL ELECTRON DEVICES MEETING, IEDM, 2003. **Technical Digest...** [S.l.]: IEEE, 2003. p. 21.5.1-21.5.4.

HENES-NETO, E.; WIRTH, G.; KASTENSMIDT, F. L. A. Using Bulk Built-In Current Sensors to Detect Transient Faults in SRAM Memory Architectures. In: LATIN AMERICAN TEST WORKSHOP, LATW, 7., March 26-29, 2006, Buenos Aires, Argentina. **Proceedings...** [S.l.]: IEEE Computer Society, 2006.

HENTSCHKE, R.; MARQUES, F.; LIMA, F.; CARRO, L.; SUSIN, A.; REIS, R. Analyzing Area and Performance Penalty of Protecting Different Digital Modules with Hamming Code and Triple Modular Redundancy. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, SBCCI, 15., 2002, Porto Alegre. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p. 95-100.

HERRERA, F. et al. Specification Components: Reusability at the HW/SW System Specification Level. In: FALL VIUF WORKSHOP, 1999. **Proceedings...** [S.l.]: IEEE, 1999. p. 50-56.

HOWARD, J. W. J. et al. Total Dose and Single Event Effects Testing of the Intel Pentium III (P3) and AMD K7 Microprocessors. In: RADIATION EFFECTS DATA WORKSHOP, 2001. **Proceedings...** [S.l.]: IEEE, 2001. p. 38-47.

HUTCHESON, G. D. Os Primeiros Nanochips. **Scientific American Brasil**, [S.l.], n.24, p. 68-75, maio 2004.

JOHNSTON, A. Scaling and Technology Issues for Soft Error Rates. In: RESEARCH CONFERENCE ON RELIABILITY, 4., 2000. **Proceedings...** Palo Alto: Stanford University, 2000.

KARNIK, T.; HAZUCHA, P.; PATEL, J. Characterization of Soft Errors Caused by Single Event Upsets in CMOS Processes. **IEEE Transactions on Dependable and Secure Computing**, [S.l.], v.1, n.2, p. 128-143, Apr.-June 2004.

KASTENSMIDT, F. L.; CARRO, L.; REIS, R. **Fault-Tolerance Techniques for SRAM-Based FPGA**. [S.l.]: Springer, 2006.

KIM, N. S. et al. Leakage Current: Moore's Law Meets Static Power. **IEEE Computer**, [S.l.], v.36, n.12, p. 68-75, Dec. 2003.

KRISHNAMOHAN, S.; MAHAPATRA, N. R. A Highly-Efficient Technique for Reducing Soft Errors in Static CMOS Circuits. In: COMPUTER DESIGN: VLSI IN COMPUTERS AND PROCESSORS, ICCD, 2004. **Proceedings...** [S.l.]: IEEE Computer Society, 2004. p. 126-131.

LABEL, K. A. et al. Commercial Microelectronics Technologies for Applications in the Satellite Radiation Environment. In: AEROSPACE APPLICATIONS CONFERENCE, 1996. **Proceedings...** [S.l.]: IEEE, 1996. p. 375-390.

LABEL, K. A. et al. A Roadmap for NASA's Radiation Effects Research in Emerging Microelectronics and Photonics. In: AEROSPACE CONFERENCE, 2000. **Proceedings...** [S.l.]: IEEE, 2000. p. 535-545.

LAPRIE, J. Dependability of Computer Systems: from Concepts to Limits. In: IFIP INTERNATIONAL WORKSHOP ON DEPENDABLE COMPUTING AND ITS APPLICATIONS, DCIA, 1998. **Proceedings...** Johannesburg: University of the Witwatersrand, 1998. p. 108-126.

LAMBERT, D. et al. Neutron-Induced SEU in Bulk SRAMs in Terrestrial Environment: Simulations and Experiments. **IEEE Transactions on Nuclear Science**, [S.l.], v.51, n.6, p. 3435-3441, Dec. 2004.

