CIRCUMVENTING RADIATION EFFECTS BY LOGIC DESIGN
Reference AO/1-3240/97/NL/FM

WP500 REPORT - COOKBOOK

	Name and function	Date	Signature
Prepared by :	Nicolas PERROT (MMS)	12/07/99	
Verified by :	Marc SOUYRI (MMS)	12/07/99	
Authorized by :	Jean-François COLDEFY (MMS)	12/07/99	

Document type	Nb WBS	Keywords :

DOCUMENT CHANGE LOG

Issue / Revision	Date	Modification Nb	Modified pages	Observations
00	19/05/99		all	

PAGE ISSUE RECORD

Issue of this document comprises the following pages at the issue shown

Page	**Issue / Revision**	**Page**	**Issue / Revision**	**Page**	**Issue / Revision**	**Page**	**Issue / Revision**
all	00						

SUMMARY

1 INTRODUCTION

This document is the WP500 report, part of the ESTEC R&D " Circumventing Radiation Effects by Logic Design " reference AO/1-3240/97/NL/FM.

One of the objectives of this R&D is to write a design manual for helping the designers to take into account the Single Event Upset (SEU) and Single Event Latchup (SEL) aspects. This manual will firstly describe the space origin of the ionising particles leading to Single Event Upsets in electronic systems, and will then give some explanations about the physical aspects of Single Event Effects (SEE). The manual will then be written as a cookbook, giving "design recipes" for chip protections against SEU and SEL. These recipes can be either at function level or at cell level for ASIC design.

The first part of this document will explain in simple terms the radiation effects on micro electronic devices and the space origin of the ionised particles (WP100).

The second part will describe methods that can be used to mitigate Single Event Upset effects for VLSI, at functional block design level (WP210), and then at cell design level (WP220).

The third part will focus on the design methods to protect components against Single Event Latchup induced by heavy ions (WP300). These methods use anti-latchup circuitry on PCBs, and the choice of the adequate solution will depend on the component to protect.

ACRONYMS

ALE : Anomalously Large Event (concerns solar flares)

ALU : Arithmetic and Logic Unit

AOCS : Attitude and Orbit Control System

ASIC : Application Specific Integrated Circuit

ASSP : Application Specific Standard Product

BCH : Bose Chaudhuri Hocquenghem (BCH code)

BED : Byte Error Detection

CCSDS : Consultative Committee for Space Data Systems

CMOS : Complementary Metal Oxide Semiconductor

CRC : Cyclic Redundant Checker

CREME : Cosmic Ray Effects on MicroElectronics

CRIER : Cosmic Ray Induced Error-Rate analysis

CRUP : Cosmic Ray Upset Program

DBD : Double Byte error Detection

DEC : Double Error Correction

DED : Double Error Detection

DFF : D-type Flip Flop

DICE : Dual Interlocked storage Cell

DRAM : Dynamic Random Access Memory

DRC : Design Rule Check

ECC : Error Correcting Code

ELDO : SPICE simulator

FPGA : Field Programmable Gate Array

FSM : Finite State Machine

GCR : Galactic Cosmic Rays

GEO : Geo-synchronous orbit

HEO : Highly Elliptical Orbit

HiRel : High Reliability

HIT : Heavy Ion Tolerant

HSPICE : SPICE simulator

IFL : Input Forming Logic (for finite state machines)

ITT : Invitation To Tender

JEDEC : Joint Electron Device Engineering Council

JTAG : Join Test Action Group

LEO : Low Earth Orbit

LET : Linear Energy Transfer (unit: MeV.cm^2/mg or MeV/mg/cm^2)

LU : Latch Up

LUDPC : Latch Up Detection and Protection Circuit

MBU : Multiple Bit Upset

MEO : Middle Earth Orbit

MMS : Matra Marconi Space

ORE : Ordinary Events (concerns solar flares)

PCB : Printed Circuit Board

PCM : Parity Check Matrix (for Hamming code)

PMOS : p-type MOS

RAM : Random Access Memory

MOS : Metal Oxide Semiconductor

MTTF : Mean Time To Failure

NMOS : n-type MOS

OFL : Output Forming Logic (for finite state machines)

SAA : South Atlantic Anomaly

SBC : Single Byte error Correction

SE : Single Event

SEB : Single Event Burnout

SEC : Single Error Correction

SEE : Single Event Effect ((or SEP Single Event Phenomenon)

SEL : Single Event Latchup

SEP : Single Event Phenomenon (see SEE)

SEU : Single Event Upset

SF : Solar Flare

SOI : Silicon On Insulator

SOS : Silicon On Sapphire

SPICE : Simulator Program with Integrated Circuits Emphasis

SRAM : Static Random Access Memory

SSO : Sun Synchronous Orbit

TE : Trapped Electrons

TED : Triple Error Detection

TMR : Triple Modular Redundancy

TP : Trapped Protons

VLSI : Very Large Scale Integration (component)

WP : Work Package

2 SPACE RADIATION ENVIRONMENT

SEU is a consequence of the random bombardment of integrated circuits by high energy ionising particles such as protons and heavy ions. This section will firstly define some useful notions, and will then give some explanations about the space origin of these particles. A summary table will close the section.

2.1 DEFINITIONS

2.1.1 Linear Energy Transfer (LET)

The heavy ions impinging a device can deposit more or less energy depending on their initial energy and their mass. The charge deposition capacity is generally described in terms of Linear Energy Transfer (LET) which corresponds to the energy deposition by length unit for a given material (most often silicon) :

$$LET = \frac{\Delta E}{\Delta x} \text{ (in MeV/cm for a given material)}$$

This definition is not easily applicable, since it depends on the material. A normalized LET is thus defined by taking into account the material density, ρ :

$$LET = \frac{1}{\rho} \cdot \frac{\Delta E}{\Delta x} \text{ (in MeV.cm}^2\text{/mg or MeV/mg/cm}^2\text{)}$$

This second definition is the most often used.

The deposited energy is equal to : $\Delta E = \frac{dE}{dx} . \frac{X}{\cos\theta}$, if θ is the ion incident angle.

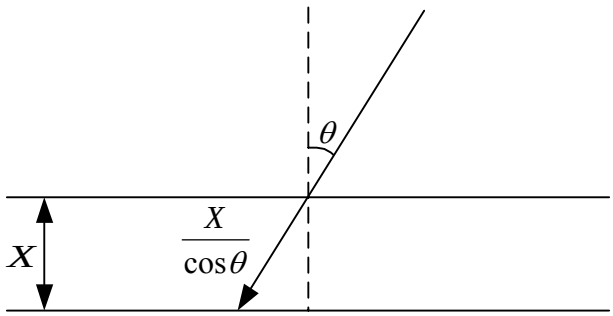

figure 2.1.1–1 – deposited energy for a heavy ion with a θ incident angle

If ΔE > Ec (critical energy), a single event phenomenon occurs. A LET threshold can thus be defined: it is the minimum LET that causes an upset effect. The more the LET threshold is high, the less the component is sensitive to single events. The JEDEC recommended definition for the LET threshold is the first effect when the particle fluence is 10^7 ions/cm^2.

The following table gives an idea of the component sensitivity against SEU according to the LET threshold:

LET threshold range (LET th)	Component sensitivity against SEU
LET th < 12 MeV.cm^2/mg	**Very sensitive component** **can be upset by proton induced events**
12 MeV.cm^2/mg < LET th < 36 MeV.cm^2/mg	**Sensitive component**
36 MeV.cm^2/mg < LET th < 110 MeV.cm^2/mg	**Low sensitive component**
110 MeV.cm^2/mg < LET th	**Insensitive component in a space environment**

For Single Event Latchup (SEL), the component is considered latchup free if the LET threshold is greater than 70 to 100 MeV.cm^2/mg.

2.1.2 Cross section

The cross section σ is defined as the ratio of the number of Single Event registered on the device by the ion fluence (event per cm^2). It corresponds to the probability that an ion impinging in a normal direction 1 cm^2 of the device triggers a single event.

2.1.3 Cross section curve (for a given device and SEE)

The cross section curve is representative of the sensitivity of the device to a given SEE (SEB, SEL, SEU). Such a cross section curve is obtained by recording the single event number of the device under heavy ions beam, following test in accelerator. The heavy ion LET can be adjusted by varying the heavy ions species or the ion energy or the tilt of the beam.

The LET is represented in the x-axis, the corresponding error rate or cross section is represented in the y-axis. For the lowest LETs, the error rate increases until it reaches a saturation region (i.e. the error rate remains stable even if the LET increases). The LET threshold can be evaluated from this curve, it is the LET that corresponds to the beginning of the saturation region (in practice, the LETthreshold corresponds to a cross section equal to $\sigma_{saturation}/10$). An example of SEU cross section curve is given hereafter, the LET threshold is approximately 15 MeV.cm^2/mg in this example.

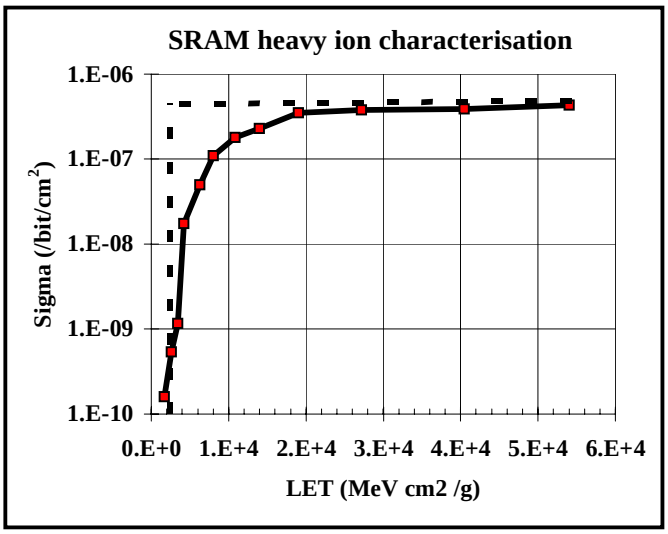

Figure 2.1.3–1 – example of SEU cross section curve induced by heavy ions

2.1.4 Integral LET spectrum (for a given mission)

The integral LET spectrum shows the number of ions impinging the device for a given orbit, solar activity and material shielding. It represents the number of particles with LET ≥ LETc that will hit a particular area per unit of time. It presents in the Y-axis the flux of ions F (in /m^2/sr/s or /cm^2/day) having a LET higher to the one defined in the X-axis. The flux F is high for low LETs and it decreases as far as the LET increases.

This spectrum is a representation of the heavy ion environment, independent from the components.

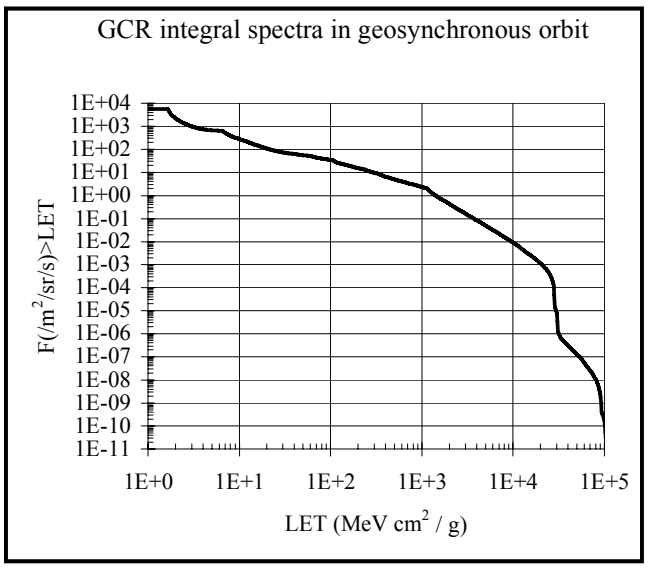

Figure 2.1.4–1 – example of integral LET spectrum

2.2 SEE RATE PREDICTION

The combination of the cross section curve (relative to the device sensitivity) and the integral LET spectrum (relative to the heavy ion environment of a given mission) gives a first approximation in orbit error prediction by convoluting the integral LET to the cross section curve.

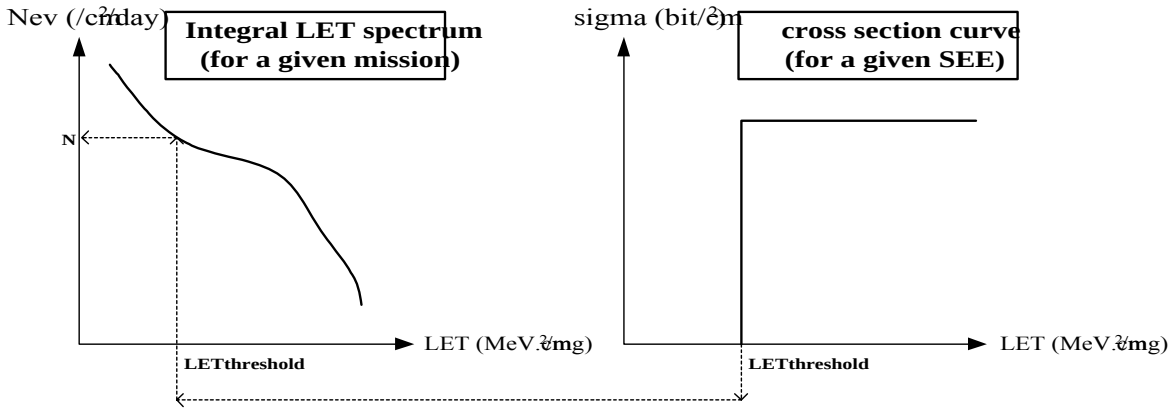

Figure 2.2–1 – SEE rate prediction from integral LET spectrum and cross section curve

This method is however rather approximate because every ion in space does not impinge the device normally to its surface (as in accelerator). A precise routine called "UPSET" included in CREME program [ADAM2] allows to consider the omnidirectional nature of the heavy ion flux encountered in space.

2.3 SOLAR FLARES

The sun emits sporadically bursts of energetic charged particles into the interplanetary space (solar wind). These flares are composed primarily of protons and a minor constituent of alpha particles (5-10%), heavy ions and electrons. Solar flares occur mainly during the so-called « Solmax » period, which lasts 7 years on a full solar cycle of 11 years (4 years at Solmin) [STAS]. Their intensity is highly variable (ordinary events (ORE) and anomalously large events (ALE) that produce a fluence at earth at least twice that of all the other flares in the active period), their composition is also highly variable (most of the solar flares do not include heavy ions, but when heavy ions are present, their relative energy spectrum and abundance is highly variable)[CHEN][GAR]. Sunspot number is representative of the solar activity. Figure 2.3-1 presents the solar activity during the cycles 9 to 22 (up to 1996) and the forecast for the cycle 23 (up to 2007). As shown on this figure, Solmax period is expected during year 2000 and Solmin period during 2006.

Solar flare flux can be completely screened by the geomagnetic field for equatorial low altitude orbits. As a consequence, solar flare risk analysis must be considered mainly for high altitude orbits such as geosynchronous or far missions.

Satellite shielding has a major influence on the solar flare ion flux. Due to their relative low energy, solar flare particles are slowed down and can be completely stopped by the satellite structure before impacting electronic devices.

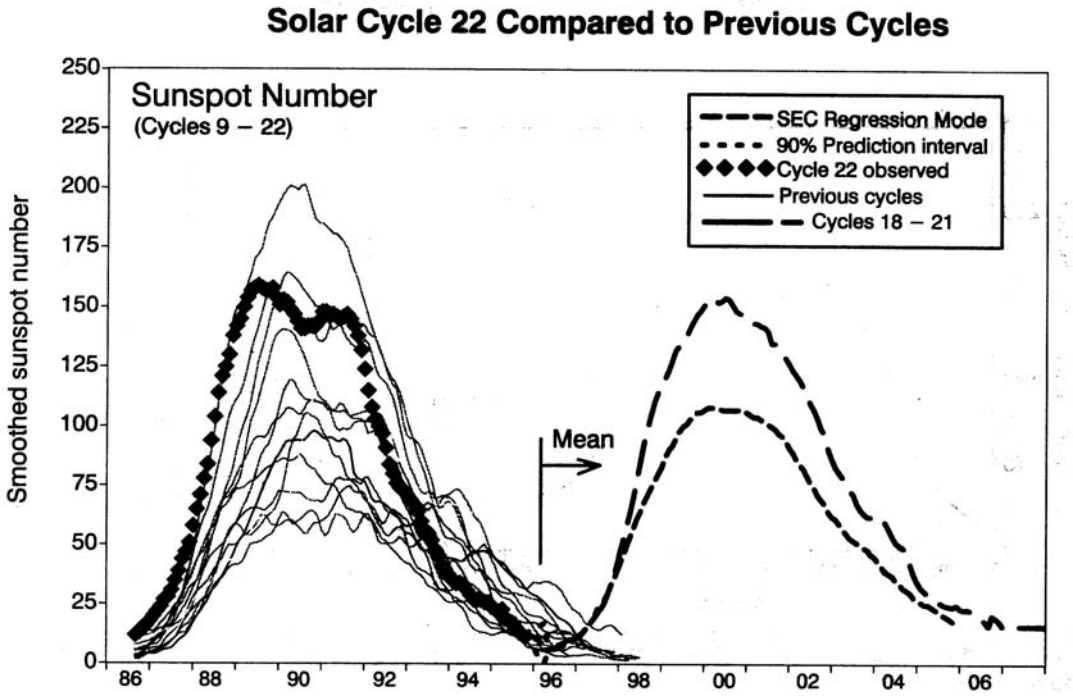

Figure 2.3-1 : Solar activity during cycle 9 to 22 (up to 1996) and cycle 23 forecast (up to 2007) [i]

2.4 GALACTIC COSMIC RAYS (GCR)

Cosmic rays consist of protons (85%), alpha particle (14%) and high-energy heavy ions (less than 1%). Due to their high ionisation capability, the heavy ions effects are predominant in devices, leading to failures (SEE).

The energy corresponding to the maximum flux is comprised between 10^2 and 10^3 MeV per nucleon, but the most energetic can reach 10^5 MeV per nucleon [ADAM1].

The geomagnetic field provides a protection against galactic cosmic rays: heavy ions interact with the geomagnetic field and are deflected at a certain depth in the magnetosphere depending on their energy [STAS]. As a consequence, the inclination and altitude of the orbit are determinant for the exposed devices [ADAM3] (the highly inclined mission and high altitude orbits are more exposed to the Galactic Cosmic Ray flux).

As shown on figure 2.4-1, the solar activity can lead to a GCR flux modulation by a factor 2 to 4 depending on the LET range. The solar wind is maximum during the Solmax period. Due to this interaction, the GCR flux is maximum during the Solmin period and is reduced during the Solmax period.

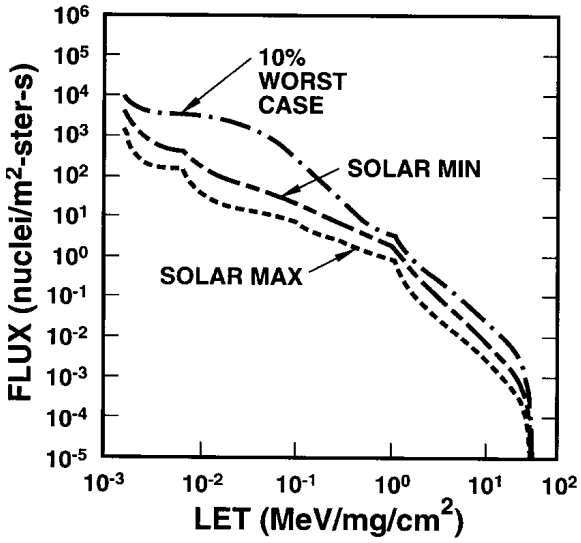

Figure 2.4-1 : Integral LET spectra at solar max, solar min and for Adam's 10% worst case

Satellite shielding has a minor influence to reduce the GCR, due to their high energy.