LAZZARI, C.; ANGHEL, L.; REIS, R. On Implementing a Soft Error Hardening Technique by Using an Automatic Layout Generator: Case Study. In: INTERNATIONAL ON-LINE TESTING SYMPOSIUM, IOLTS, 11., 2005. **Proceedings...** [S.l.]: IEEE Computer Society, 2005. p. 29-34.

LERAY, J. et al. Atmospheric Neutron Effects in Advanced Microelectronics, Standards and Applications. In: INTERNATIONAL CONFERENCE ON INTEGRATED CIRCUIT DESIGN AND TECHNOLOGY, ICICDT, 2004. **Proceedings...** [S.l.]: IEEE, 2004. p. 311-321.

LIDÉN, P. et al. On Latching Probability of Particle Induced Transients in Combinational Networks. In: INTERNATIONAL SYMPOSIUM ON FAULT-TOLERANT COMPUTING, FTCS, 24., 1994. **Digest of Papers...** [S.l.]: IEEE, 1994. p. 340-349.

LIMA, F.; COTA, E.; CARRO, L.; LUBASZEWSKI, M.; REIS, R.; VELAZCO, R.; REZGUI, S. Designing a Radiation Hardened 8051-Like Micro-Controller. In: SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN, SBCCI, 13., 2000. **Proceedings...** Los Alamitos: IEEE Computer Society, 2000-a. p. 255-260.

LIMA, F.; REZGUI, S.; COTA, E.; CARRO, L.; LUBASZEWSKI, M.; VELAZCO, R.; REIS, R. Designing and Testing a Radiation Hardened 8051-like Micro-controller. In: INTERNATIONAL CONFERENCE ON MILITARY AND AEROSPACE APPLICATIONS OF PROGRAMMABLE LOGIC DEVICES, MAPLD, 2000. **Proceedings...** [S.l.: s.n.], 2000-b.

LIMA, F. **Single Event Upset Mitigation Techniques for Programmable Devices**. 2000-c. 102 f. Qualifying Examination (Ph.D) – PPGC, Instituto de Informática, UFRGS, Porto Alegre.

LIMA, F.; REZGUI, S.; CARRO, L.; VELAZCO, R.; REIS, R. On the Use of VHDL Simulation and Emulation to Derive Error Rates. In: EUROPEAN CONFERENCE ON RADIATION AND ITS EFFECTS ON COMPONENTS AND SYSTEMS, RADECS, 2001. **Proceedings...** [S.l.]: IEEE Computer Society, 2001-a. p. 253-260.

LIMA, F.; CARMICHAEL, C.; FABULA, J.; PADOVANI, R.; REIS, R. A Fault Injection Analysis of Virtex FPGA TMR Design Methodology. In: EUROPEAN CONFERENCE ON RADIATION AND ITS EFFECTS ON COMPONENTS AND SYSTEMS, RADECS, 2001. **Proceedings...** [S.l.]: IEEE Computer Society, 2001-b. p. 275 -282.

LIMA, F.; CARRO, L.; VELAZCO, R.; REIS, R. Injecting Multiple Upsets in a SEU Tolerant 8051 Micro-Controller. In: LATIN AMERICAN TEST WORKSHOP, LATW, 2002. **Proceedings**... Amissville: IEEE Computer Society, 2002-a.

LIMA, F.; CARRO, L.; VELAZCO, R.; REIS, R. Injecting Multiple Upsets in a SEU Tolerant 8051 Micro-Controller. In: IEEE INTERNATIONAL ON-LINE TESTING WORKSHOP, IOLTW, 8., 2002. **Proceedings...** [S.l.]: IEEE Computer Society, 2002-b. p. 194.

LIMA, F.; CARRO, L; REIS, R. Techniques for Reconfigurable Logic Applications: Designing Fault Tolerant Systems into SRAM-based FPGAs. In: INTERNATIONAL DESIGN AUTOMATION CONFERENCE, DAC, 2003. **Proceedings...** New York: ACM, 2003-a. p. 650-655.