2.5 VAN ALLEN BELTS

Earth's radiation belts are composed of particles trapped by the earth magnetic field. There are two electrons belts (at low altitude around 3000 km, and at high altitude around 20000 km) and one protons belt (around 3000 km)[STAS][BOUR]. One particularity of Van Allen belts concerns the South Atlantic Anomaly (SAA): in the South Atlantic region, anomalous large electron and proton fluxes are encountered at low altitude (500 km)[STAS].

The highest energy is typically 400 MeV for protons and material shielding is relatively inefficient to reduce their flux (several hundreds millimetres of aluminum are necessary to screen these protons).

The highest energy is 7 MeV for electrons. Electrons do not cause SEE since their energy is too low. Material shielding can be used to stop the highest energetic electrons, but energy loss by electrons passing through matter is converted in high energetic photons (Bremsstrahlung) which are difficult to screen.

2.6 SUMMARY TABLE

The following table summarizes the different characteristics for each type of particle source.

	Solar flares	**Galactic cosmic rays (GCR)**	**Van Allen Belts**
Type of particles	• Protons • Alpha particles (5-10 %) • Heavy ions • electrons	• Protons (85%) • Alpha particles (14%) • Heavy ions (<1%)	• Electrons (2 belts at 3000 and 20000 km) • Protons (1 belt at 3000 km)
Energy level	low	10^2…10^3 MeV per nucleon 10^5 MeV for the most energetic	400 MeV max for protons 7 MeV max for electrons
Shielding efficiency	high	low	high for electrons low for protons
Exposed orbits	High altitude orbits: • Geosynchronous • Far missions	• Polar orbits • High altitude orbits	• Intermediary orbits around 3000 km • Low orbits above South Atlantic
Comments	*Cyclic activity* : solar flares occur mainly during 7 years (Solmax period) on a full solar cycle of 11 years	Attenuation by geomagnetic field Modulation by solar flares (GCR flux is maximum during Solmin period)	*South Atlantic Anomaly:* anomalous large electron and proton fluxes in this region Bremsstrahlung: secondary emission of energetic photons due to interaction between electrons and material in case of shielding

3 SINGLE EVENT EFFECTS (SEE) ON MICROELECTRONIC DEVICES

When an ion strikes a pn junction in silicon, a current peak is generated. As a consequence, Single Event Effects (SEE) occur. This section will briefly describe why and how SEU occur on storage elements and combinational logic, and will then give elements to understand Single Event Latchup (SEL).

3.1 CHARGE COLLECTION MECHANISM

Sophisticated numerical simulations have put in evidence the existence of a prompt charge collection mechanism induced by drift and funneling effect (duration ≈ 100 ps) and a delayed component corresponding to the charge diffusion toward the depletion region (duration ≈ 10 ns)[HSI][PICK]. The transient current intensity and shape following the ion strike depend on several parameters:

- the technology [DOD1][DOD2]
- the ion energy [KNU]
- the localisation [MUS1]

3.2 SINGLE EVENT UPSET IN STORAGE ELEMENT (SEU)

The charges generated by ions or protons and collected at sensitive nodes of the device can trigger a state change in a storage element. Several sensitive electrical nodes can exist, depending on the type of device [MUS2].

Tolerance to SEU can vary widely between technologies and several parameters can influence the device sensitivity. In particular, the reduction of the geometry size or the supply voltage tend to decrease the critical charge leading to a higher SEU sensitivity (see section 5.3.3).

3.3 SINGLE EVENT UPSET FOR COMBINATORIAL LOGIC (SEU)

When a particle hits a sensitive volume in combinational logic it can generate a voltage pulse that propagates through sensitized paths and causes a possible erroneous bit value to be loaded into one or several latches. Temporary glitches can be transmitted to a DFF clock input or to a D clock input: in both cases, the value of the data stored in the DFF may be corrupted. Analysis of such events in combinational logic is more complex than for storage element for four reasons [MAS]:

- the circuits in combinational logic do not consist in a repetition of storage elements as for RAMs
- the clock timing of the circuit to legitimate the signal relative to current pulse is crucial. The combinatorial signals can temporarily change if submitted to SEU, and so they must be sampled at the right time
- the paths an error signal can follow in its propagation depend on the active paths at one time
- multiple errors can be generated by a single hit requiring simulation of many error paths.

Hits in the combinational logic cause bit flips only if the voltage pulse arrives to a latch during the latching window, i.e. the time during which the voltage pulse can alter the bit value loaded into the latch. The

latching window constitutes only a small fraction of the total time and the probability is low that a voltage pulse originating from combinational logic becomes latched.

Full modelling of SEU through a combinational logic system is a very complex task [KAUL]. It consists to evaluate the impact of a pulse current in all sensitive nodes in the circuit combined with all input vectors. Such an exhaustive analysis is generally not possible on complex VLSI logic circuit and a simplified methodology is considered. Such methodology consists to evaluate the most vulnerable nodes in the circuit and a set of input vectors which facilitate the upset propagation through available signal paths.

These "combinatorial SEU" become of more importance with the new thinner technologies.

3.4 SINGLE EVENT LATCHUP EFFECT ON CMOS DEVICE (SEL)

Single Event Latchup (SEL) is defined as triggering a parasitic thyristor of PNPN structure existing in CMOS or bipolar devices by ion strikes. When it occurs, an important current flows and increases the local temperature of the die, having destructive effects.

The following figures show the parasitic transistor and the equivalent model of the parasitic thyristor.

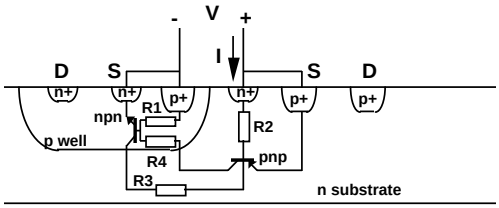

Figure 3.4–1 – cross section of a standard CMOS section showing the parasitic transistors

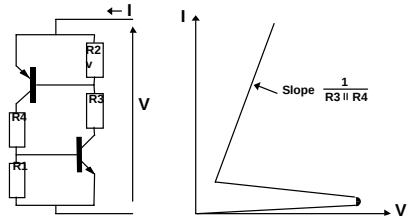

Figure 3.4–2 – equivalent model of the parasitic thyristor

SEL occurs only for CMOS technologies if the following conditions are met:

- Parasitic thyristors exist
- The parasitic transistors become biased into the forward active mode (heavy ions crossing the P^- well/N^-substrate junction)
- The parasitic transistor gain product (β_{npn} x β_{pnp}) exceeds a required minimum of 1 for regeneration to occur
- The bias supply is able to deliver a current greater than the holding current I_H, which is fixed by the physical characteristics of the structure.

Countermeasures against SEL are described in section 6.

4 SEU HARDENING AT FUNCTION DESIGN LEVEL

The use of radiation hardened technology based on isolating substrates (SOI or SOS), associated with the use of an hardened library drastically reduces the probability of SEU occurrence. This hardening strategy is called “fault avoidance” [SIEW82].

Another hardening method is used at function design level when fault avoidance is not possible. This hardening strategy is called “fault tolerance” [SIEW82]. It consists of adding specific structures, generally based on redundancy, to detect and correct the faults caused by a SEU. Different protection levels can be used, in order to design a functionally acceptable unit.

This section will detail the different possible strategies, and will then examine fault detection and fault masking techniques. Software methods will be briefly described and logic SEU hardening methods will then be explained by type of function (finite state machines, counters, registers, …). A specific paragraph will be dedicated to FPGAs, and different methods for simulating SEU effects will be exposed.

4.1 ASSESSMENT OF THE SEU RATE AND SEU TOLERANCE STRATEGIES

Before implementing SEU hardening, the designer shall answer these two questions:

- *What effects will have the SEU on the functions and on the system?*
  - the elementary functional blocks (memory, registers, synchronisation stages, counters, finite state machines …) shall be identified in the architecture.
  - the number of latching elements (DFFs, memory cells, latches …) shall be estimated for each elementary functional block since SEUs are located in storage devices.
  - by analysing all possible changes in the different latching elements of the blocks, the functional effects induced by SEU shall be identified
- *What is the probability of these effects?*
  - the occurrence probability of the identified SEU effects shall be estimated (the SEU sensitivity of the component being known)

From this point, the decision for the implementation of protections against each SEU effect will be made on the base of the required fault tolerance level.

If the effect is acceptable, no additional protection is required.

If the effect is not acceptable, fault detection or fault masking techniques shall be implemented:

- Fault detection can be applied when a disturbance in the mission is acceptable. It provides no tolerance to SEU effects and must be followed by countermeasures to suppress the SEU effects and recover a functional system (recovery actions).
- Fault masking is needed when no disturbance of the mission is acceptable, since it suppresses the effects of SEU. It can be associated or not with an SEU error reporting.

The following flowchart summarises the possible SEU tolerance strategies:

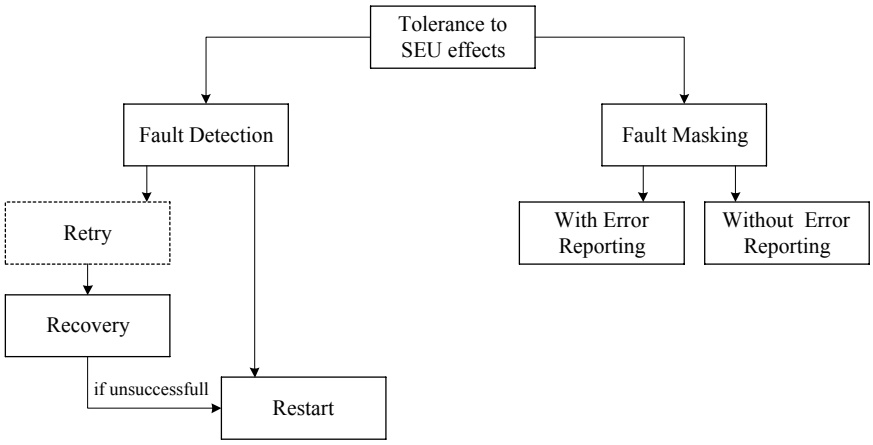

Figure 4.1–1 – Flowchart of SEU tolerance strategies

4.2 FAULT DETECTION

4.2.1 Redundancy

The redundancy mechanism is based on the structure depicted in figure 4.2.1-1. A nominal function F is associated with a second function F' which duplicates in a certain way the behaviour of F and a comparator function that detects differences in the response of F and F'. Special attention must be paid to the design of the comparison function to prevent failures generating either no error-detection or permanent or occasional false detection. It is necessary to resynchronise the comparator output, since delays may exist between signals coming from F and F'.

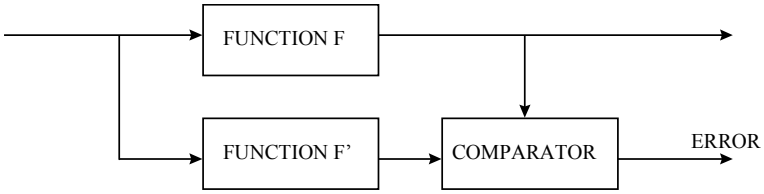

Figure 4.2.1-1 - Basic structure of redundancy mechanism

FAULT DETECTION TECHNIQUES - REDUNDANCY

- ***Advantages:***

  - safety

- ***Drawbacks:***

  - hardware overhead (area more than doubled because of the comparison function)

  - no error correction (fault detection mechanism)

  - testability

- ***Application field:***

  - not specific

4.2.2 Parity

In order to reduce the hardware overhead, an efficient implementation of this architecture uses error-detection codes. The simplest code is the parity, built with EXOR or EXNOR gates. In the case of SEU, an extra bit (F') is added to a register. A parity generator is used to generate the code and a parity checker to detect the error. This method is commonly used when designing ASICs. Each group of D flip-flops is associated with a parity bit, the results of elementary detection are ORed and used to generate a fault signal outside the ASIC.

FAULT DETECTION TECHNIQUES - PARITY

- ***Advantages:***

  - detection of single errors or odd number of errors

- ***Drawbacks:***

  - no detection for double error or even number of errors

  - hardware overhead (25 to 35 % of the total gate count for the ERC32 companion chip (MEC))

  - not applicable for big data structures (1 parity bit for 8 data bits seems to be a good choice)

- ***Application field:***

  - memories and registers

4.2.3 M-of-N code

An m-out-of-n code (m/n code) consists of n-bit code words in which m and only m bits are one. For example the 2/4 code has six possible code words : {1100, 1010, 1001, 0101, 0011, 0110}. These codes require generally more redundancy than other codes, which limits them to specific applications. M-out-of-n codes can also be used to detect errors, but they do not bring a significant improvement in this case.

FAULT DETECTION TECHNIQUES – M-OF-N CODE

- ***Advantages:***

  - nothing specifically

- ***Drawbacks:***

  - overhead (intermediary between redundancy and other codes)

- ***Application field:***

  - not often used

4.2.4 Arithmetic code

Arithmetic codes are codes that have the following property:

$$A(a*b) = A(a) * A(b)$$

where

a and b are operands

A(x) is the arithmetic code of x

* is an operation such as addition or multiplication.

Most common arithmetic codes are residue codes defined by R(N) = N mod m. These codes have a specific interest to design arithmetic units that are self checking. It limits hardware increase since the arithmetic unit is not duplicated as it can be seen in figure 4.2.4-1.

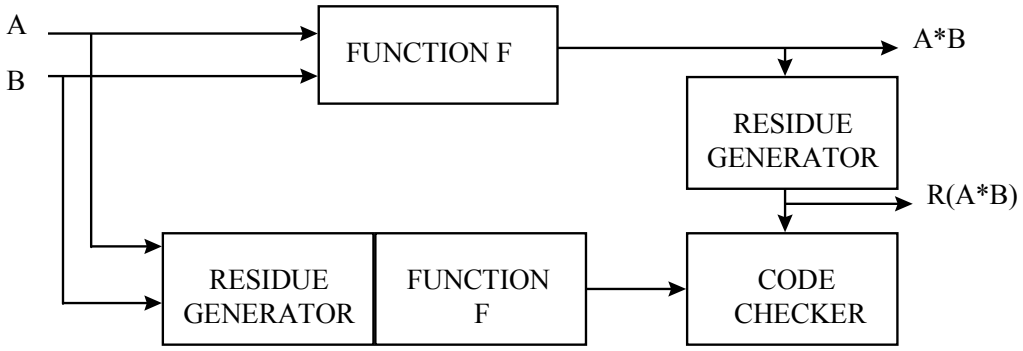

Figure 4.2.4-1 - An arithmetic function using an arithmetic code as error detection mechanism

FAULT DETECTION TECHNIQUES – ARITHMETIC CODE

- ***Advantages:***

  - self checking arithmetic unit

- ***Drawbacks:***

  - limited interest in SEU protection since the area overhead applies on the combinatorial part and not only on the registers (this method does not prevent to harden the registers following the arithmetic function F)

  - not applicable for logic function protection (and, or, shift …)

- ***Application field:***

  - adders and multipliers

4.3 FAULT MASKING

4.3.1 Triplication

The error correction mechanism can be done at the basic level of the ASIC or the FPGA by the designer himself. It is generally done by using three D flip-flops followed by a majority voter (TMR, Triple Modular Redundancy). It has the drawback to more than triple the gate count of a single D flip-flop, and to create testability problems since redundancy is introduced.

Some foundries proposed "ultimate" D flip-flop which is a D flip-flop almost insensitive to SEU [BESS93]. For example, TEMIC offer such a D flip-flop in its library. It is called HDFFR when it has an asynchronous reset. The SEU hardened HDFFR has a gate count of 16 compared to 6 for the ODFFR (which is the normal D flip-flop with reset in the same library). In any case, using this method on all the D flip-flops of an ASIC or an FPGA is quite expensive in terms of area and the power consumption is significantly increased. But the testability of HDFFR cells is possible with scan since HSFFR cell is provided that includes features for scan. The gate count of HSFFR cell is 20 gates.

It is sometimes possible by a detailed analysis to find a subset of D flip-flops to be hardened in order to allow safe processing of the control part of the chip while tolerating errors in the operative part. Hardened D flip-flops can be used for example to secure configuration registers that contain most often information used to define operational modes of the ASIC.

At system level, the same approach can be envisaged by using 3 identical functions, but this solution is generally not feasible within one single ASIC for area reasons.

FAULT MASKING TECHNIQUES – TRIPLICATION

- ***Advantages:***

  - error correction mechanism

- ***Drawbacks:***

  - area overhead (more than triple)

  - testing difficulty

- ***Application field:***

  - registers, fsm, counters, resynchronisation DFFs

4.3.2 Hamming codes

4.3.2.1 SEC or DED Hamming code

A k-bit information word is encoded into a n-bit code word, which is composed of the information word itself (k bits) and a check word of r check bits (r = n–k).

Assuming that a k bits data word shall be protected, the number r of check bits necessary to build a (n,k) Hamming code that detects and corrects single errors or detects double errors is the lowest integer r that corresponds to $k \leq 2^r - 1 - r$ (if $k = 2^r - 1 - r$, the efficiency of the code, i.e. the relation data bits/check bits is maximum. The following table shows the check bits count r for a given number of data bits k:

Data bit k	Check bits r	Code word bit n=k+r
$k = 1$	2	$n = 3$
$2 \leq k \leq 4$	3	$5 \leq n \leq 7$
$5 \leq k \leq 11$	4	$9 \leq n \leq 15$
$12 \leq k \leq 26$	5	$17 \leq n \leq 31$
$27 \leq k \leq 57$	6	$33 \leq n \leq 63$
$58 \leq k \leq 120$	7	$65 \leq n \leq 127$
$121 \leq k \leq 247$	8	$129 \leq n \leq 255$

Table 4.3.2.1-1 - SEC or DED Hamming code : check bits count versus data bit count

The methodology for building a SEC or DED (n,k) Hamming code is detailed in Appendix (in section 8.1.1).

There are some tricks for the physical implementation, in order to simplify the logic synthesis, to reduce the critical path and the gate count.

This method of error detection and correction allows double error detection. If a double error occurs, the syndrome will be different from all zeroes, but there are no means to distinguish between single and double errors and the circuit will correct this error as if it was a single error.

FAULT MASKING TECHNIQUES – SEC OR DED HAMMING CODE

- ***Advantages:***

  - single error correction, double error detection mechanism

- ***Drawbacks:***

  - no means to distinguish between single and double errors (syndrome ≠ 0)

  - area overhead

  - testing difficulty

- ***Application field:***

  - registers, memory

4.3.2.2 SEC and DED modified Hamming code

In the SEC or DED Hamming code, there is no way to distinguish between single and double errors. The SEC and DED modified Hamming code allows the distinction between the single errors and the double errors, by using an extra check bit.

The following table shows the check bits count r for a given number of data bits k.