LIMA, F. **Designing Single Event Upset Mitigation Techniques for Large SRAM-Based FPGA Components**. 2003-b. 157 f. Thesis (Ph.D) – PPGC, Instituto de Informática, UFRGS, Porto Alegre.

LUBASZEWSKI, M.; HUERTAS, J. L. Test and Design-For-Test of Mixed-Signal Integrated Circuits. In: REIS, R. (Ed.). **Information Technology:** Selected Tutorials. Boston: Kluwer Academic, 2004. p.183-212.

MA, T.; DRESSENDORFER, P. **Ionizing Radiation Effects in MOS Devices and Circuits**. New York: John Wiley & Sons, 1989.

MAHESHWARI, A.; KOREN, I.; BURLESON, N. Techniques for Transient Fault Sensitivity Analysis and Reduction in VLSI Circuits. In: INTERNATIONAL SYMPOSIUM ON DEFECT AND FAULT TOLERANCE IN VLSI SYSTEMS, 18., 2003. **Proceedings...** [S.l.]: IEEE, 2003. p. 597-604.

MAIZ, J. et al. Characterization of Multi-bit Soft Error Events in Advanced SRAMs. In: INTERNATIONAL ELECTRON DEVICES MEETING, IEDM, 2003. **Technical Digest...** [S.l.]: IEEE, 2003. p. 21.4.1-21.4.4.

MASSENGILL, L. W. et al. Analysis of Single-Event Effects in Combinational Logic – Simulation of the AM2901 Bitslice Processor. **IEEE Transactions on Nuclear Science**, Reno, NV, USA, v.47, n.6, p. 2609-2615, Dec. 2000.

MENTOR GRAPHICS CORPORATION. **ModelSim Manuals**. USA, March 2004.

NEVES, C.; HENES-NETO, E.; RIBEIRO, I.; WIRTH, G.; KASTENSMIDT, F. L.; GUNTZEL, J. L. A. Automatic Evaluation of Single Event Transient Propagation in CMOS Logic Circuits Based on Topological Timing Analysis. In: LATIN AMERICAN TEST WORKSHOP, LATW, 7., March 26-29, 2006, Buenos Aires, Argentina. **Proceedings...** [S.l.]: IEEE Computer Society, 2006-a.

NEVES, C.; HENES-NETO, E.; RIBEIRO, I.; WIRTH, G.; KASTENSMIDT, F. L.; GUNTZEL, J. L. A. Avoiding Circuit Simulation for the Analysis of Single Event Transient Propagation in Combinational Circuits. In: EUROPEAN TEST SYMPOSIUM, ETS, 2006. **Proceedings...** [S.l.]: IEEE, 2006-b.

NICOLAIDIS, M. Time Redundancy Based Soft-Error Tolerance to Rescue Nanometer Technologies. In: VLSI TEST SYMPOSIUM, 17., 1999. **Proceedings...** [S.l.]: IEEE Computer Society, 1999. p. 86-94.

NICOLAIDIS, M.; PEREZ, R. Measuring the Width of Transient Pulses Induced by Ionising Radiation. In: INTERNATIONAL RELIABILITY PHYSICS SYMPOSIUM, 41., 2003. **Proceedings...** [S.l.]: IEEE, 2003. p. 56-59.

NORMAND, E.; BAKER, T. J. Altitude and Latitude Variations in Avionics SEU and Atmospheric Neutron Flux. **IEEE Transactions on Nuclear Science**, New York, v.40, n.6, p. 1484-1490, Dec. 1993.

NORMAND, E. et al. Single Event Upset and Charge Collection Measurements Using High Energy Protons and Neutrons. **IEEE Transactions on Nuclear Science**, [S.l.], v.41, n.6, p. 2203-2209, Dec. 1994.