Data bit k	Check bits r	Code word bit n=k+r
$k = 1$	3	$n = 4$
$2 \le k \le 4$	4	$6 \le n \le 8$
$5 \le k \le 11$	5	$10 \le n \le 16$
$12 \le k \le 26$	6	$18 \le n \le 32$
$27 \le k \le 57$	7	$34 \le n \le 64$
$58 \le k \le 120$	8	$66 \le n \le 128$
$121 \le k \le 247$	9	$130 \le n \le 256$
$248 \le k \le 502$	10	$258 \le n \le 512$
$503 \le k \le 1013$	11	$514 \le n \le 1024$

Table 4.3.2.2-1 - SEC and DED modified Hamming code: check bits count versus data bit count

The methodology for building a SEC and DED (n,k) modified Hamming code is almost the same as for the SEC or DED Hamming code, it is described in Appendix (section 8.1.2).

It is possible to output 2 error signals, one for the single corrected errors and the other for the double uncorrected errors.

FAULT MASKING TECHNIQUES – MODIFIED SEC AND DED HAMMING CODE

- ***Advantages:***
  - **single error correction, double error detection mechanism**
  - **distinction between single (corrected) and double (uncorrected) errors**
- ***Drawbacks:***
  - **area overhead**
  - **testing difficulty**
- ***Application field:***
  - **registers, memory**

4.3.3 BCH code

Binary BCH codes belong to the family of cyclic codes, BCH stands for Bose, Chaudhuri and Hocquenghem who discovered this code in 1959-1960. The theory of cyclic codes is complex and is out of the scope of this manual: detailed explanation can be found in [LIN83]. As for the Hamming codes, a k-bit information word is encoded into a n-bit code word that contains k information bits and n-k check bits.

It can be shown that for any positive integer m ($m \geq 3$) and t ($t < 2^{m-1}$), a binary BCH code exists with the following parameters:

- Block length (code word) $n = 2^m - 1$
- Check bits count $n-k \leq m.t$
- Minimum distance $d_{min} \geq 2t + 1$

This code is capable to correct t or less errors in a block of $n = 2^m - 1$ digits.

Table 4.3.3-1 gives the size in bits of the information (k), of the number of check bits (n-k), and of the codeword (n) for BCH codes able to correct 1, 2 or 3 errors. It must be noticed that the values of k are discrete. For example, for BCH codes able to correct one error, there are no values of k between 11 and 26. Thus correcting a 16-bit word leads to use a code for 26 bits, which is shortened.

Correction of one error (t=1)			**Correction of two errors (t=2)**			**Correction of three errors (t=3)**		
k (data)	**n-k (checkbits)**	**n (codeword)**	**k (data)**	**n-k (checkbits)**	**n (codeword)**	**k (data)**	**n-k (checkbits)**	**n (codeword)**
4	3	7						
11	4	15	7	8	15			
26	5	31	21	10	31	16	15	31
57	6	63	51	12	63	45	18	63
120	7	127	113	14	127	106	21	127
247	8	255	239	16	255	231	24	255
502	9	511	493	18	511	484	27	511

Table 4.3.3-1 - Relation between the code and check bit size for BCH codes

FAULT MASKING TECHNIQUES – BCH CODE

- ***Advantages:***
  - adapted technique for long blocks
- ***Drawbacks:***
  - number k of data bits is discrete (4, 11, 26 …)
  - area overhead
  - testing difficulty
- ***Application field:***
  - transmission

4.3.4 Reed-Solomon codes

Reed-Solomon codes are a special class of BCH codes. Instead of using binary codes, non-binary codes are considered. Symbols are taken from the Gallois field GF(q), and we will consider here only the case when $q = 2^m$. A t-error-correcting Reed-Solomon code with symbols from $GF(2^m)$ has the following characteristics :

- Block Length : $n = 2^m - 1$
- Number of parity-check digits : $n - k = 2t$
- Minimum distance : $d_{min} = 2t + 1$

Physical implementations of Reed-Solomon encoders and decoders are described in Appendix (section 8.2).

FAULT MASKING TECHNIQUES – REED-SOLOMON CODE

- ***Advantages:***
  - adapted technique for long blocks
- ***Drawbacks:***
  - area overhead
  - testing difficulty
- ***Application field:***
  - transmission

4.4 LOGIC SEU HARDENING METHODS

There are 2 criteria of choice for SEU hardening strategy: the type of function to harden (counter, register, finite state machine …) and the way of hardening this function (triplication, Hamming code ….).

4.4.1 Protection of Control logic

The different architectures presented below result from a selection taking into account the feasibility of implementation using VHDL. In particular, it is preferable that the protection part can be separated from the functional part, for better design validation, understanding, verification, modification, troubleshooting and documentation.

For each category of control logic (except counters), two classes of protection architectures will be presented:

- Fault masking structures provide masking of an SEU effect occurring in one element of the memory section
- Fault detection structure is a minimum hardware solution, used when on-fly correction is not needed.

4.4.1.1 Protection of Finite State Machines

A survey of fault tolerant architectures for finite state machines can be found in [NIRA96].

In the further descriptions the following terminology is used:

- the state variables are stored in the state flip-flops (*memory section* of the FSM), which provide the *present state* of the machine;
- the combinatorial logic that provides the *next-state* is the *Input Forming Logic (IFL)*.
- the combination logic that provides the outputs is the *Output Forming Logic* (*OFL*)

The fault masking structures are:

- *Triple Modular Redundancy (TMR):* all state flip-flops are triplicated; a majority voting circuit is implemented along with each set of three flip-flops, in order to determine the effective present state. Each set of three flip-flops receives the same input signals from the IFL section.

  The implementation of the TMR architecture for SEU effects mitigation differs from the full TMR principle used for failure tolerant architectures by the fact that triplication is implemented only on the flip-flops, and not also in the combinatorial logic circuit.

  This design has the following features:

  - an upset in one of the three flip-flops does not affect the present state (the upset is masked),
  - the upset flip-flop recovers its correct state at the next system clock event

- *Duplex architecture*: two identical memory sections are implemented, receiving the same signals from the IFL. The IFL issues the N + 1 bits, which are the N bits of the next state code, and an associated parity bit. Each memory section includes an additional flip-flop, which stores the parity bit. A parity control is performed at the output of the memory sections, and a selection logic selects the effective present state, depending on the value of error signals issued from the parity controllers. In the case where both parity control circuits signal an error, the FSM can be forced into a safe state.

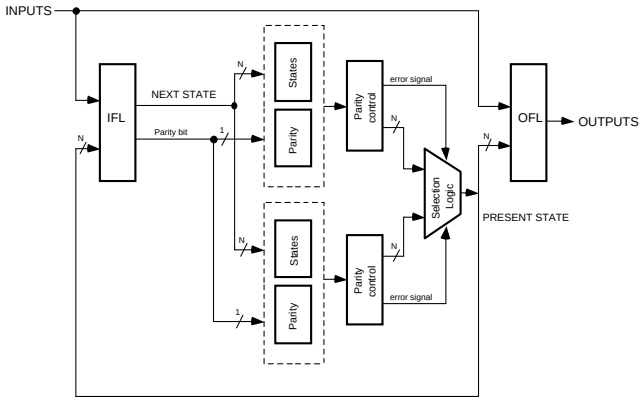

Figure 4.4.1.1-1 – FSM duplex architecture [NIRA96]

- *Error Correction Architecture*: the states are encoded with a Hamming code (Hamming distance 3) in order to provide correction of single errors in the state flip-flops; the N + R next state bits are provided by the IFL. The checkbits are stored along with the state bits. At the output of the

memory section the bits are analysed by an error detection and correction circuit (only the state bits are corrected, not the checkbits); the output of this circuit issues the correct present state.

Figure 4.4.1.1-2 shows a block diagram of this architecture.

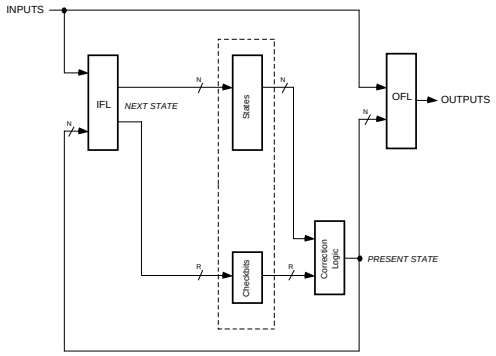

Figure 4.4.1.1-2 – FSM Error correction architecture [NIRA96]

The techniques are all based on a Hamming distance of 2 between the states or between the most critical states; when such a solution is chosen, the effects of the SEU on the system must be carefully analysed. The fault detection techniques are:

- appending a parity bit to the state codes, with next state equal to an idle state in case of error detection;
- definition of a special encoding of the states in order to have a minimum Hamming distance of 2 between any couple of critical states. The encoding is studied in order to carefully control the effect of the SEU at each state;
- use of one flip-flop per state ("one-hot-encoding"). This trivial code (which is not a linear code) uses more flip-flops than a more compact code, but it has the following advantages: a simple error leads to all bits at zero or two bits at one (detectable by simple xoring), and the decoding of the states is easier for the generation of the output signals.

The following table shows the results obtained for the different FSM protection solutions [NIRA96]:

	Fault masking	**Fault detection**	**Area overhead**	**Delay overhead**
TMR	X		45 %	30 %
Duplex	X		90 %	130 %
Error correction	X		45 %	45 %
Parity		X	Negligible	Not applicable
Special encoding		X	Code dependent	Not applicable
One-hot encoding		X	?	Not applicable

For finite state machines, Triple Modular Redundancy is the recommended fault masking technique for SEU protection, but error correction can also be used (speed performance is degraded). If fault masking is not required, one-hot encoding can be used.

4.4.1.2 Protection of counters

Due to the usual purpose of the counter functions in the logic design, fault detection without masking is not commonly required. The fault masking elementary techniques used for SEU counter protection are:

- *Triple Modular Redundancy (TMR):* this implementation is the simplest and the most efficient way in order to mask transient faults due to SEU. It consists in replacing each flip-flop by a "TMR cell" (i.e. 3 flip-flops and a majority voting element) in the existing architecture of the counter. Figure 4.4.1.2-1 shows a typical counter architecture.

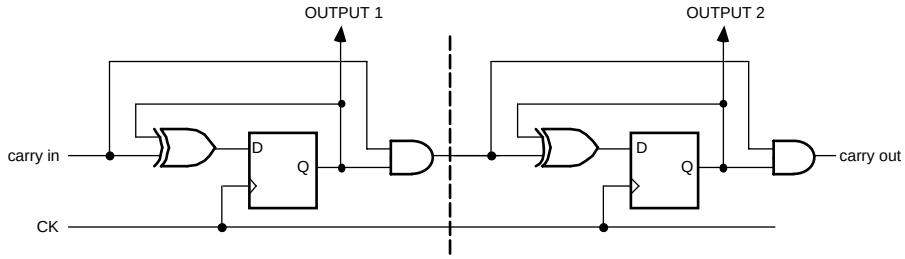

Figure 4.4.1.2-1 – Typical counter architecture

- *Coding technique*: this method can be applied in order to implement SEU tolerant counters. A possible implementation of such a coding technique is based on Reed-Muller codes [REED70] and is described in Appendix (Section 8.3).

The test vehicle developed in the frame of the "Circumventing Radiation Effects by Logic Design" R&D (WP400) incorporates three different 16-bits counter designs: unprotected counter, code protected counter (Reed-Muller) and TMR protected counter. The comparison between the Reed-Muller implementation and the TMR protected implementation shows that the TMR protection design is more efficient for counter protection:

- in terms of area : for a 16-bit counter, the TMR protected counter overhead is 180 %, the Reed-Muller code protected counter overhead is 280 %
- in terms of timing, the Reed-Muller code protected counter has a lower frequency performance than the TMR counter, due to its combinatorial logic.

For counters, the Triple Modular Redundancy is the most efficient technique (in terms of area and speed) among the fault masking techniques. Fault detection without masking is not commonly required

4.4.1.3 Protection of data storage registers

The fault masking structures are the same as for the FSM:

- *TMR structures*: each D flip-flop is triplicated and a majority voter is added for each bit
- *Correction codes* such as Hamming codes

The fault detection structures are generally based on a *parity* generator associated to a n-bit register, as depicted in the figure below.

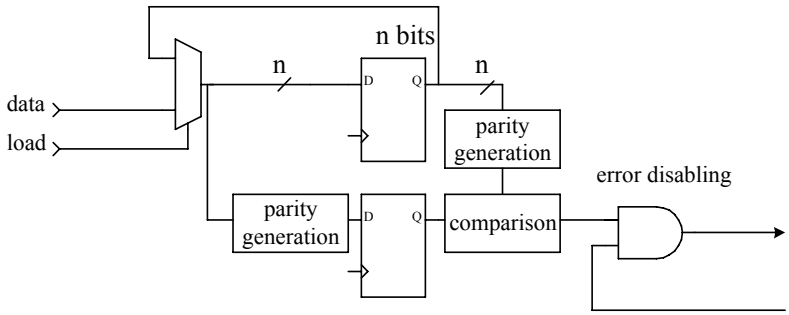

Figure 4.4.1.3-1 – Error detection in registers using parity

It can be shown that (see WP210 report for further details):

- the area overhead brought by SEU detection applied to all the D flip-flops of an ASIC can be estimated at:

$$overhead\ in\ \% = \frac{(number\ of\ D flip-flop) * 2 * Size(xor)}{(number\ of\ D\ flip-flop) * 2 * 10} * 100 = \quad 10 * Size(xor)$$

  (assuming that the estimated number of gates for an ASIC is: (number of DFFs)*2*10)

- in terms of speed degradation, a critical path is introduced by the parity generation. For a n-bit register, the timing overhead can be estimated at:

  $timing\ extra\ path \approx log_2(n) * propagation\ time(xor2)$

In order to prevent from false detection, the error detection should be inhibited as long as the register is not used (i.e. a write is necessary to refresh it). In this case care has to be taken in order to ensure that a stored error does not disturb the system performance. In order to mask the parity bits, a control signal shall be generated with a limited number of gates, and without adding extra D flip-flops. This signal can be generated either at register level or at function level (creation of an idle signal which is high when the function is high).

For data storage registers, the most efficient protection is the Triple Modular Redundancy. If fault masking is not required by the system, redundancy techniques can be used.

Other techniques, called "protection of registers by protocol" can be successfully applied to critical registers loaded from a PROM or an SEU hardened RAM. They consist in:

- Systematically loading the register to protect before each read,
- Loading the register to protect before the beginning of a new processing
- Loading the register to protect if an SEU detection mechanism reports an error

Registers may also contain data that are used to control processing (number of words, pointer in RAMs…). It is crucial to ensure that the data stored in these registers are automatically recovered between two processing operations. This can be done by storing these data in protected registers. A special attention will be paid to data such as "frame length" or "pointers to the next data unit in the buffer" in packet management. These data must be stored in SEU protected registers instead of sensitive memories.

4.4.1.4 Protection of resynchronisation functions

For such function, the best SEU protection technique that brings SEU protection as well as metastability protection is based on TMR:

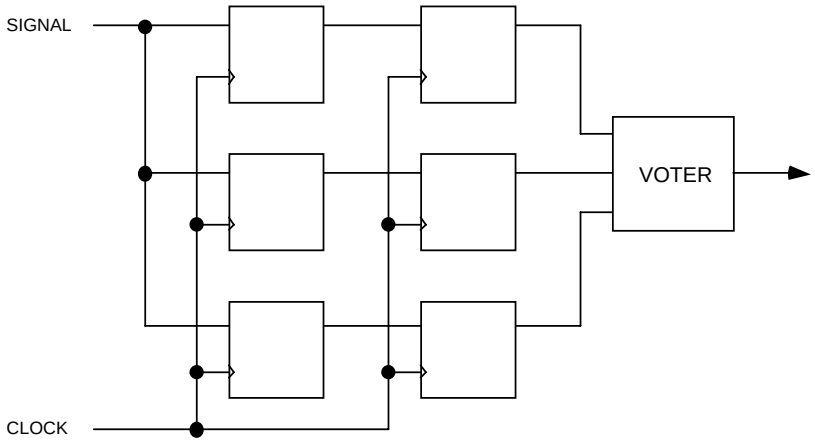

Figure 4.4.1.4-1 – TMR protection for resynchronisation functions

Other solutions, such as EDAC protection or parity detection are not suitable, because transients or wrong errors may occur (see WP210 report for more details).

For resynchronisation flip-flops, the only possible protection is the Triple Modular Redundancy. Other solutions, such as EDAC protections or parity detections are not suitable in this case.

4.4.2 Protection of RAM memory

Memory functions are of particular concern for SEU circumvention since they use a very large amount of sequential logic elements. The proposed solutions will be applicable either to ASIC integrated RAM areas, or for external memory.

Due to the characteristics of memory, heavy ions can lead to single-bit errors, multiple-bit errors or transient events in the control logic, the most likely being the single-bit errors. Protection against multiple-bit errors should not be necessary in most applications, except in some particular cases.

The protection strategy will be chosen between fault detection and fault masking:

- Fault masking is generally required since memory read and write cycles are often involved in program execution and real-time data processing: a good level of fault tolerance is thus achieved.
- Fault detection without any fault masking could be implemented in not critical applications.

The fault detection structures are:

- *Parity on the data words*: this technique is dedicated to single-bit errors, it is the simplest one but the less efficient (double errors or even number of errors are not detected). In case of error detection, the system shall interrupt the operation and carry out recovery actions. The parity bit shall be associated to a memory area (for example 1 parity bit every 8 data bits). It shall be generated upon write accesses and checked upon read accesses.
- *SEC-DED-SBD codes (Single Byte Detection or BED, Byte Error Detection):* these codes apply when the useful data is composed of n bytes of b bits each (usually b = 4, 8,…); they are able to detect multiple-bit errors. They have the same capabilities as SEC-DED codes, and can detect all error configurations within single bytes. Information about these codes can be found in [CHEN84], [JOHA96], [FUJI95].

The fault masking structures are:

- *Single bit error masking*: Masking of single-bit errors is classically achieved with the use of an EDAC, based on Hamming Single Error Correction codes (see section 4.3.2). Data bits and check bits are stored in the memory area. The data is not corrected in the memory, but if one bit of a stored word is incorrect (inverted), due to a SEU, it is acquired without error by the microprocessor. The detection of double errors can generally also be made by the EDAC: if double errors are detected, recovery actions shall be initiated at higher level. The most commonly used codes are of the modified Hamming type (SEC and DED), and are chosen in order to minimise the number of XOR gates (parity check matrix with the fewest number of 1's). Two EDAC implementations are possible:
    - the "serial configuration" ("flowthrough" or "correct always"): the EDAC is located between the memory and the microprocessor; when the memory is accessed in reading, the memory issues the data to the EDAC circuit, and the EDAC circuit issues a corrected data to the microprocessor.

- the "parallel configuration" ("bus watch or check only"): In the parallel configuration the EDAC checks the read data in parallel on the bus; the data issued from the memory is directly read by the microprocessor. In case of one error, the EDAC has to lengthen the access cycle and to disable the memory driver in order to correct the data.

- *Multiple bit error masking*: memory chips insensitive to multiple bit errors shall be used in association with a classical EDAC implementation (most of SRAM and DRAM components have a low sensitivity with respect to multiple-bit errors, due to their internal topology).