NORMAND, E. Single-Event Effects in Avionics. **IEEE Transactions on Nuclear Science**, [S.l.], v.43, n.2, p. 461-474, Apr. 1996-a.

NORMAND, E. Single Event Upset at Ground Level. **IEEE Transactions on Nuclear Science**, New York, v.43, n.6, p. 2742-2750, Dec. 1996-b.

NORMAND, E. Correlation of In-Flight Neutron Dosimeter and SEU Measurements with Atmospheric Neutron Model. **IEEE Transactions on Nuclear Science**, New York, v.48, n.6, p. 1996-2003, Dec. 2001.

O'BRYAN, M. V. et al. Single Event Effect and Radiation Damage Results for Candidate Spacecraft Electronics. In: RADIATION EFFECTS DATA WORKSHOP, 1998. **Proceedings...** [S.l.]: IEEE, 1998. p. 39-50.

O'BRYAN, M. V. et al. Recent Single Event Effects Results for Candidate Spacecraft Electronics for NASA. In: RADIATION EFFECTS DATA WORKSHOP, 2005. **Proceedings...** [S.l.]: IEEE, 2005. p. 26-35.

SAGGESE, P. G. et al. An Experimental Study of Soft Errors in Microprocessors. **IEEE Micro**, [S.l.], v.25, n.6, p. 30-39, Nov.-Dec. 2005.

SEE SYMPOSIUM, FIFTEENTH BIENNIAL. April 10-12, 2006, Long Beach, USA. Available at: <http://radhome.gsfc.nasa.gov/radhome/SEE/seesym.htm>. Visited on April 2006.

SHIVAKUMAR, P. et al. Modeling the Effect of Technology Trends on the Soft Error Rate of Combinational Logic. In: INTERNATIONAL CONFERENCE ON DEPENDABLE SYSTEMS AND NETWORKS, DSN, 2002. **Proceedings...** [S.l.]: IEEE Computer Society, 2002. p. 389-398.

SMITH, M. J. S. **Application-Specific Integrated Circuits**. Reading: Addison-Wesley, 1997.

SROUR, J. R.; MARSHALL, C. J.; MARSHALL, P. W. Review of Displacement Damage Effects in Silicon Devices. **IEEE Transactions on Nuclear Science**, [S.l.], v.50, n.3, p. 653-670, June 2003.

SYNOPSYS, INC. **Tool Manuals**. USA, June 2004.

THIBAULT, S. **GM HC11 CPU Core**. USA, August 2000. Available at: <http://www.gmvhdl.com/hc11core.html>. Visited on March 2006.

TOSAKA, Y. et al. Measurement and Analysis of Neutron-Induced Soft Errors in Sub-Half-Micron CMOS Circuits. **IEEE Transactions on Electron Devices**, [S.l.], v.45, n.7, p. 1453-1458, July 1998.

VAHID, F.; GORDON-ROSS, A. A Self-Optimizing Embedded Microprocessor using a Loop Table for Low Power. In: INTERNATIONAL SYMPOSIUM ON LOW POWER ELECTRONICS AND DESIGN, ISLPED, August 6-7, 2001, Huntington Beach, California, United States. **Proceedings...** New York, NY, USA: ACM Press, 2001. p. 219-224.

VELAZCO, R.; KAROUI, S.; CHAPUIS, T. SEU Testing of 32-Bit Microprocessors. In: RADIATION EFFECTS DATA WORKSHOP, 1992. **Workshop Record...** [S.l.]: IEEE, 1992. p. 16-20.

WAGNER, F. R. **Metodologias de Projeto**. Aula 2 da Disciplina de Arquitetura e Projeto de Sistemas VLSI I, 2004. PPGC, Instituto de Informática, UFRGS, Porto Alegre.

ZHANG, M.; SHANBHAG, N. R. An Energy-efficient Circuit Technique for Single Event Transient Noise-Tolerance. In: INTERNATIONAL SYMPOSIUM ON CIRCUITS AND SYSTEMS, ISCAS, 2005. **Proceedings...** [S.l.]: IEEE, 2005. p. 636-639.