- *Use of one-bit-per-memory chips*: one-bit-per-memory chips shall be used along with a Single Error Correction EDAC. However, this architecture requires a large number of memory chips: this is not very easily compatible with the present trend of increasing storage capacity and integration density in the processing functions

- *Use of Error Correcting Codes*: for the correction of more than 1 bit, more complicated codes than the classical SEC-DED Hamming codes have to be implemented [CHEN84].

  In the case where the memory is implemented in a b-bits per chip organisation, SBC-DBD (single byte error correction, double byte error detection) codes are of interest. DEC-TED (double-bit error detection, triple-bit error detection) codes, derived from the BCH code theory, can also be used. For 2^m data bits, the number of check bits required for DEC-TED BCH codes is 2m+3 [CHEN84].

  The following table gives the number of checkbits required for some SBC-DBD codes [CHEN84]:

Bits per byte (b)	**Data bits per Error Correcting Code word**			
	16	**32**	**64**	**128**
2	8	10	10	12
3	9	12	12	12
4	12	12	14	16
$b \geq 5$	3b	3b	3b	3b

- *Scrubbing*: since some data may stay a long time in the memory, it is necessary to periodically correct all single errors to protect it against accumulation of SEU errors (which would lead to multiple errors): this is done by mean of the *scrubbing technique*. This mechanism will ensure that all single errors are detected by the EDAC and corrected by the re-writing. Except in the case of very high refresh rate of the data, EDAC protection and scrubbing have to be used in conjunction.

  It has been shown [WHIT82] that the mean number of SEU hits in a memory system before a second error occurs in one word is given with a good approximation by: $h=1+\sqrt{2N}$, where N is the number of words in the memory system.

  The period of the scrubbing mechanism has to be calculated in such a way that the probability of two errors or more in a word (assuming that each SEU leads to a single bit error within a given word) is less than a specified value. It has been shown [WHIT82] that the mean time before an uncorrectable error (i.e. a two-bit error or more in a word) occurs in such a memory system is given with a good approximation by:

$$MTTF = \frac{t_i}{1 - \left[(1+\mu)\cdot e^{-\mu}\right]^N}$$ where:

MTTF : mean duration in seconds before an uncorrectable error occurs in memory system
t_i : period of the memory scrubbing operation in seconds
μ : number of SEU per memory word during t_i
N : total number of words in the memory system

if μ is small the MTTF can be expressed as :

$$MTTF\,(year) = \frac{473{,}42 \times N}{t_i \times h_r^2}$$ where:

MTTF : mean duration in years before an uncorrectable error occurs in memory system
N : total number of words in the memory system
t_i : period of the memory scrubbing operation in seconds
h_r : SEU rate of the memory system expressed in SEU per day

The following example illustrates the efficiency of the scrubbing technique: a 40 960 word memory system, protected by a SECDED EDAC (16 data bits + 6 check bits), with a SEU rate of 10 hits per day for the whole memory has a MTTF of 28.7 days without scrubbing. If scrubbing is applied to this memory system with a scrubbing period of 8 s, the MTTF will be 24 239 years (example issued from [WHIT82]).

The following circumvention techniques can be applied to ASIC integrated RAMs as well as external memory.

Fault masking is generally required to achieve a good level of fault tolerance, but fault detection without any fault masking could be implemented in not critical applications.

Protection against single-bit errors is generally sufficient, but in some cases it can be necessary to implement protection against multiple bit errors.

The different circumvention techniques used for memory protection are:

- ***Protection against single-bit errors*:**

  ***Fault detection*: parity bit along with the data bits**
  ***Fault masking*: EDAC (SEC, SEC+DED) along with scrubbing**

- ***Protection against multiple-bit errors*:**

  ***Fault detection*:**

  -> error detection codes (SEC+DED…)

  ***Fault masking*:**

  -> use of 1-bit-per-chip memories,
  -> for n-bit-per-chip architectures, use of insensitive to multiple-bit error RAM devices, with SEC+DED EDAC and scrubbing,
  -> Error Correcting Codes (SBC-DBD, DEC-TED)

4.4.3 Protection of Data processing logic

A data processing function processes an input data stream and issues an output data stream; it includes the following blocks:

- the input circuit (decoder). This function has two main purposes: decoding of information in the input data (e.g. synchronisation pattern), and checking (to some extent) the validity of the data. This validity checking is a part of the means which are implemented along the path of the information in order to harden it against an aggressive environment (noise...);
- the output circuit (encoder), which provides the processed data to the upper layer. This function performs the reverse operation of the input circuit: bringing to the output data the useful protection features (encoding) before it is sent in its further path;
- a set of combinatorial functions separated by registers (pipeline structure)

All the blocks involved in the processing function contain registers, used within elementary functions such as FSM, synchronisation registers, storage registers... Therefore all the circumvention methods previously discussed can be applied to them.

In some cases SEU-induced faults in data processing might not be as much critical as in control logic functions (for example an incorrect pixel in an image can be acceptable). This is to be taken into account in the architecture analysis which is done in order to define the SEU tolerance needs.

In order to protect the data stream within the processing function a way could be analysed in greater detail: the idea is that when entering the processing function the data is encoded into an error detection code, and the encoded data goes through all the stages of the processing; at each stage a validity checking is done and in case of error detection a flag is asserted toward the higher layers. This supposes the identification of a code which features suitable properties with respect to the different processing's (e.g. stability, linearity).

Moreover it must be noted that upstream of the input interface and downstream of the output interface the transmission of the data flow can be protected by codes (Cyclic Redundant Checker (CRC), BCH, Reed-Solomon, convolutional code...).

For data processing logic, the SEU tolerance might not be as much critical as in control logic functions. Circumvention methods previously discussed in section 4.4.1 can thus be applied.

An alternative way of protecting data processing logic from SEU is to use an error detection code for the input data. At each stage a validity checking is done and a flag is asserted in case of error (this code shall be compatible with the different data processing's). Cyclic Redundant Checker (CRC), BCH, Reed-Solomon, convolutional code can already be used to protect the data flow transmission

4.4.4 Protection of Testability functions

Testability functions implemented in components are dedicated to test on ground and shall not be exercised in flight normal operation. Therefore the problem raised by the testability functions lies in the fact that a component must not be disturbed in its nominal operation when the testability functions are reached by SEU.

Test mode configuration is often stored within registers in the ASIC. In order to secure flight operation against the activation of the test mode by an SEU, the following countermeasures can be implemented:

- if pins are available, it is recommended to control the test mode by a TEST pin. This TEST pin will force the register used by the test in an inactive mode by reset.
- if a pin is not available for test mode, hardened D flip-flop shall be used to secure the test mode.

If JTAG (Join Test Action Group) is used within the ASIC, it is necessary to ensure that it will not be activated in flight. It is highly recommended to use a separated TRST (Test Reset) pin which will deactivate the JTAG in flight. The designer should verify that:

- The registers of the TAP controller state are asynchronously reset by TRST in the idle state,
- The ClockDR, ShiftDR, UpdateDR, ClockIR, ShiftIR, UpdateIR signals are asynchronously forced inactive by the TRST signal
- The Instruction Register is asynchronously reset by TRST
- The boundary scan cells are forced in Non Test operation mode. For observe-only cells no specific cares are required, for other cells it is necessary to ensure that the "Mode" signal that control the multiplexer is asynchronously forced in an inactive state by the TRST pin (please refer to figure 10-30 of the IEEE 1149.1 standard [JTAG90]).

To avoid a perturbation on the nominal operation when testability functions are reached by SEU, external TEST pins shall be used for controlling the test mode (if pins are available), or hardened DFFs shall be used in the test registers to secure the test mode.

If JTAG is used, a separated TRST (Test Reset) pin shall be used for asynchronous reset or deactivation of all internal signals and registers.

4.4.5 Protection of Combinatorial functions

SEU in combinatorial functions can impact the state of DFFs, since glitches propagate through the logic for arriving at the DFF inputs. If the data input is reached, a wrong state may be stored only if transient arrives nearby the sensitive edge of the clock. If the clock or the asynchronous set or reset inputs are reached, a wrong state may be memorised whatever the clock value is. Moreover, clock and reset signals are most often structured as trees, increasing the probability of an SEU to reach many D flip-flops.

The limitations of the effects of SEU in combinatorial logic by only logic design are quite limited. Nevertheless some methods exist that will be detailed hereafter:

- *Glitch filtering*: input glitches are filtered by the following circuit, proposed by ESTEC; library cells from a standard digital library are used, the delays being made with chains of inverters.

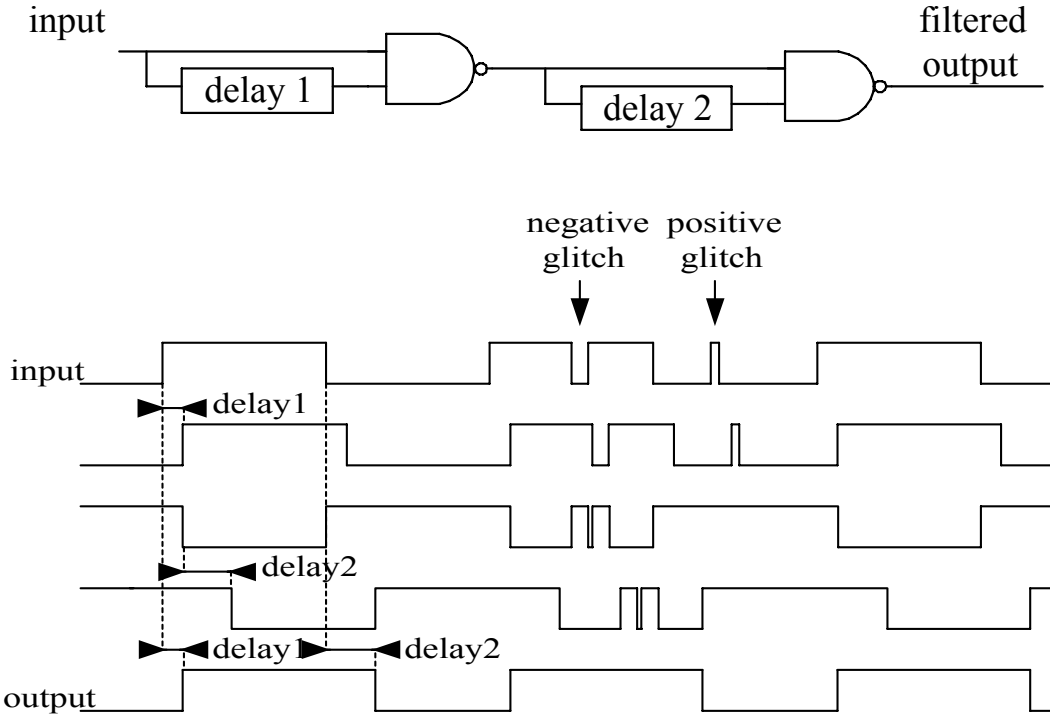

Figure 4.4.5-1 – Glitch filter by using library cells

The left part of the schematic is used to filter the positive glitches (delay 1), and the right part of the schematic is used to filter the negative ones (delay 2).

The following requirements exist:

- Delay1 and Delay2 shall be greater than the length of the SEU pulse with a margin
- |Delay1 – Delay2| shall be greater than the length of the SEU pulse, i.e. Delay1 and Delay2 shall be different
- The effective area of the filter shall be minimised

- The filter shall be located near the D flip-flop

The test vehicle developed in the frame of the "Circumventing Radiation Effects by Logic Design" R&D (WP400) incorporates this structure, the filtered output of the glitch filter being connected to the D input of a DFF. No SEUs have been seen during heavy ions irradiation.

- *Cell selection*: the behaviour of each combinatorial cell to SEU can be simulated by the foundry; a special attention will be turned to its ability to produce glitches of sufficient duration and to absorb glitches. This work leads to give recommendations concerning the use of cells to minimise SEU effects in combinatorial logic. Specific rules will be adapted for each technology, but the following recommendations generally apply for CMOS technologies in the range of 0.5 µm to 1 µm:
  - The clock pad and its associated buffer have to be carefully selected.
  - Cells with high drive strength should be selected. By using high drive strength elements some SEU glitches can be absorbed in the cell itself. But when selecting cells, even of high drive, a special attention must be paid to the internal logic of the cell that may contain elements with small drive leading to a higher sensitivity than expected. In fact, a characterisation by simulation is the only way to classify the cells.
  - 5V logic must be preferred to 3V logic
  - The use of inverting cells should be preferred to the use of non-inverting cells that generally contains small drive elements.
  - The use of cells with a small number of inputs (2 or 3) should be preferred to the use of cells with 4 input pins or more.
  - D flip-flop cells having a lower sensitivity to glitches on their input should be selected (if they exist).

The limitations of SEU effects in combinatorial logic by only logic design are limited. However, two methods are used for reducing the SEU effects in combinatorial logic:

- **The use of glitch filtering at the inputs of DFFs**
- **A cell selection in the library (only the less sensitive cells shall be used: cells with high drive strength, inverting cells, cells with a small number of outputs …) and the use of 5V logic instead of 3V logic if possible.**

4.5 SOFTWARE METHODS

The software based methods apply to functions containing a microprocessor that runs code. They can be used in two ways to mitigate SEU effects:

- By implementing specific error correction methods in software:
  - CRC can be calculated in software to detect an error over a data entity or a memory area,
  - cyclic codes (such as Reed-Solomon codes) can be software implemented to detect and correct errors in small and medium size memory areas, which are seldom modified [BOW].

  The algorithms used shall be as fast as possible (use of a lookup table for Reed-Solomon coding algorithm, for example).
- By allowing retry operation if an error is detected: this method consists in using software capabilities to recover from an SEU error without resetting the microprocessor. Backward error recovery techniques must be implemented: execution is rolled back to a point before the occurrence of the error [SIEW82]. Two methods are applicable to correct SEU errors:
  - *Retry techniques* are the fastest form of error recovery. Immediately after the SEU error is detected, the instruction is retried. Retry techniques require hardware for fast SEU error detection, knowledge of the error location and of the state of the microprocessor before the error. If the error has corrupted information that cannot be corrected, the retry operation will be unsuccessful. Moreover the error able to be recovered by a retry operation has to be filtered depending on the considered DFF and on the functional state of its environment.
  - *Checkpointing* is a technique that is most often implemented in software and requires little or no extra hardware. In checkpointing a subset of the system state is saved at specific points during the process execution. After an error detection, a rollback is performed which consists in resetting the microprocessor to the state stored at the latest checkpoint. A loss in computation time occurs, but it must be noticed that the data received by the microprocessor between the last checkpoint and the rollback are also lost. The use of checkpoints creates a decrease in performance of the microprocessor, and software techniques must be used to optimise the overhead.

  The implementation of software based techniques to mitigate SEU can be based on the following principles:
  - the error detection mechanism of the flow control can be improved for example by using signature-monitor techniques. A signature is calculated by the hardware at each instruction execution. This signature is also computed in advance by the program compiler. Hardware and software signatures are compared in the microprocessor at specific points, and when different an error trap is signalled. Such a method requires modifying the compiler, which is not an easy task in space programs. Moreover, this technique is very difficult to apply, due to the impact and handling of loops, interrupts, exceptions, …
  - the error location must be analysed with accuracy to determine the seriousness of the fault and the way to restart the program (retry, roll back or reset). This is done by sharing the internal

registers in several areas from a functional point of view, and by controlling the latency time between the SEU error and the detection activation. The following classification is a simplification of the one used in the ERC32 Integer Unit [GAIS96]:

⇒ Restartable and precise error: this error can be removed by retrying the failing instruction. Recovery is performed by returning from the trap routine, which will resume execution at the location of the failing instruction. Errors of this type originate from parity errors in temporary registers. Retrying the instruction reload these registers and suppress the effects of the SEU.

⇒ Non-restartable and precise error: the failing instruction is known but will not suppress the SEU effects if retried. Removing the error requires restarting the current task.

⇒ Other errors : in fact all the detected errors which do not belong to the 2 previous classes require a reset of the component since they are not software correctable.

As it can be seen, circumventing of SEU errors by software methods requires a highly programmable component, which is not often the case with an ASIC.

The software based methods for SEU hardening apply to functions containing a microprocessor that runs code.

A first method consists in implementing specific error correction mechanisms in software (Reed-Solomon, BCH), but this limits the performance of the microprocessor.

A second method consists in allowing retry operation to recover from an SEU error without resetting the microprocessor. For this, two backward error recovery techniques can be implemented:

- **the retry technique: requires extra-hardware but is the fastest recovery technique,**
- **the checkpointing technique: is most often implemented in software and requires little or no extra hardware, but decreases the performance of the microprocessor**

4.6 PARTICULAR CASE OF FPGAS

FPGAs are nowadays often used for space applications for replacing glue logic. They can be classified according to the type of programming element they use:

- Antifuse (ACTEL)
- SRAM (XILINX, ATMEL)
- Floating gate (EPROM or EEPROM)

All the FPGAs are sensitive to SEU. But this sensitivity applies at two levels:

- The intrinsic sensitivity of FPGA concerns their internal logic (D flip-flop, latches, clock trees…). The problem is the same as for ASICs, and similar circumventing methods shall be used.
- The operation of FPGAs can be perturbed by SEUs on their programming element:

- SRAMs used for SRAM based devices are sensitive to SEU; errors may affect the programming of the component, causing changes in the circuit functionality.
- EPROMs or EEPROMs can be affected by total dose effects, but are generally insensitive to SEU. However, the write operation is controlled by sequencers that are most often highly sensitive to SEU.

Since ACTELs are the most commonly used FPGAs in space applications, circumventing methods dedicated to ACTELs will be described hereafter. The case of SRAMs FPGAs will also be examined.

4.6.1 Protection of ACTEL devices

ACTEL devices are available in normal version (low total dose tolerance) and RH version (300 krads total dose tolerance).

Protection solutions will be given for ACTEL 1280 device, but the circumventing principles will remain the same for other ACTEL devices.

The standard ACTEL 1280 have a low total dose tolerance, but they are latchup immune [KATZ94]. The RH version does not improve significantly SEU performance compared to A1280 devices, and 3.3 V operation increases the SEU rate by a factor of 2 [KATZ94].

ACTEL documentation contains a set of useful application notes related to radiation hardening such as [ACT97].

Two types of logic modules are used in ACTEL 1280 devices:

- C-modules, which are combinatorial modules
- S-modules which are sequential modules and thus contain an additional flip-flop

There are two methods to improve the SEU sensitivity of a DFF of an ACTEL 1280:

- *Use of combinatorial structures for DFFs*: different associations of modules can be used to build a DFF (C-C, C-S, S-C, S-S or single S). The less SEU sensitive DFF is built with two C-modules, and the improvement factor is about 60 compared with a DFF built with a single S-module ("SFF") [MATT96]. When using a VHDL based method, most of the synthesis tools allow to avoid the use of SFFs in a whole entity (for further details, please refer to the ACTEL application note "Using Synopsys to design ACTEL's Radiation Hard FPGAs"). If only specific registers must be protected, the problem is more complex and may lead to instantiate the components in the VHDL model (the code will be less readable).
- *Triplication of DFFs*: TMR is made as in ASIC by using 3 DFFs, but the implementation shall be optimised for ACTEL (the voter is made by using a MUX4 cell, the error signal is made by using an inverter and another MUX4).

  The triplicated DFF is shown below:

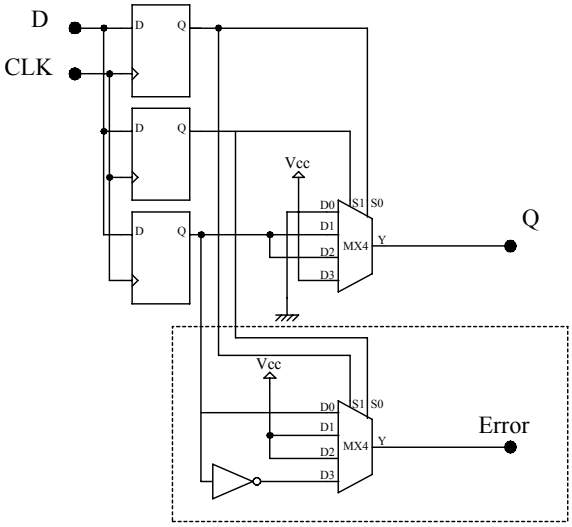

Figure 4.6.1 -1 - D flip-flop implemented with Triple Modular Redundancy and error reporting signal [ACT97]

Care must be taken when using TMR since the ou optional ter is not hazard-free. This signal must never be used to feed the clock input of DFFs, or the set or reset asynchronous inputs of DFFs. It must neither be used as read or write signals to control memory devices.

It must be noticed also that the TMR devices is functional only if the D flip-flops are refreshed by an active clock. If a gated clock is used, for example to save power consumption, the refreshment of the D flip-flops is not done and multiple SEU errors can occur. This problem can be solved by using a TMR structure with refresh as depicted in Figure 4.6.1-2. If the enable "E" signal is high the three D flip-flops are loaded with the new value. If the enable signal is low the memorised value is loaded in the three D flip-flops after the majority voter, i.e. after error correction.

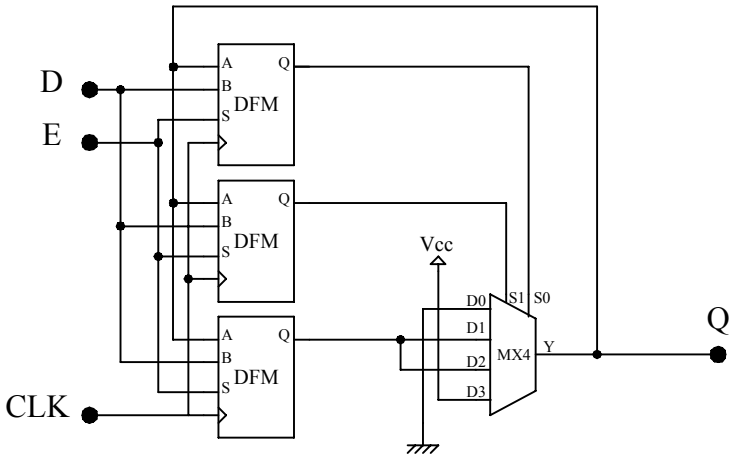

Figure 4.6.1 –2- Register element using TMR with continuous refreshment [ACT97]

Care must be taken when using TMR since the output of the voter is not hazard-free. This signal must never be used to feed the clock input of DFFs, or the set or reset asynchronous inputs of DFFs. It must neither be used as read or write signals to control memory devices.

…

The testability of TMR structures is as problematic as in ASIC. If a failure occurs in the TMR devices it may be masked by the majority voter. To cope with this problem, it is possible to connect all the error outputs together by using OR gates, for detecting permanent errors at DFF outputs. Another solution is to use the ACTEL ActionProbe to observe ACTEL internal signals, but it is difficult to apply in a production process of flight components and boards.

Two SEU protections for ACTEL devices can be envisaged:

- **use of combinatorial structures for DFFs (DFF built with 2 C-modules)**
- **triplication of DFFs (TMR)**

Moreover, it can be noticed that:

- **ACTEL RH versions do not improve significantly SEU sensitivity**
- **3,3 V operation (instead of 5 V) will degrade by a factor of 2 the SEU sensitivity**

4.6.2 Circumventing in SRAM based FPGAs

On the contrary to antifuse-based FPGAs, SRAM-based FPGAs keep their configuration by using SRAM cells. The consequences of a SEU in SRAM cells can be the followings [KATZ97]:

- A stress of the FPGA by creating a contention on two drivers of two internal cells, or a bus conflict on internal tri-state busses.
- A change in the functionality of the FPGA
- A stress of the component surrounding the FPGA by changing the direction of an input buffer to an output buffer.

Heavy ion testing of XILINX and ATMEL SRAM-based FPGAs shows that configuration memories are very sensitive to SEU with an extremely low Linear Energy Transfer (LET) of about 4-5 Mev.cm^2/mg. A large number of memory bits is required to program a single gate (typically from 13 to 32) which increases the cross section of SRAM based FPGAs. Since 1 million gates FPGAs are announced in the near future, configuration memories in the order of megabits can be expected.

Very few countermeasures exist to prevent a change in the configuration memory of SRAM based FPGAs.

It is not possible to use Triple Module Redundancy or correction codes for the configuration memory since it is not offered by the manufacturer. Moreover, the large number of bits required for configuration would lead to a huge overhead in silicon if such a method was used.

It is generally possible to read the configuration memory of SRAM based FPGAs during operation, and to compare it to a reference. A XILINX application note provides information on the subject HOF15]. Three solutions can be envisaged :

- The configuration data issued from readback is compared to the configuration loaded in the XILINX PROM. Such operation is attractive but not often possible, since the data issued from readback differs from initial configuration data specially if internal CLB RAM is used in the XILINX.
- A checksum of the content of the configuration memory is made. It is compared to a reference checksum held in a trusted register. A sufficient number of bits shall be used for the checksum to allow a safe protection. XILINX provides a readback checksum of 11 bits, i.e. one in 2048 errors might go undetected.
- The XILINX itself can be triplicated, and the readback performed simultaneously on the three components. When the bit streams differ, the outputs of the differing FPGA are disabled. But this solution seems complicated to implement at hardware level.

The readback is thus a possible solution to detect a change in the configuration memory of SRAM based FPGAs. It will not prevent an SEU to change the content and thus the program of the FPGAs. Extra studies are necessary to guarantee that the FPGA will not enter in a state that will destroy it or will damage its environment.

SRAM-based FPGAs are very sensitive to SEU, but very few countermeasures exist to prevent a change in the configuration memory. The only possible circumventing technique consists in reading the configuration memory during operation, and to compare it to a reference, but fault masking is not performed with this solution.

As a consequence, it seems very difficult to accurately prove that an SRAM-based FPGAs used in flight equipment will behave safely, and to recommend its use in critical functions such as AOCS. Nevertheless, some XILINX component were used in flight for scientific experiments.

4.7 SIMULATION OF SEU EFFECTS AT LOGIC LEVEL

The simulation of SEU effects at logic level requires being able to simulate the non faulty device. For this:

- A model of the device is required (behavioural model (c, VHDL, Verilog), RTL model (VHDL, Verilog) or gate level netlist (edif, VHDL, Verilog)
- A simulator able to exercise the model of the device is necessary
- A set of patterns to be applied to the device is needed

It must be noticed that in the European Space industry the high-level description language commonly used is VHDL that will be the baseline language in this section.

Different fault injection methods by simulation are given hereafter:

- a first fault simulation method using the VHDL language is based on the definition of a new type derived from the bit type with three new states representing the fault information [NAVA94]. The bit

type becomes the fbit type (for faulty bit) which is defined by the following enumeration : '0', '1', 'none', 'sa0', 'sa1'. This method seems difficult to use because:

- defining a new type requires to overload all the logic operators such as NAND, NOR etc...,
- a new bit type and in fact a new std_logic type with all the relevant procedures shall be defined,
- the code used for simulation is completely modified compared with the initial code.

- another method consists in modifying the model by insertion of "saboteurs" and "mutant" in order to inject fault by modifying the VHDL code [JENN94]. A *saboteur* is a VDHL component that alters the value or timing characteristics of one or several signals when activated. A *mutant* is a component description that replaces another component description. It can be easily done in VHDL by replacing an architecture by another one by the mean of a configuration mechanism. Nevertheless, the task of coding a mutant may require significant efforts depending on the complexity of the fault injection process that has to be modelled.

- the other method uses the commands of the VHDL simulator (Force/Noforce mechanism), and is based on signal and variable manipulations [JENN94]. It has the drawback to be dependent of the chosen tool, for the commands as well as for the results. Nevertheless, this method has the advantage to be simple to use since no modification of the VHDL model is done. The following commands must be applied to the simulator :
    - Run of the simulator up to a given time
    - Force a signal with a given value
    - Run the simulator for a given duration
    - Unforce the signal
    - Run the simulator for a given value to observe the fault effects.

The injection of SEU errors by using the simulator commands seems the easiest method to check if the SEU protection structures behave as expected. Nevertheless, such a method gives only an indication, since exhaustive simulations cannot be done, the number of cases to test being too large.

The following elements are required to test SEU with a VHDL simulator:

- *A model of the system*. RTL or gate-level models are preferred.
- *An understanding of the fault mechanism*. The possible effects of SEU in digital design are either a hit of a storage element, or a hit of the combinational logic that propagates an error to storage elements, or clock inputs of latches.
- *A modelling of the fault effects*. The gate level model allows SEU fault injection in the clock tree and in combinational logic.

  The RTL model allows SEU fault injection in combinational logic, but the correlation between the VHDL model and the hardware is sometimes difficult and thus simulated effects may have no

meaning. Nevertheless it must be noticed that the effects of SEU in combinational logic are transient effects, that are mitigated by the propagation through gates. An analog simulator would be preferable to characterise such mechanism.

SEU Errors can be injected in SRAM by modifying their behavioural model, since this model is most often the only available. If the memory cells are stored in an array of variable, the fault injection can be done easily. If the behavioural model of the RAM uses a dynamic management of the RAM, the injection may be more complex.

- *A set of test vectors* to exercise fully the model (generally available for an ASIC design).
- *A test plan* defining the simulation to be run, the location and the injection time of the faults. For a given simulation that lasts Nck clock cycles, if the design is a synchronous design having Ndff DFFs, the number of possibilities to inject a single fault in a DFF is Nck*Ndff. Exhaustive simulations cannot be run in these conditions, but:
  - simulation of a given function of the ASIC during a very limited duration can be envisaged,
  - fault injection can be exercised by a random mechanism that chooses the DFFs to flip and the clock cycle to make it. This mechanism is similar to the SEU testing of component by using a particle accelerator, except that the speed of experimental testing is more than 10^6 faster than VHDL simulation.
- *A self-checker test bench* must be developed. In case of an ASIC design, patterns are compared with a reference, either by a continuous VHDL mechanism or by an off-line comparison. In case of SEU error simulation, the simulation must stop when the error is detected to limit its length. It must be noticed that testbenches are generally written to exercise the ASIC with functional pattern that are compliant with a specification. The testbench may be quite disappointed by the ASIC response after an SEU hit. Adaptation may be required.

The injection of SEU errors by using simulator commands seems the easiest method to check if the SEU protection structures behave as expected (detection signals are correctly activated, the ASIC enters in the expected mode). Nevertheless, such a method gives only an indication, since exhaustive simulations cannot be done, the number of cases to test being too large.

The following elements are required to simulate SEU error injection:

- **A RTL or gate-level VHDL model of the system**
- **An understanding of the fault mechanism and a modelling of the fault effects**
- **Test vectors**
- **A test plan**
- **A testbench**

4.8 TESTABILITY OF PROTECTED FUNCTIONS

For a given type of circuit, high testability will guarantee that bad circuits are rejected and good circuits are accepted. Hardening techniques used to protect functions against SEU lead to testability degradation. In particular, if a failure occurs in a fault masking protected devices (error correcting code or TMR), it may be masked by the correction mechanism (majority voter for TMR or correcting code). Some methods to improve testability of protected functions will be shortly described hereafter.

- Scan methods can be applied to protected devices as for unprotected devices. For this, scan DFFs shall be available in the library, at the price of an increased area of the chip.
- Another solution is to use self-checking structures, able to detect both transient and permanent hardware faults during the normal operation of the circuit. However, self-checking techniques are generally employed to meet the most stringent safety or reliability requirements, not only for improving the testability. Twin-rail techniques combined with Berger codes or Smith codes can be used for this with a strong area overhead and timing performance degradation.
- Error signals generated by correcting codes or majority voters can be connected together by using OR gates. Permanent errors at DFF output will then be detected, with extra combinatorial logic.
- Extra observation signals can be added to increase observability of the circuit. These signals can be connected together by using XORs, in order to detect single errors.

All these methods will add extra gates to the protected devices for testability improvement. Compromises shall be found between area overhead, timing degradation and testability improvement: the adequate solution will depend on the circuit, synthesis and simulations shall be run with different methods to compare them.

5 SEU HARDENING AT CELL DESIGN LEVEL

Single Event Upsets cause bit flips in storage elements, or transients that propagate in combinatorial cells. To fight against these effects, a first possibility is to develop a hardened library for reducing the intrinsic SEU sensitivity of the used cells. Different hardening possibilities for the storage cells and a qualitative approach for hardening combinatorial cells will be described in this section.

5.1 SEU ASSESSMENT

5.1.1 Numerical SEU assessment

The first SEU assessment method uses dedicated software programs, the accuracy strongly depends on the models (environmental model, 'ion track' – 'sensitive volume' interaction). These computer calculations are complex and involve many assumptions, particularly about device geometry and thus can be used as a means to perform a relative SEU assessment of digital circuits. The CRIER (Cosmic Ray Induced Error-Rate analysis) program, the CRUP (Cosmic Ray Upset Program) and the CREME (Cosmic Ray Effects on MicroElectronics) program are the most often used.

5.1.2 Practical SEU assessment for MOS circuits

The second SEU assessment method is more practical. The aim is to calculate the probability that a node with a known surface area will be hit. For MOS circuits, mainly drains of n and p transistors that are in an off-state are sensitive to SEU.

For the Adams 90% worst case environment (corresponding to the integral LET spectrum for a standard geosynchronous orbital environment where the particle distributions are exceeded 10 % of the time), the following approximate formula is often used:

$$Phit = \frac{5.10^{-10}.ab.c^2}{Q_{crit}^2},$$

where :

- Phit : hit probability : number of particles with $Q_{coll} > Q_{crit}$ that hit area ab per day
- ab : surface of the drain area of the sensitive node in square microns.
- Q_{crit} : critical charge in pC of the sensitive node. If the amount of charge collected by the sensitive node at the moment of a SE impact (collected charge Q_{coll}, proportional to the LET) exceeds the critical charge Q_{crit}, then the passage of the ion will upset the circuit. The order of magnitude for the maximum value for Q_{coll} is in the range of 3 to 6 pC,
- c : collection length in microns

In order to reduce the hit probability, 3 parameters can be manipulated: ab, Q_{crit} and c. The first two parameters are under control of the circuit designer, the last parameter is technology dependent (it can be kept low by using epi layer technology instead of bulk CMOS or by using CMOS SOI/SOS instead of standard CMOS).

To obtain the error rate of a specific node, Phit must be multiplied with the probability that the hit node is sensitive and the hit really leads to an error:

- if the node is a 'memory node', i.e. a node contained in the cross-coupled inverter loop of the memory element, then the probability of the node to be 1 (or 0) is assumed to be ½
- if the node belongs to a clock inverter then the distributions of logical values of clock, data and memory loop nodes are involved. Each of these probabilities can be assumed to be equal to ½, which makes the probability of the node equal to 1/8

The error rate for a circuit containing multiple sensitive nodes is simply the sum of the error rates of the individual sensitive nodes. For a single cell, the error rate tends to be dominated by the most sensitive node. The figure obtained should be more interpreted as a « figure of merit » rather than an absolute indication for the error-rate (please refer to WP220 document for more details).

5.2 SIMULATION OF SEU AT THE CELL LEVEL

Simulation of SEU at the cell level can be done with one of the commonly available circuit simulators (SPICE, HSPICE, ELDO,…). In the simulation an ion hit is emulated by extracting (or adding) charge to the sensitive node. If the circuit upsets, a critical charge has been deposited: the pulse shape (time distribution of the transient pulse) shall be described as well as the critical charge. This shape however is very dependent on parameters like LET, technology (epitaxial layer thickness, doping levels, …), location of the perturbation. The amplitude and time profile of the transient pulse should reflect as much as possible the phenomena in reality.

The charge collected (Q_{coll}) by a sensitive node in a circuit as a result of a single event hit is comprised of three components:

- the diffusion component results from direct ionisation due to the particle impact. It includes charge that moves under the influence of excess-carrier gradients
- the drift component (often referred to as 'prompt charge') results from direct ionisation due to the particle impact. It is comprised of a charge component that results from carriers generated by the passage of the ion through the associated depletion region of the sensitive node augmented by a "funnel" charge component
- the parasitic bipolar component is significant for CMOS/SOI devices [KERN] and also devices with small feature sizes. Indeed, an event induced voltage perturbation in the device body can, as secondary effect, activate the parasitic bipolar inherent in these structures. The influence of this last component is not negligible and can have current amplitudes of up to 60 % of the current generated by direct ionisation. The bipolar contribution is significant and neglecting this component results in an underestimation of the SEU vulnerability in terms of critical LET. Normally, in order to have a

good idea of this bipolar contribution, a non-evident careful characterisation of this parasitic bipolar structure is needed

Simulation and hardening of SEU initially happens at the circuit level. However, as soon as possible, simulations should be performed on layout extractions to include all parasitic (capacitive) effects because the effects of these parasitic may be non-negligible (in the good sense).

For the simulation of SEU at the cell level, two major methods are reported in literature: the Current source method and the Rabe & Golke method.

5.2.1 Current Source method

Typical transient current shapes in p/n junctions following an ion strike are known in literature. They are obtained either by sophisticated numerical simulations or by experiment. However, these shapes are, as already mentioned, very dependent on parameters like LET, technology (epitaxial layer thickness, doping levels, feature size, …), location of the perturbation, ... Specific data for a given technology are often not available. This means that for simulation some typical transient current shape must be applied to the circuit. This is done by means of a current source with appropriate current shape. In practice different kinds of shapes are used (see WP100 report).

A very commonly used shape is depicted below:

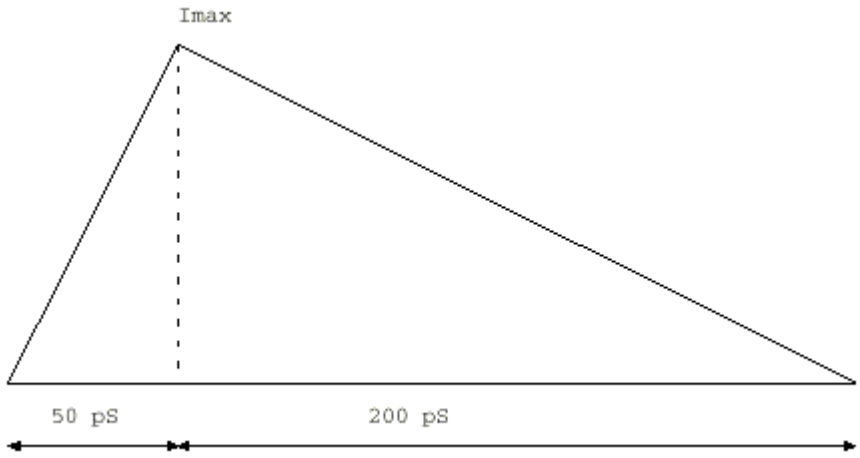

Figure 5.2.1–1 - Typical current shape used for SEU simulation

The value for IMAX is typically in the order of tens of milli-amps.