ZIEGLER, J. F. et al. IBM Experiments in Soft Fails in Computer Electronics (1978-1994). **IBM Journal of Research and Development**, [S.l.], v.40, n.1, p. 3-18, Jan. 1996.

ZIEGLER, J. F. et al. Cosmic Ray Soft Error Rates of 16-Mb DRAM Memory Chips. **IEEE Journal of Solid-State Circuits**, [S.l.], v.33, n.2, p. 246-252, Feb. 1998.

APPENDIX PROJETO DE UM MICROPROCESSADOR ROBUSTO A SOFT ERRORS

Resumo da Dissertação em Português

O avanço das tecnologias de circuitos integrados (CIs) levanta importantes questões relacionadas à confiabilidade e à robustez de sistemas eletrônicos. A diminuição da geometria dos transistores, a redução dos níveis de tensão, as menores capacitâncias e portanto menores correntes e cargas para alimentar os circuitos, além das freqüências de relógio elevadas, têm tornado os CIs mais vulneráveis a falhas, especialmente àquelas causadas por ruído elétrico ou por efeitos induzidos pela radiação.

Os efeitos induzidos pela radiação conhecidos como Soft Single Event Effects (Soft SEEs) podem ser classificados em: Single Event Upsets (SEUs) diretos em nós de elementos de armazenagem que resultam em inversões de bits; e pulsos transientes Single Event Transients (SETs) em qualquer nó do circuito. Especialmente SETs em circuitos combinacionais podem se propagar até os elementos de armazenagem e podem ser capturados. Estas errôneas armazenagens podem também serem chamadas de SEUs indiretos.

Falhas como SETs e SEUs podem provocar erros em operações funcionais de um CI. Os conhecidos Soft Errors (SEs) são caracterizados por valores armazenados erradamente em elementos de memória durante o uso do CI. SEs podem produzir sérias conseqüências em aplicações de CIs devido à sua natureza não permanente e não recorrente. Por essas razões, mecanismos de proteção para evitar SEs através de técnicas de tolerância a falhas, no mínimo em um nível de abstração do projeto, são atualmente fundamentais para melhorar a confiabilidade de sistemas.

Nos dias atuais, a complexidade dos circuitos através de System-On-Chips (SOCs), o usual time-to-market e as restrições orçamentárias de projeto têm levado projetistas a investigar técnicas de tolerância a falhas e fluxos de projeto mais versáteis. A reusabilidade de IPs de hardware desenvolvidos em alto nível e fluxos de projeto para CIs baseados em ferramentas de CAD auxiliam engenheiros a enfrentar tais exigências. Por outro lado, algumas técnicas de tolerância a falhas podem implicar em modificações indesejadas em características padrões de um sistema, especialmente quando o alvo é a reusabilidade de sistemas baseados em arquiteturas padrões como microprocessadores comerciais. Por exemplo, algumas técnicas exigem redes de relógio adicionais para detecção de falhas e ciclos de relógio extras para correção de falhas. A fim de economizar tempo de projeto e custo de desenvolvimento, deseja-se geralmente que as técnicas escolhidas não somente garantam a confiabilidade e reusabilidade de suas aplicações de hardware e software. Também se deseja que elas sejam facilmente ou no mínimo aplicáveis no nível de projeto alvo e que elas se adaptem a cores comerciais padrões.

Microprocessadores tais como algumas arquiteturas AMD, IBM e Intel (LIMA et al, 2000-a, 2000-b; COTA et al, 2001; IYER et al, 2005) geralmente são protegidas contra SEUs diretos, mas não usualmente contra SEUs indiretos. As condições tecnológicas e a redução dos transistores tendem a exigir proteções também contra tais SEUs indiretos (SHIVAKUMAR et al, 2002).