A typical circuit that can be used for current source SEU simulation is shown below:

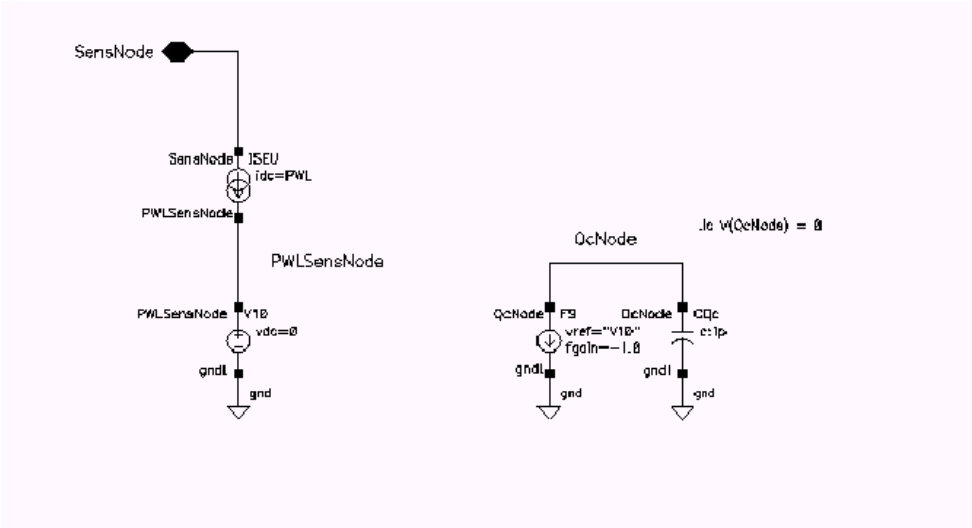

Figure 5.2.1–2 - Circuit for SEU simulation with the Current Source Method

The hit node should hereby be connected to SensNode. The circuit is comprised of a current source exhibiting the desired pulse shape together with a current mirror and a capacitor of 1pF. In order to obtain the figure for Q_{crit} the current of the current mirror is integrated by the capacitor. The voltage in Volts over the capacitor equals Q_{crit} in pF (for more details, please refer to WP220 report).

5.2.2 Rabe & Golke

This method was developed by R.Rabe and K.Golke of Honeywell in 1982 [DAWE] and is illustrated in Figure 5.2.2-1. The proposed model is a fairly crude but satisfactory representation of the actual collection process. The method differs from the previous one in that a fixed current profile is not implied, but that a user-predefined amount of charge from the sensitive node is extracted.

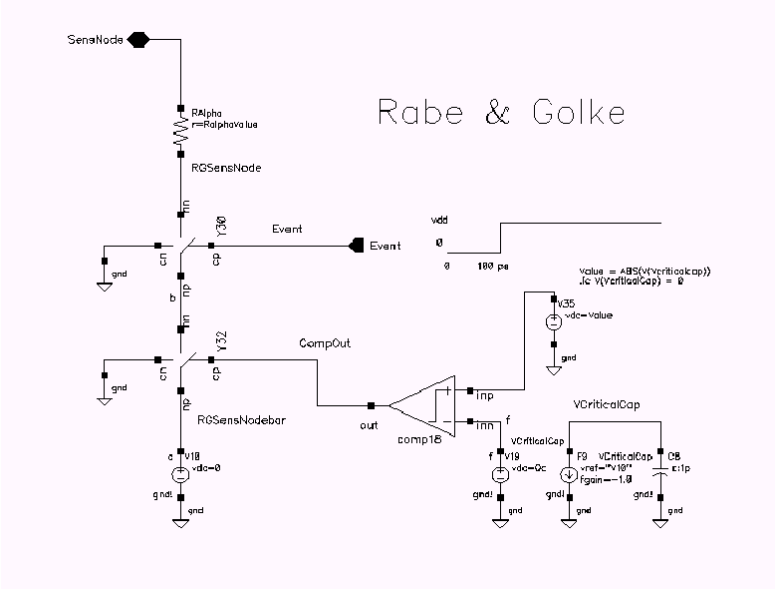

Figure 5.2.2–1 - Circuit for SEU simulation with the Rabe & Golke Method

…

The bullet at the top side of the resistor RALPHA is to be connected to the sensitive node through a zero-volt voltage source. During the event the sensitive node is 'shortcut' to the substrate (or well) through RALPHA via switches Y30 and Y32. To trigger the event, a step function is applied to node Event which closes switch Y30. The current through RALPHA is mirrored and integrated on capacitor C8 (1pF). If the voltage on the capacitor has reached the user-predefined threshold value the comparator opens switch Y32 and the sensitive node is disconnected (for more details, please refer to WP220 report).

A difficulty that needs to be solved here is the determination of the value for resistor RALPHA. A good value for RALPHA is obtained by tailoring it in such a way that the values for critical charge, obtained by simulation using both Rabe & Golke and Current Source methods, are the same.

5.2.3 Rabe & Golke versus Current Source Method

There are some limitations with the current source method:

1. The application of a typical, not technology specific, predefined current shape of fixed duration remains more or less arbitrary and accuracy of results can only be as good as the accuracy of the applied current waveforms.
2. The method may incorrectly predict Q_{crit} because the current source causes often the junction being hit to forward-bias and sink a large amount of charge to the supply. Lumped-parameter circuit simulators, such as SPICE, are designed to simulate normal operation of solid-state devices. Perturbations introduced by single events may force devices in operation modes for which they were not designed (e.g. forward biasing of drain-substrate junction). SPICE models of MOS devices are not designed to include device operation without junction isolation and should therefore be modified to include also these operation modes. This can be done by using diodes and bipolar transistors configured in parallel with the MOS device. Each technology poses its own set of challenges to do this in an accurate way. With more accurate modelling for the MOS transistors more accurate values for the critical charge will be obtained and in this special case a finite value for Q_{crit} can be expected. A general conclusion anyway is that this node is not very sensitive to upset.

The explanation above indicates that, from a practical point of view, the current source method is not so easy to use. The Rabe & Golke method does not exhibit the problems reported for the current source method because it works fundamentally differently, all devices being simulated in their normal operation mode (no diode clamping etc.). However the link to the physical phenomena during SEU is less direct than for the current source method. From a practical point of view the Rabe & Golke method is preferred to the current source method.

5.2.4 Simulation Conditions

The simulations for SEU should be done in worst-case conditions. The main criterion for worst-case condition is the ability of the "ON"-transistor connected to the struck node to provide sufficient resupply to recover from the perturbation. Taking this into account the worst-case conditions are :

- Maximum temperature

- Minimum supply voltage
- Maximum total dose irradiation
- Worst-case process
- No capacitive loading of the circuits outputs.

Simulation of SEU at cell design level shall be run with available circuit simulators, such as SPICE, HSPICE or ELDO, in worst-case conditions (maximum temperature, minimum supply voltage, maximum total dose irradiation, worst-case process and no capacitive loading of the circuit outputs).

Emulation of the ion hit can be made by injecting to the sensitive node a critical charge with appropriate current shape (current source method), or by extracting from the sensitive node a user-predefined amount of charge (Rabe & Golke method).

From a practical point of view, the Rabe & Golke method is preferred to the current source method.

5.3 SEU HARDENING

Different solutions for hardening storage and combinatorial cells at the cell design level will be proposed hereafter. In general, it is very difficult to completely prevent a device against SEU without loosing its timing performance and integration. A compromise between area, speed and SEU hardness must always be made.

Three ways are possible to harden devices against SEU:

1. by minimising the amount of charge Q_{coll} that can be collected by a sensitive node per event. This can only be accomplished by process enhancements:
   - Use of highly doped substrate: this limits the extension of the electrical field in the depletion region following an ion strike, and thus reduces the funneling effect and the charge collection.
   - Use of an epitaxial layer instead of bulk CMOS: this limits the charge collection distance to the thickness of the epitaxial layer.
   - Use of CMOS SOI/SOS instead of standard CMOS, with the drawback of lower density and higher processing costs
2. by maximising the critical charge necessary to produce a logical upset; different methods can be used:
   - enhancement of stored information (capacitive hardening, drive strength hardening)
   - addition of redundancy to stored information (transistor hardening)
   - isolation from stored information (resistive hardening, capacitive hardening, glitch filtering)

All these methods will be detailed hereafter. Techniques that introduce high static power consumption will not be considered.

3. by minimising the area of the sensitive node. At layout level this can be accomplished by sharing as much as possible the sensitive areas of the same type (p/n) connected to the same node.

5.3.1 Storage cells

The SEU sensitivity of storage cells can be split into 2 types:

- 'Internal' sensitivity: the storage cell upsets due to a hit on a sensitive node internal to the cell. This can be handled and solved in the cell itself, using resistive, capacitive, drive strength and transistor hardening techniques.
- 'External' sensitivity: the storage cell upsets due to transients (caused by ion hits elsewhere in the circuit) on its inputs (clock, data, asynchronous set/reset lines). This can be handled both internally (glitch filtering) and externally (combinational logic hardening).

5.3.1.1 Drive strength hardening

This technique increases the drive strength of the sensitive node, and thus decreases the internal sensitivity of storage cells. In practice the drive strength hardening method requires that sizes of transistors are increased. Drive strength hardening always also implies capacitive hardening because parasitic capacitances increase if transistor dimensions increase.

On the layout level, drive strength of nodes can be increased by reducing as much as possible parasitic resistance in the restoring path.

SEU HARDENING AT CELL DESIGN LEVEL - DRIVE STRENGTH HARDENING

The drive strength hardening technique consists in increasing the drive strength of the internal nodes, by increasing the sizes of the transistors.

- ***Advantages*:**

  - High speed performance

  - Conventional CMOS process

- ***Disadvantages*:**

  - Area penalty: order of 100% and more to obtain required SEU performance

  - Propagation of transients not prohibited but even enhanced. For example : in a memory cell a 'drive-strength-hardened node' becomes indeed harder but the other memory node becomes weaker because propagation of transients through the memory feedback loop has improved. This means that both memory nodes must be hardened.

  - Increase of dynamic power (proportional to drive strength)

  - Scaled devices are increasingly difficult to harden

5.3.1.2 Capacitive hardening

Capacitive hardening can be applied by adding capacitances (linear, non-linear MOS capacitance) but is most of the time combined with drive strength hardening (increase transistor sizes to increase parasitic capacitances). This technique decreases the internal sensitivity of storage cells.

This technique increases node capacitances and thus acts as a filter on the transients (reduction of the amplitude and pulse width of voltage transients).

SEU HARDENING AT CELL DESIGN LEVEL - CAPACITIVE HARDENING

The capacitive hardening technique consists in increasing node capacitances in order to filter the transients.

- ***Advantages*:**
  - **Conventional CMOS process**
- ***Disadvantages:***
  - **Area penalty: order of 100% and more to obtain required SEU performance**
  - **Speed penalty: capacitive hardening applied without drive strength hardening leads to degraded speed performance similar to resistive hardening**
  - **Increase of dynamic power (proportional to node capacitance)**
  - **Slow recovery after perturbation (if not in combination with drive strength hardening)**
  - **Scaled devices are increasingly difficult to harden**

5.3.1.3 Resistive hardening

A very common type of single-event upset hardening is resistive hardening. This technique requires that resistors are introduced in the feedback loop of the cross-coupled inverters that form the storage element. Values for the resistors depend on placement of resistors and on the number of resistors used. The range for the resistor values is very broad, from a few hundreds ohms up to 1MΩ.

This technique slows down the propagation of fast transients, thus increasing switching time constants and allowing recovery of the cell before logic upset. For bistable data-storage elements, this involves the principle of discrimination between a short single event transient and a longer legitimate write signal.

A variety of possibilities exist for the number and placement of the resistors. Detailed descriptions are given in Appendix (section 8.4).

SEU HARDENING AT CELL DESIGN LEVEL - RESISTIVE HARDENING

The resistive hardening technique consists in introducing resistors in the feedback loop of the cross-coupled inverters that form the storage element, in order to slow down the propagation of fast transients.

- ***Advantages*:**

  - No power penalty

  - Limited area penalty because the resistor structure can be incorporated in the usual interconnection network (specially SRAMs).

- ***Disadvantages:***

  - Resistive hardening requires the availability of high-ohmic polysilicon resistors in the technology:

    - process control with acceptable tolerance limits for the high-ohmic polysilicon resistance values remain difficult and is still area of active research

    - a negative temperature coefficient of the high resistance polysilicon causes a slowed down write response of the storage device at low temperatures. The sheet resistance figure of the high-ohmic polysilicon at maximum temperature is to be used to ensure hardness over the full temperature range

  - Speed penalty: as resistive hardening always slows down the circuit response time, it will always mean a compromise between speed requirements and SEU tolerance. Always some aspect of timing performance is affected (setup & hold times, minimum clock widths, propagation times).

  - scaled devices are increasingly difficult to harden: as SEU immunity requires a minimum delay in the feedback loop of the memory element, the value of the resistors will increase with device scaling. As a consequence, write times do not scale any more according to the standard CMOS scaling rules.

5.3.1.4 Glitch filtering

As typical time constants of nowadays technologies (feature sizes of 0.8μm and less) continue to decrease, transients in combinatorial logic due to single events have no difficulties to propagate through the logic towards data inputs of memory devices. The glitch filtering technique consists in minimising the 'external' sensitivity of a memory device on its data input. Other inputs (clock and asynchronous reset/preset) do not need glitch filtering since they are assumed to be "clean" (specific buffers are used for clock and reset tree, see section 5.3.2.1). A reasonable objective for a glitch-filter that must filter out these transients is that it must be capable to filter out pulses with a duration up to 1 ns [BLA87].

Examples of implementations are given in Appendix (section 8.5).

SEU HARDENING AT CELL DESIGN LEVEL – GLITCH FILTERING

The glitch filtering technique consists in minimising the 'external' sensitivity of a memory device on its data input.

- ***Advantages*:**
  - **No power penalty**
  - **Conventional CMOS process**
- ***Disadvantages:***
  - **fixed speed penalty**
  - **area penalty: minimum 50%**

5.3.1.5 Transistor hardening

Transistor hardening uses the principle of redundancy of information combined with appropriate "state restoring" feedback from the uncorrupted data source. Most topologies proposed in literature exhibit:

- Two latch sections that store the same data.. The 'backup' latch is most of the time a latch consisting purely of NMOS or PMOS devices. The restriction to one type results in the latch being sensitive to only one type of hit (p or n hit) and this property is used in the hardened cell. The drawback is that logic levels in the 'backup' latch are degraded which can lead to solutions with high static current consumption. These solutions are not discussed here.
- Feedback loops to obtain state-dependent active feedback circuits.
- Use of ratioed inverters to avoid transient pulse propagation.

Obviously, the main challenge is to organise this extra latch and extra transistors (which imply new sensitive nodes!) in an efficient topology (minimal number of transistors) without affecting SEU sensitivity. Transistor hardened memory cells are inherently immune to ion hits on a single sensitive node. This means that for this type of events the pulse shape and time duration of the perturbation are completely irrelevant and the corresponding error-rate is equal to 0. However, a secondary error mechanism, Multiple Bit Upset (MBU) here is used to evaluate error probabilities: incident heavy ions may affect more than one sensitive node at the same time thus causing the cell to upset. The order of magnitude for these kinds of error probabilities is very low so that their contribution to the overall error probability of a system is most often negligible. On the layout level the probability can be made very low if the transistor drain areas occupied by the simultaneously sensitive node pairs are well spaced on the cell of the layout, so that the critical charge cannot be collected simultaneously at both nodes.

Different topologies have been proposed in the literature, some of them are discussed in Appendix (section):

- The HIT (Heavy Ion Tolerant) cell [BESS93]: composed of 12 transistors organised as two storage structures interconnected by feedback paths, it is designed for fast recovery after upset, low static consumption and no speed performance degradation. Based on the same principle, the ROCKETT cell (16 transistors) [ROC88] and the LIU cell (14 transistors) [LIU92] are bigger than the HIT cell.

- The DICE (Dual Interlocked Storage Cell) cell [CAL96]: based on feedback loops within a dedicated latch architecture, it uses an original principle called "dual node feedback control". The area overhead is high (between 70 and 100 %).

These cells can be used to construct latches and flip-flops if inverters for clock and/or data are added.

SEU HARDENING AT CELL DESIGN LEVEL – TRANSISTOR HARDENING

The transistor hardening technique uses the principle of redundancy of information combined with appropriate "state restoring" feedback from the uncorrupted data.

- ***Advantages*:**
  - **High speed performance**
  - **Conventional CMOS process**
  - **For the basic RAM cell, this kind of technique ensures SEU *immunity* to events on single nodes and not just a relative improvement of the SEU tolerance as other techniques do**
  - **Fast recovery after upset**
  - **Independent of supply voltage and temperature**
- ***Disadvantages:***
  - **Cell area**
  - **Ratioing of transistors not Total Dose independent**
  - **Sensitive to transients (due to combinatorial perturbations) at the cell inputs**

5.3.1.6 Hardening methodology for a storage cell

The development of hardened cells is not easy and this shall be left to specialised design centers. Hardening is an iterative process and at the beginning of each iteration the relative contributions of the different sensitive nodes are taken as basis for the next iteration of the hardening process. One iteration of the hardening process is shortly described below:

- The global error rate of the memory cell shall be assessed. For this, SEU sensitivity of all internal nodes of the cell connected to n- or p- drains of MOS transistors (sensitive nodes) shall be evaluated, the cell being in memorisation mode, with no active Set or Reset. For each sensitive node, the hit probability and the error rate shall be calculated (see section 5.1 and WP220 document for further details). The global error rate of the memory cell is the sum of all the individual error rates of the sensitive nodes.
- Based on the relative contributions of each sensitive node to the error rate, a hardening strategy is developed.
- A hardening iteration is performed, in order to reduce the sensitivity of the most sensitive node(s).

The hardening process can be stopped when a compromise has been found between the reached global error rate, the area increase and the timing degradation.

The following table summarizes the characteristics of each SEU hardening technique for storage cells:

	Principle	Advantages	Disadvantages	Comments
Use of highly doped substrate	Reduction of the funneling effect and the charge collection		Specific process	
Use of an epitaxial layer	Limitation of the charge collection distance		Specific process	
Drive strength hardening	Increase of the size of transistors	High speed performance Conventional CMOS process	Area penalty Increase dynamic power Transient propagation enhanced Difficulty to harden scaled devices	Makes intrinsic capacitive hardening because transistor dimensions increase
Capacitive hardening	Increase of the node capacitances to filter the transients	Conventional CMOS process	Area penalty Speed penalty Increase dynamic power Difficulty to harden scaled devices Slow recovery after perturbation	Generally combined with drive strength hardening
Resistive hardening	Introduction of resistors in the feedback loop of the cross coupled inver-ters	No power penalty Limited area penalty	Speed penalty Difficulty to harden scaled devices Requires high-ohmic polysilicon resistors	Alternative methods use only one resistor or active resistors. Prevents only the propagation of transients generated in their fanin
Glitch filtering	Glitch filtering with a duration up to 1 ns	No power penalty Conventional CMOS process	Fixed speed penalty Area penalty > 50%	Uses resistive or capacitive hardening techniques
Transistor hardening	Information redundancy with "state restoring" feedback from the uncorrupted data source	High speed performance Conventional CMOS process fast recovery after upset	Cell area Sensitive to transients at the cell input Ratioing of transistors not total dose independent	HIT cells and DICE cells use this principle

5.3.2 Combinatorial cells

Ions can interact with combinatorial logic to produce errors, but the mechanism is somewhat different from that in storage cells.

In order for errors to occur due to hits in combinatorial logic the following conditions apply:

1. A SE initiates a propagating voltage transient on a sensitive node.
2. A logical path (through combinatorial logic) from hit node to memory element exists.
3. At arrival at the memory element, the amplitude and width of the transient are sufficient to change the state of the memory element.
4. If the affected input of the memory element is the data input: the voltage transient has the correct timing with regard to the clock of the memory element in order to produce an error.
5. The value of the memory element is opposite to the SE induced effect.

A quantitative approach can be used for hardening the combinatorial cells. For this, a very good knowledge of the circuit is required, many parameters are needed and each sensitive combinatorial node shall be individually evaluated (for more information, please refer to WP220 document).

A qualitative approach is preferred. The two main principles that are used are:

- Limiting amplitude and width of the ion-induced voltage transient at the sensitive node to such an extent that connected circuitry is not affected by the perturbation (no upset on memory devices). This is generally carried out by capacitive hardening and/or drive strength hardening, with the drawback of density penalties.
- Preventing the propagation of the ion-induced transient. Because nowadays technologies have short propagation delays, induced transients are almost always propagated without hardening. The delay time of the combinatorial cell shall thus be increased to prevent voltage transients to propagate: transients with pulse width smaller than the delay of the cell shall not be propagated, transients with pulse width exceeding the delay of the cell shall be simply propagated (transients are assumed to last 1 ns). This is generally carried out by resistive hardening, with the drawback of speed performance penalties. Note also that a cell that is resistively hardened prevents transients at its inputs to propagate, but cannot prevent transients due to hits on its own sensitive nodes to propagate.

5.3.2.1 Clocks

Synchronous clocks, asynchronous reset/preset lines are special lines in a system (in the rest of this paragraph referred to as ‘clocks’). A good hardening policy for these nets consists in suppressing transients before they are transmitted to the memory device itself.

For a clock, the hardware structure can be split in 2 parts. Hits can occur in both parts, but are treated differently:

- the clock source: this part is made with combinatorial/sequential functions of limited complexity (multiplexer, clock divider …). Hardening of the clock source follows the rules of storage cells

hardening (see section 5.3.1) for the sequential part, or combinatorial logic hardening (see section 5.3.2.2) for the combinatorial part.

- the clock tree connected to the actual clock: it usually consists of a network of large buffers driving nets with high capacitive load. Because of the high drive strength of the buffers and the high capacitive load on the clock net, transients on the clock net are not likely to have direct impact on the connected memory devices. In practice only dedicated buffer cells should be used in the clock tree. All stages (especially input stage) of these cells should be hardened against SEU by the drive strength hardening technique, assuming that the capacitive load on the cell will never be lower than a well-chosen minimum capacitance C_{min}. A reasonable value for C_{min} can be obtained by assuming reasonable minimum numbers of connected cells (buffers or clock inputs of memory devices). On system level the minimum capacitance requirement must be guaranteed by a design rule check (DRC check). Dummy capacitance must be added if necessary. In this context it is also better to have one large central clock buffer than a distributed clock tree with clock buffers with lower drive capability (indeed, the effect of a single event is much better compensated in a centralized very powerful clock buffer than in a tree where less powerful buffers are distributed).

Moreover, clock hardening requires a filter buffer placed at the input of the clock tree in order to prevent transient propagation from the clock source. The most appropriate technique consists in placing a resistor of a few kΩs between the two stages of the dedicated clock buffer cell (resistive hardening technique). For timing reasons it is recommended to use this 'filter-buffer' only once per clock tree. A circuit diagram of the 'filter-buffer' is shown below.

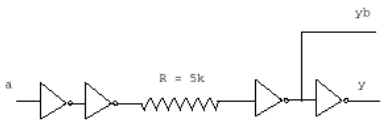

Figure 5.3.2.1–1 - Filter-buffer that filters out all voltage transients up to 1 ns

The part of the 'filter-buffer' behind the resistor (last stage(s) of the buffer) is usually SEU hard by itself because of the drive capability of the buffer. If not, it must be additionally hardened by the drive strength hardening technique.

SEU HARDENING AT CELL DESIGN LEVEL – CLOCK HARDENING

Clock source hardening follows the rules of storage cells hardening for the sequential part, or combinatorial logic hardening for the combinatorial part.

Clock tree hardening requires:

- the exclusive use of dedicated clock buffers,

- a minimum capacitive load Cmin for each stage of these cells (checked by DRC tools)

- the use of one large central clock buffer (preferred to a distributed clock tree with clock buffers having lower drive capability)

- the use of a filter-buffer (resistor of a few kΩ between 2 stages of the dedicated clock buffer)

5.3.2.2 Combinatorial logic

Combinatorial logic leading to normal data inputs of memory devices can be very complex. Because of density and/or performance requirements, straightforward use of resistive hardening or capacitive/drive strength hardening on combinatorial logic is not possible.

- Capacitive/drive strength hardened cells impose an area penalty and can therefore only be used for smaller circuits, for example controllers.

- Resistively hardened cells decrease timing performance and only prevent propagation of transients generated in their fanin; they cannot therefore be used for the last cell in front of the memory device data input. In order for a data input of a memory device to be 'transient safe', the last cell in front of the data input must be resistively AND capacitively (drive strength) hardened. A simpler solution is to incorporate the filtering within the memory device but also here this is at the cost of decreased timing performance (see section 5.3.1.4 concerning glitch filtering).

- Another solution to this problem can be provided at the system level by using fault tolerant techniques (see section 4 concerning SEU hardening at function level design).

Different sensitivities of NMOS transistors versus PMOS transistors can lead to a preferred choice for implementation of logic functions. For example, a sensitive NMOS device leads to bigger PMOS devices for compensation: in this case NAND is preferred to NOR.

SEU HARDENING AT CELL DESIGN LEVEL – COMBINATORIAL LOGIC HARDENING

- **Capacitive/drive strength hardened cells can only be used on smaller circuits, due to area overhead.**
- **Resistively hardened cells decrease timing performances. For the last cell in front of memory device data inputs, both resistively and capacitively (drive strength) hardened cells shall be used.**
- **Fault tolerant techniques can also be used at system level to cope with combinatorial cells hardening**
- **A cell selection in the library can be necessary for a good compromise between the hardening performance, the area overhead and the speed degradation.**

5.3.3 Future trends

As microelectronic device features are getting smaller, the corresponding response times become faster and the amount of stored charge representing information is decreasing. As a consequence, the device SEU sensitivity has a strong relationship with the feature size. The consequences are that :

- Scaled devices are more sensitive to SEU for their combinatorial part, since response times are lower and voltage pulses due to particle hits are no more filtered
- Scaled devices are increasingly difficult to harden and hardening penalties increase regardless which hardening method is applied :

1. Drive strength hardening : drive capability of standard cell stages decreases with smaller feature size, leading to a higher SEU sensitivity. To keep the same SEU sensitivity (in terms of drive strength) as before scaling, transistor widths should remain constant regardless of the feature size.

2. Resistive hardening : the basic time constant for hardening equals ≈1 ns. With ever decreasing device capacitance values while sheet resistance remains more or less constant (independent of feature size), this results in the use of resistors with ever increasing resistance values as function of feature size.

3. Capacitive hardening : maintaining identical SEU sensitivity (in 'capacitive' terms) after scaling requires node capacitances to remain constant independently of feature size, which is not the case. This means that also this method does not scale well with feature size.

- SEU is not limited any more to the galactic environment : advanced integrated circuits will also be sensitive to earth environments. Future designs of all integrated circuits will be guided by principles developed for SE hardening
- In the galactic environment, for instance for solar flares, the problem becomes worse.

A 'feel' for device sensitivity to the feature size can be obtained from the observation that the critical charge scales as the square of the feature size, i.e. : $Q_{crit} \sim l^2$. The data from years of empirical measurements over a broad range of technologies and feature sizes confirm this relationship [PET82]. This proportionality makes sense because charge is generally stored capacitively and capacitance scales as the square of the feature size.

New technologies with reduced supply voltage are expected to become more sensitive because switching levels are reduced, lowering critical charges compared to circuits with higher supply voltages [NASA].

6 DESIGN METHODS TO PROTECT AGAINST SINGLE EVENT LATCHUP

Single Event Latchup (SEL) is defined as triggering a parasitic thyristor of PNPN structure existing in CMOS or bipolar devices by ion strike (see section 3.4). When it occurs, an important current flows and increases the local temperature of the die, having destructive effects, unless power supply is quickly switched off (less than few hundreds of microseconds, depending on the type). When the component triggers in latchup, the latchup current is higher than 200 mA.

Latchup free technologies shall be used when it is possible (CMOS/SOS and CMOS/SOI are intrinsically latchup free technologies). CMOS/bulk (sensitive) and CMOS/epi (normally not sensitive, but new thinner technologies shall be evaluated) are considered latchup free when LET threshold is greater than 70 to 100 $MeV.cm^2/mg$.

If it is not possible to use such a technology, there are different ways to mitigate or suppress the latchup triggering, described in the following paragraphs.

6.1 LATCHUP PROTECTION OF COMPONENTS AT PCB LEVEL

6.1.1 General design guidelines

When a latchup free technology cannot be used, a possible SEL protection is to use an anti-latchup function located close to the circuit to protect on the PCB.

The role of the anti-latchup function consists in detecting the latchup current and limiting it, or switching off the component. Since the latchup effect is very fast (some few hundreds of microseconds), the anti-latchup function can be made neither by software, nor by using an electromechanical relay that switches in several milliseconds. When protection operates, a flag shall be activated for internal reconfiguration in case of redundancy or for ground information in the telemetry frame. When the component to protect is switched off by the anti-latchup function, it can be supplied back by its inputs or outputs, it is thus necessary to connect its inputs and/or outputs to VDD or VSS.

The anti-latchup function shall be adapted to the component to protect. Three categories of components can be identified, depending on the ratio between average current and latchup current. The method to choose the latchup protection design is the following:

1. Evaluation of the maximum average current of the circuit in worst case conditions of voltage, temperature, radiation and loads. This evaluation can be a paper task from the data sheet of the circuit or a breadboarding to have a real estimate if the data are not accurate enough. For microprocessors, this average current is very dependent of the software tasks. For other components, it can be simple if the component has only one functional mode. Else, the designer shall evaluate the more dimensioning mode.

2. Selection of one of the 3 cases:

- If the latchup triggering level is close or lower to maximum average current (less than 2 times), the design of the latchup protection is complex and shall take into account the operational modes of the circuit to protect. The measurement of the current shall be made with an inductor.
- If the latchup triggering level is close to maximum average current but not so far (between 2 to 4 times), the circuit will be more sensitive to current variation due to another cause (PCB common mode, EMC, etc.) than the latchup and could trigger for a false condition. It could trigger often depending on the quality of the design, the power supply, etc…
- If the latchup triggering level is far enough from the maximum average current (more than 4 times), it will only trigger for latchup conditions what are the quality of the design, the power supply, etc…

3. Margin calculation: it is recommended to take enough margin. 1,5 times to 2 times the maximum average current is a good compromise.
4. Validation of the design in worst case conditions (voltage, temperature, functional mode). A good way is to implement inside the detection circuit a mechanism to simulate the latch-up current.

6.1.2 Anti-latchup function for components with latchup current greater than average current

For devices having latchup current greater than average current (at least 4 to 5 times), a very simple protection circuit can be designed. This category concerns components with an average current less or equal than 40 mA (such as DRAM memories with an average current of 1 mA).

During normal operation (no latchup), the latchup protection circuit shall maintain the power supply voltage within the worst case limits and above the minimum voltage value.

When the circuit triggers in latchup, the latchup protection circuit shall limit the current, switch off the circuit and set a flag.

The simplest protection consists in inserting a resistor in the power supply line: when latchup occurs, overcurrent will be limited by the resistor, but there is no flag to warn the system.

Three parameters must be taken into account to design the latchup protection: the power supply voltage, the current flowing in the circuit and the response time.

The supply current must be averaged to eliminate spikes and peak currents, and it can then be measured by using a resistor in the power supply line or by measuring a flux variation in an inductor.

The use of a resistor R has the drawback to decrease the voltage level of the circuit ($V_{DD} = V_{in} - R*I_{average}$): a compromise shall be found between the voltage on the resistor ($R*I_{average}$) and the voltage level of the circuit (V_{DD}). The voltage on the resistor is compared with a threshold value, in order to activate the protection or not. The comparison can be done by a voltage comparator circuit (more accurate) or by a transistor (easier to implement, but less sensitive because requiring more voltage on the resistor). For this category of circuits to

protect, a transistor circuit can be used because the discrimination between latchup current and average current is easy. An example of such an anti-latchup circuit is shown below.

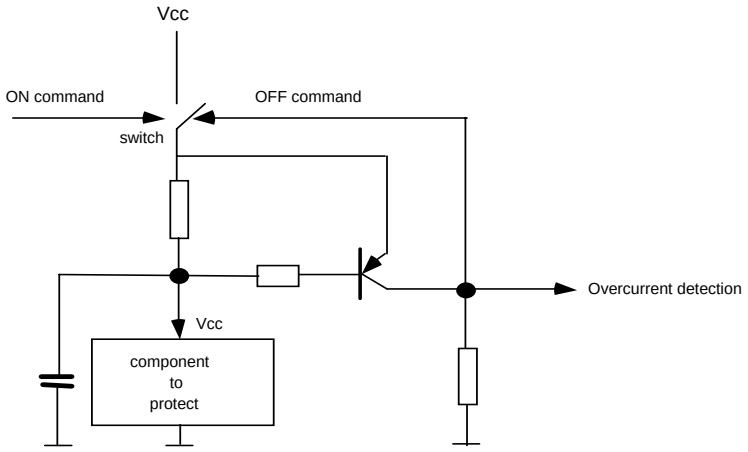

Figure 6.1.2–1 - synoptic of anti-latchup circuit for components with latchup current greater than average current

The PNP transistor is used like a comparator: when $V_{be} < 0{,}2$ V, it is blocked; when $V_{be} > 0{,}6$ V, it is passing.

The voltage on the resistor supplies the V_{be} of the transistor. The resistor is calculated depending on the average current. The latch-up detection is done when the transistor is passing. In normal conditions, the transistor is locked because its V_{be} is less than 0.2 V.

The current is filtered by a capacitor to get the average current of the component.

When the transistor is passing, the voltage on the collector pin is about Vcc. This voltage is used to switch off the Vcc voltage. When the circuit is switched off, it cannot be supplied again because Vcc is OFF. The transistor also delivers a latch-up flag, which can be used to trigger a local reconfiguration for redundant units or to flag in the telemetry frame. The flag shall be memorised because Vcc is then OFF after latch-up.

Then, an ON command has to be sent to supply the component again. This ON command is under control of a local reconfiguration unit or of ground telecommand (through OBDH for instance).

Another way of measurement is the use of current probe, which does not degrade the power supply of the circuit and provides a galvanic isolation. The inductor L detects the current flux variation by transforming it in voltage level variation, as shown in the following figures.

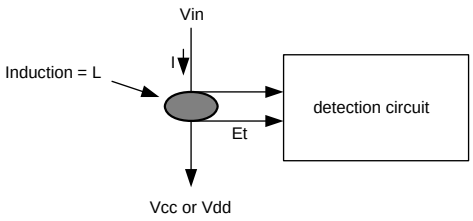

figure 6.1.2–2 – principle of current probe design

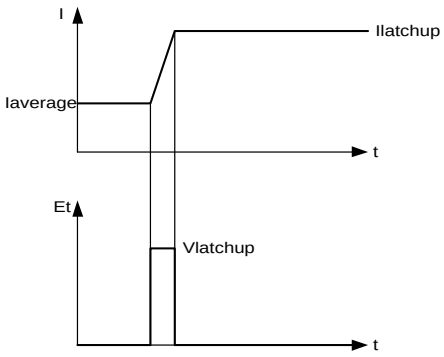

figure 6.1.2–3 – Voltage/Current relationship

The voltage on the inductor is Et = L. $\frac{dI}{dt}$. It can be amplified and integrated before its comparison with a threshold value, in order to activate the protection or not. The value L of the inductor depends on the latchup current and the average current.

ANTI-LATCHUP FUNCTION FOR COMPONENTS WITH LATCHUP CURRENT GREATER THAN AVERAGE CURRENT

For components with latchup current at least 4 to 5 times greater than average current (i.e. average current ≤ 40 mA : DRAM memories, for example), anti-latchup function can be:

- a simple resistor for current limitation (with no indication flag in case of latchup detection)

- a transistor used as a comparator with a current probe made with a resistor or an inductor. In case of latchup detection, the transistor is passing and its collector voltage is used to switch off the component-to-protect supply. An external command has to be sent to supply the component again

6.1.3 Anti-latchup function for components with latchup current and average current of similar category

For devices having latchup current greater than average current but not so far (between 2 to 4 times), a rather simple protection can be designed. This category concerns components with an average current between 40 mA and 100 mA (such as ASICs and microprocessors).

The measurement of the latchup can be similar to the previous case. A first solution based on a resistor can be put in the supply voltage line to detect the latchup overcurrent. The detection circuit compares the voltage on the resistor to the latchup voltage threshold. For this second category of circuits to protect, a voltage comparator circuit shall be used instead of a transistor circuit, because the transistor is not able to detect so low current variation. An example of such an anti-latchup circuit is shown below.

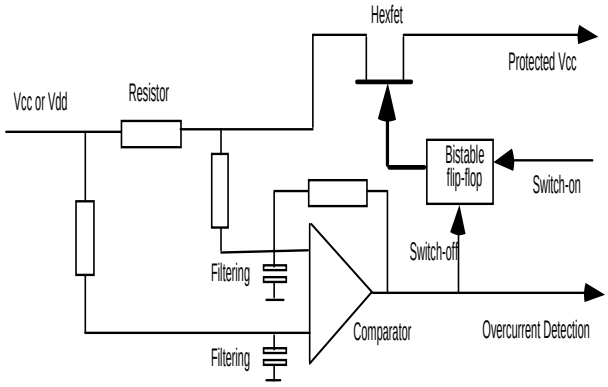

Figure 6.1.3–1 - synoptic of anti-latchup circuit for components with latchup current and average current of similar category

The circuit shown in figure 6.1.3–1 is built around a comparator which senses the current of the Vcc power supply line on the serial resistor. The current is filtered by 2 resistor/capacitor 1st order filters in each branch of the differential measurement to get the average current of the sensitive component. This filtering is necessary in this case because the short duration peak current can be of the same order of magnitude order as the latch-up current and could trigger the comparator.

At power ON, the comparator is in a predefined no latchup state and the flip-flop is reset. Upon a latchup, the current increases and the comparator output state changes triggering the flip-flop to switch-off the circuit. This flip-flop registers the latchup state, since after switching off the latchup state disappears. It also delivers an overcurrent flag which can be used to warn a local processing or to flag in the telemetry frame. Then, an ON command can be sent to supply the component again. This command is under control of a local software or ground telecommand (through OBDH for instance).