O propósito deste trabalho de dissertação é robustecer a SEs um microprocessador comercial 8-bits da família M68HC11 (FREESCALE, 2002) para a fabricação futura de um IC. A fim de economizar tempo de projeto, algumas restrições iniciais de projeto foram estabelecidas. O projeto do circuito tolerante a falhas deveria ser desenvolvido em alto nível como o nível RT. As técnicas de tolerância a falhas implementadas não deveriam usar múltiplas redes de relógio. Para qualquer aplicação, as técnicas deveriam preservar o número total de ciclos de relógio, mesmo que sob uma ocorrência de falha. Tais restrições iniciais mantêm as características da arquitetura padrão e assim a reusabilidade de aplicações do microprocessador. Além disto, estas restrições economizam custo de desenvolvimento.

SETs em circuitos combinacionais do microprocessador, que podem potencialmente causar SEUs indiretos, são aliviados através do uso de uma técnica de Redundância no Tempo (TR). O trabalho em (NICOLAIDIS, 1999) sugere mas não implementa uma abordagem de TR baseada em um elemento especial chamado Code Word State Preserving (CWSP) como aquele da Figura 1 (a). Um outro trabalho (ANGHEL; ALEXANDRESCU; NICOLAIDIS, 2000-b) avalia essa abordagem em área e performance através do uso de simples circuitos de teste, como somadores e multiplicadores, e portas não-padronizadas, tais como aquelas da Figura 1 (b), para implementar os elementos CWSP. Em (LAZZARI; ANGHEL; REIS, 2005), a mesma avaliação é feita para dois microprocessadores, MIPS e 8051, mas um gerador automático especial de layout implementa as portas não-padronizadas que caracterizam os elementos CWSP. A fim de aliviar SEUs diretos, em (LAZZARI; ANGHEL; REIS, 2005) uma versão da técnica de Redundância Modular Tripla (TMR) que exige três sinais de relógio foi também implementada. No presente trabalho de dissertação foi implementada uma alternativa mais simples e rápida para projetar por meio do uso de apenas portas padrões, como aquela da Figura 1 (a), e sem uma ferramenta de layout extra como aquela apresentada em (LAZZARI; ANGHEL; REIS, 2005). As definidas restrições iniciais de projeto são encontradas através desta alternativa. A meta foi avaliar os custos em área, performance e também potência e outros resultados de projeto dessa abordagem de tolerância a falhas no microprocessador alvo. Além disto, os elementos do esquema TR+CWSP e os registradores do microprocessador foram protegidos de acordo com a Figura 1 (c) através do uso de uma versão da técnica TMR que exige apenas um sinal de relógio para aliviar os SEUs diretos.

A fim de obter os custos da robustez implementada na CPU alvo através do uso de apenas portas padrões, três versões do microprocessador foram desenvolvidas baseadas em uma descrição VHDL M68HC11 (THIBAULT, 2000): **versão Não-Protegida**, que é a arquitetura reorganizada da CPU sem qualquer mecanismo de tolerância a falhas; **versão TMR** que é somente protegida por TMR nos registradores e assim ela alivia apenas SEUs diretos; e **versão TMR+TR+CWSP** que é a versão robusta a SEUs diretos e indiretos através do esquema TMR+TR+CWSP. As três versões foram implementadas usando um fluxo de projeto para CIs baseado em células padrões em tecnologia CMOS (AMS 0.35 µm, 4 níveis de metal, 3.3V) (AUSTRIAMICROSYSTEM, 2003) e ferramentas de CAD (CADENCE, 2002; MENTOR, 2004) para simulação, síntese, posicionamento, roteamento, extração, verificação e análise.

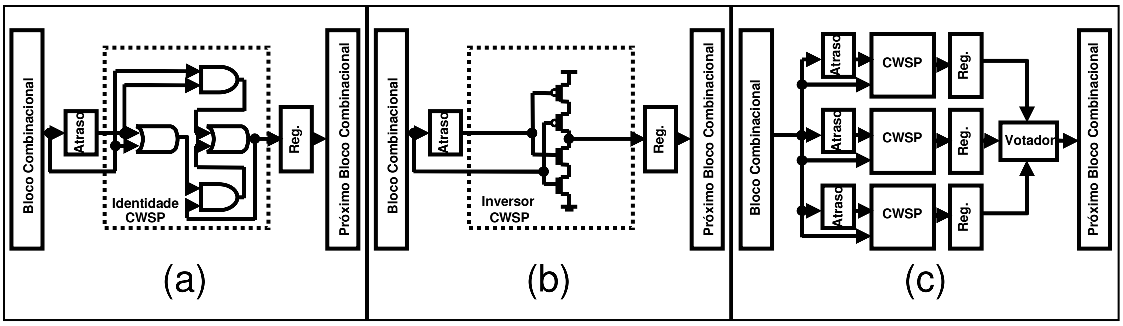


Figura 1: Um esquema de proteção TR+CWSP através de portas padrões (a) e através de portas não-padronizadas (b). Em (c), o esquema de proteção TMR+TR+CWSP usado

Três abordagens de simulação para verificação do projeto foram desenvolvidas para cada uma das versões do microprocessador com o objetivo de detectar eventuais erros de projeto. Um experimento para testagem funcional foi realizado através de simulações de verificação comportamental e simulações de verificação pré-layout e pós-layout no nível de portas. Um experimento de injeção de falhas foi feito por meio das simulações de verificação pós-layout no nível de portas. Além disso, uma análise de timing estática e um DRC foram realizados através de ferramentas de CAD.

A Tabela 1 mostra os resultados de projeto estimados das três versões do microprocessador que foram desenvolvidas.

Tabela 1: Resultados dos projetos em área, performance e outros

Versões do Microprocessador				
Não-Protegida	**TMR** Protegida contra SEUs Diretos		**TMR+TR+CWSP** Protegida contra SEUs Diretos e Indiretos	
Área do Core (mm^2)				
0,397	0,569	+ 43,32 %	0,804	**+ 102,52 %**
Pior Tempo de Chegada (ns)				
69,45	75,93	+ 9,33 %	78,29	**+ 12,73 %**
Comprimento Total das Conexões (mm)				
450,49	527,72	+ 17,14 %	668,64	**+ 48,43 %**
Área Exigida pelos Elementos da Árvore do Relógio (µm^2)				
4076,8	7516,6	+ 84,38 %	14414,4	**+ 253,57 %**
Consumo de Potência Leakage (µW)				
0,2440	0,3645	+ 49,39 %	0,5024	**+ 105,90 %**

Observe na Tabela 1 que para a implementação TMR+TR+CWSP a performance foi afetada em 12,73 % como um resultado da restrição do máximo pulso de SET (cerca de 1 ns), os buffers para o bloco de Atraso (para produzir a comparação pelo princípio da TR) e as portas para os blocos CWSP e Votador. O esquema de proteção da Figura 1 (c) implementado em todos os registradores aumentou a área em 102,52 % e a potência estática em 105,90 %. O comprimento total de todas as conexões geradas pelo roteamento foi aumentado em 48.43 %. Os custos extras em área devido as inserções de buffers e inversores para construir a árvore do relógio são também detalhados na Tabela 1. A versão Não-Protegida é constituída de 3211 células padrões combinacionais (+79,76 % para versão TMR+TR+ CWSP) e de 187 células padrões seqüenciais (+200 % para versão TMR+TR+CWSP). A área das células combinacionais da versão TMR+TR+CWSP corresponde a 51,90 % da area do core, a área de células seqüenciais 19,05 % e a área de células filler 29,05 % (ou seja, o espaço para o roteamento finalizar com sucesso). O layout final com pads da versão de CI TMR+TR+CWSP resultou em uma área total de cerca de 5,707 mm^2.