Another way of measurement of the latchup current is the use of a current probe, which does not degrade the power supply of the circuit and provides a galvanic isolation. The inductor L detects the current flux variation by transforming it in voltage level variation, as shown in the figures 6.1.2–2 and 6.1.2–3.

ANTI-LATCHUP FUNCTION FOR COMPONENTS WITH LATCHUP CURRENT AND AVERAGE CURRENT OF SIMILAR CATEGORY

For components with latchup current greater than average current but not so far (between 2 to 4 times) (i.e. average current between 40 mA and 100 mA : ASICs, microprocessors for example), anti-latchup function can be:

- a comparator with a current sensor made with a resistor or an inductor. The sensed current shall be filtered to get the average current of the sensitive component. In case of latchup detection, the component-to-protect is switched off. An external command has to be sent to supply the component again.

6.1.4 Anti-latchup function for components with latchup current similar or lower than average current

For devices having latchup current similar or lower than average current (less than 2 times), a protection design solution is not possible by only monitoring the supply current. A more sophisticated design shall take into account operational modes of the circuit to protect. The protection function will be hard to design and will be sensitive to other disturbances such as common mode, overshoot on input signals, etc… In some cases, a protection design solution is not possible. This category concerns components with an average current greater or equal than 100 mA.

In this case, it is not possible to detect the latch-up current by direct electrical measures as before: the current cannot obviously be measured through a resistor because the average current is too important. The only way is to measure it with an inductor as proposed on the figure 6.1.2-2 to maintain the nominal voltage on the circuit.

The average currents can be less or greater than the latchup current, depending on the various operation mode of the circuit to protect. It is not possible in this case to have a steady state current. It could be possible to synchronise the operation of the circuit with the measurement of the current. Latchup detection shall be enabled during the standby mode of the circuit to protect (when the average current is less than the latchup current). Operation modes with average currents greater than latchup currents shall be limited in duration as far as possible, otherwise the circuit could be destroyed (if these modes last more than the burnin delay of the latchup event, i.e. a few hundreds of microseconds).

The following architectural design could apply to make the detection and the protection:

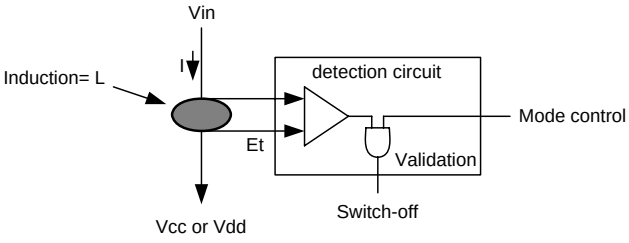

Figure 6.1.4-1 - Anti latch-up synoptic for components with latchup current similar or lower than average current

The mode control inputs can be given by the circuit itself, depending thus on the behaviour of the circuit upon latchup. A better solution is to generate the mode control inputs from a specific hardware, not disturbed by the latchup effect, which controls the sensitive circuit.

ANTI-LATCHUP FUNCTION FOR COMPONENTS WITH LATCHUP CURRENT SIMILAR OR LOWER THAN AVERAGE CURRENT

For components with latchup current similar or lower than average current (less than 2 times) (i.e. average current greater than 100 mA), an anti-latchup function will be hard to design and sensitive to other disturbances such as common mode, overshoot on input signals. Latch-up current cannot be directly measured with a resistor, an inductor shall be used. Latch-up detection shall be enabled only when average current is less than latchup current (mode control input is required).

6.2 ANTILATCHUP DESIGN RULES FOR INTEGRATED DEVICES

6.2.1 Technological countermeasures against latchup

At technological level, ASIC or components manufacturers use different proprietary techniques to diminish the latchup sensitivity of integrated devices. Two of these techniques are briefly described hereafter, but they cannot apply to the use of standard ASIC technology:

- The "guard ring" technique consists in implementing a N well around the MOSFET and supplying it through this well, in order to introduce parasitic transistors and reducing the substrate resistance. These additional rings are shorted to Vcc or ground. This solution has been used by NTT [SHIO], NS [WAK], RCA [DENN] and many other manufacturers.
- Another technique consists in reducing the gain of at least one of the transistors which build the SCR. The SCR exists if the gain multiplication of the 2 transistors constituting the SCR is larger than 1 [WAK].

These solutions cannot apply to the use of standard ASIC technology.

6.3 USE OF AN EXTERNAL LATCHUP DETECTION AND PROTECTION CIRCUIT

Another way to protect sensitive devices against latchup is to implement an additional chip performing the latchup detection function outside the component to protect. The name of this component is LUDPC (LatchUp Detection and Protection Circuit).

The LUDPC is a chip which is connected between the 5 V or 3 V power supply and a latchup sensitive components. The function of the LUDPC is to detect an abnormal current flowing into the latchup sensitive components and to switch it off.

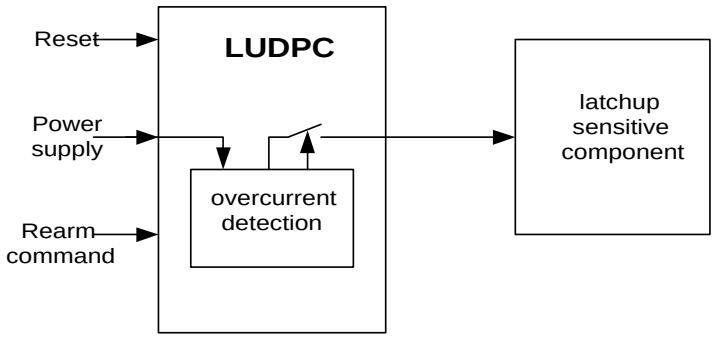

Figure 6.3-1 – LatchUp Detection and Protection Circuit

Req 1 - The average supply current of the latchup sensitive component shall be less than 100 mA (tbc) which corresponds to 0,5 W dissipation under 5 V or 0.3 W under 3 V.

Req 2 – The LUDPC is like a switch and shall have 2 states; a "switch-on" state in which the latchup sensitive component is supplied and a "switch-off" state in which the latchup sensitive components is OFF.

Req 3 – The LUDPC shall detect abnormal supply current greater than 200 mA (tbc).

Req 4 – The LUDPC shall detect the abnormal supply current in less than 100 μs (tbc)

Req 5 – Upon this detection, the LUDPC shall switch the power supply off in less than 100 μs (tbc) and enter in a "switch off" state.

Req 6 – Once, the LUDPC has triggered, it shall stay in the "switch off" state.

Req 7 – Once, the LUDPC is in the "switch off" state it shall wait an external Rearm command to enter back into "switch-on" state.

Req 8 – The LUDPC shall be able to control up to 8 (tbc) latch-up sensitive components

Req 9- Upon Reset signal is active, the LUDPC shall enter the "switch-on" state.

Req 10 – The LUDPC shall provide commands signals in correlation with the 2 states to drive external CMOS transistor switch. The interface is tbd.

7 REFERENCES

REF	Authors	Title	Publication	Date
[ACT97]	ACTEL	Design Techniques for Radiation Hardened FPGAs	ACTEL Application Note	
[ADAM1]	J.H. Adams et al	Cosmic Ray Effects on Microelectronics, Part 1: The Near-Earth Particle Environment	NRL Memorandum Report 4506	August 25, 1981
[ADAM2]	J.H. Adams	Cosmic Ray Effects on MicroElectronics, Part IV	NRL Memorandum Report 5901	December 31, 1986
[ADAM3]	J.H. Adams	The Ionizing Particle Environment Near Earth.	AIAA Aerospace Sciences Meeting	1982
[BESS93]	D. Bessot R. Velazco	Design of SEU-hardened CMOS memory cells: the HIT cell	Proceedings of 2nd European Conference Radiations and their Effects on Devices and Systems (RADECS), p.563-70	1993
[BLA87]	Blaqiere Savaria	Area Overhead Analysis of SEF : a Design Methodology for Tolerating SEU	IEEE Transactions on Nuclear Science, Vol. NS34, No.6, p 1481	1987
[BOUR]	J.Bourrieau	Space Environment	Radecs Short Course	1991,
[BOW]	H. Bowers Zhang Hui	Comparison of Reed-Solomon Codec Implementation	http://infopad.eecs.berkeley.edu/~hui/cs252/rs.html	December 1, 1996
[CAL96]	T. Calin M. Nicolaidis R. Velazco	Upset Hardened Memory Design for Submicron CMOS Technology	IEEE Transactions on Nuclear Science, Vol. NS-43, No. 6, pp. 2874	Dec 1996
[CHEN]	D.L. Chenette W.F. Dietrich	The Solar Flare Heavy Ion Environment for SEU",	IEEE Transactions on Nuclear Science, Vol. NS 31, No.6, pp 1217	Dec 1984
[CHEN84]	C. L. Chen M. Y. Hsiao	Error-correcting codes for semiconductor memory applications: a state-of-the-art review	IBM J. Res. Develop., Vol 28, N°2	March 84
[DAWE]	Dawes, McLean, Robinson, Silver	Hardening semiconductor components against radiation and temperature	Noyes Data Corporation, Park Ridge, New Jersey U.S.A. ISBN. 0-8155-1212-0, pp. 194.	1989
[DENN]	W.J. Dennehy	Non-latching integrated circuit	RCA	

REF	Authors	Title	Publication	Date
[DIE83]	S.E.Diehl J.E.Vinson B.D.Shafer T.M.Mnich	Considerations for single event immune VLSI logic	IEEE Transactions on Nuclear Science , Vol. NS-30, No. 6, pp 4501-4507	Dec 1983
[DOD1]	P.E. Dodd et al	Three-Dimensional Simulation of Charge Collection and Multiple-Bit Upset in Si Devices	IEEE Transactions on Nuclear Science, Vol 41, n°6, pp 2005-2017	Dec 1994
[DOD2]	P.E. Dodd et al	Impact of technology trends on SEU in CMOS SRAMs	IEEE Transactions on Nuclear Science, Vol 43, n°6, pp. 2797-2804	Dec 1996
[FUJI95]	E. Fujiwara M. Kitakami	A class of optimal fixed-byte error protection codes for computer systems	Proceedings on 25th Inter. Conference on fault-tolerant computing, pp 310-319	June 1995
[GAIS96]	Jiri Gaisler	Concurrent error-detection and modular fault-tolerance in an 32-bit processing core for embedded space flight applications	On-board Data Division - European Space Research and Technology Centre	1996
[GAR]	H. Garrett	Radiation Environments within Satellites	IEEE NSREC Short Course, chapter 2	1993
[HAS89]	K.J.Hass R.K.Treece A.E.Giddings	A radiation-hardened 16/32-bit microprocessor	IEEE Transactions on Nuclear Science , Vol. NS-36, No. 6, pp 2252-2257	Dec 1989
[HOF15]	Wolfgang HOFLICH	Using the XC4000 Readback Capability XILINX Application note XAPP 015.000	available at http://www.xilinx.com/xapp/xapp015.pdf	Issue 00 (no date)
[HSI]	C.M. Hsieh et al	A field funneling effect on the collection of alpha-particle-generated carriers in silicon devices	IEEE El. Dev. Lett., n°6, pp 686-693	Jun 1983
[JENN94]	E. Jenn J. Arlat M. Rimén J. Ohlsson J. Karlsson	Fault injection into VHDL models : the MEFISTO tool	Proceedings of IEEE 24th International Symposium on Fault-Tolerant Computing , IEEE, p. 66-75	1994
[JOHA96]	R. Johansson	Two error-detecting and correcting circuits for space applications	Proceedings of FTCS-26, 1996 IEEE	1996

REF	Authors	Title	Publication	Date
[JOHN86]	Richard L.Johnson Jr. Sherra E.Diehl	An improved single event resistive-hardening technique for CMOS static RAMs	IEEE Transactions on Nuclear Science, Vol. NS-33, No. 6, pp 1730-1733	Dec 1986
[JTAG90]	IEEE Computer Society	IEEE standard Test Access Port and Boundary-Scan Architecture	IEEE Std 1149.1	1990
[KATZ94]	R. Katz R. Barto P. McKerracher B. Carkhuff R. Koga	SEU hardening of field programmable gate arrays (FPGAs) for space applications and device characterization	IEEE Transactions on Nuclear Science, Vol. 41, no.6, pt.1 p.2179-86 IEEE (NSREC '94)	1994
[KATZ97]	R. Katz B. Cronquist J.J. Wang R. Koga S. Penzin G. Swift	Radiation Effects on Current Field Programmable Technologies	IEEE Transactions on Nuclear Science, Vol. 44, N° 6	Dec 1997
[KAUL]	N. Kaul	Computer-Aided Estimation of Vulnerability of CMOS VLSI Circuits to SEU	PhD Dissertation, Vanderbilt University	1992
[KERN]	S.E. Kerns L.W. Massengil D.V. Kerns, Jr M.L. Alles	Model for CMOS/SOI single event vulnerability	IEEE Transactions on Nuclear Science, Vol. NS36, N°6, p.4493	Dec 1989
[KNU]	A.R. Knudson A.B. Campbell	Comparison of Experimental Charge Collection Waveforms with PISCES Calculations	IEEE Transactions on Nuclear Science, Vol. 38, n°6, pp. 1540-1545	Dec 1991
[LIN83]	Shu Lin Daniel J. Costello Jr	Error Control Coding: Fundamentals and Applications	Prentice-Hall, Englewood Cliffs	1983
[LIU92]	M. Norley Liu S. Whitacker	Low Power SEU Immune CMOS Memory Circuits	IEEE Transactions on Nuclear Science, Vol. NS-39, No. 6, pp. 1679	Dec 1992
[MAS]	Lloyd Massengil	SEU Modeling and Prediction Techniques	IEEE NSREC Short Course chapter III	1993
[MATT96]	S. Mattsson M. Wiktorson	Radiation Pre-Evaluation of Field Programmable Gate Array (FPGA)	ESA Contract N° 11407/95/NL/CN – Final report	30 august 1996

REF	Authors	Title	Publication	Date
[MUS1]	O. Musseau et al	Analysis of Multiple Bit Upset (MBU) in a CMOS SRAM	IEEE Transactions on Nuclear Science, Vol. 43, n°6, pp 2879-2888	Dec 1996
[MUS2]	O. Musseau	Effet des Ions Lourds energétiques sur les circuits intégrés. Application au cas de circuit MOS, MOS sur isolant et GaAs	Thèse de doctorat en science, University of Paris-Sud, Centre d'Orsay	1991
[NASA]	NASA	SEECA: Single Event Effect Criticality Analysis	http://flick.gsfc.nasa.gov/radhome/papers/seecai.htm	February 15, 1996
[NAVA94]	Z. Navabi N. Cooray R. Liyanage	Using VHDL in parallel Fault Simulation		
[NIRA96]	S. Niranjan J.F. Frenzel	A comparison of fault-tolerant state machine architectures for space-borne electronics	IEEE Transactions on Reliability, vol.45, n°1, p.109-13	
[PET82]	E.L. Petersen P. Shapiro J.H. Adams E.A. Burke	Calculation of cosmic-ray-induced soft upsets and scaling in VLSI devices	IEEE Transactions on Nuclear Science , Vol. NS-29, 2055-2063	1982
[PICK]	J.C. Pickel	Single Event Upset Mechanism and Predictions	IEEE NSREC Short Course, Gatlinburg, IEEE, New York	1983
[REED70]	I.S. Reed, A.C.L Chiang	Coding techniques for failure tolerant counter	IEEE transactions on computer, vol C-19, n° 11	Nov 1970
[ROC88]	Leonard R.Rockett, Jr	An SEU-hardened CMOS data latch	IEEE Transactions on Nuclear Science, Vol. NS-35, No. 6, pp. 1682-1687	Dec 1988
[ROC92]	Leonard R.Rockett, Jr	Simulated SEU hardened scaled CMOS SRAM cell design using gated resistors	IEEE Transactions on Nuclear Science, Vol. NS-39, No. 5, pp 1532-1541	October 1992
[SHIO]	N. Shiono, Y. Sakagawa T. Matsumoto	A 64K SRAM with high immunity from heavy ion latch-up	NTT	
[SIEW82]	Siewiorek D.P., Swarz R.S.	The Theory and Practice of Reliable System Design	Editor : Digital Press	1982
[STAS]	E.G. Stassinopoulos	Radiation Environment of Space	IEEE NSREC Short Course, chapter 1	1990

REF	Authors	Title	Publication	Date
[TONG90]	Po Tong	A 40-MHz Encoder-Decoder Chip generated by a Reed-Solomon Code Compiler	IEEE Custom Integrated Circuits Conference	1990
[WAK]	L. Wakeman	Enhancement eliminate CMOS SCR latch-up	NS	
[WHIT82]	J.B. White Jr.	Fault-tolerant Memory System Architecture for Radiation Induced Errors	IEEE transactions on Aerospace and Electronic Systems, Vol 18, pp. 39-47	January 1982

8 APPENDIXES

8.1 IMPLEMENTATION OF HAMMING ENCODER AND DECODER

8.1.1 SEC or DED Hamming code

The following table shows the check bits count r for a given number of data bits k.

Data bit k	Check bits r	Code word bit n=k+r
k = 1	2	n =3
2 ≤ k ≤ 4	3	5 ≤ n ≤ 7
5 ≤ k ≤ 11	4	9 ≤ n ≤ 15
12 ≤ k ≤ 26	5	17 ≤ n ≤ 31
27 ≤ k ≤ 57	6	33 ≤ n ≤ 63
58 ≤ k ≤ 120	7	65 ≤ n ≤ 127
121 ≤ k ≤ 247	8	129 ≤ n ≤ 255
248 ≤ k ≤ 502	9	257 ≤ n ≤ 511
503 ≤ k ≤ 1013	10	513 ≤ n ≤ 1023

Table 8.1.1-1 - SEC or DED Hamming code : check bits count versus data bit count

The methodology for building a SEC or DED (n,k) Hamming code is the following:

1. The number r of check bits is extracted from the table, according to k (number of data bits to protect).

2. Then the r rows and n columns Parity Check Matrix (PCM) is built :

   Each column corresponds to one bit of the code word, corresponding to a different code made with r bits. The only rule is *to generate linearly independent columns*. Among the 2^r possible codes :

   - one code is reserved for “no error” (all zeroes),
   - k codes are made with the r bits syndrome that should be generated in case of error on one bit of the data word
   - r codes are reserved for the check bits (only one “1”).

Each row corresponds to one check bit : the "1" corresponding to the information bits are XORed or XNORed together in order to build the corresponding check bit (odd or even parity of a subset of the information bits).

One example of such a PCM is shown hereafter for a (15,11) code :

k information bits											r check bits			
10	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**0**	**1**	**2**	**3**
1		1		1		1	1		1	1	1			
1	1			1	1		1	1		1		1		
1	1	1	1				1	1	1				1	
1	1	1	1	1	1	1								1

Table 8.1.1-2 - Parity Check Matrix for SEC or DED Hamming (15,11) code

3. The generation of the check bits is made from the PCM. Each check bit is represented by a row in the PCM, it is built by calculating the parity over the selected information bits. From the previous PCM example:

   Check bit 0 = parity over information bits 10, 8, 6, 4, 3, 1 and 0

   Check bit 1 = parity over information bits 10, 9, 6, 5, 3, 2 and 0

   Check bit 2 = parity over information bits 10, 9, 8, 7, 3, 2 and 1

   Check bit 3 = parity over information bits 10, 9, 8, 7, 6, 5 and 4

   If odd and even parities are required for different check bits of the same code, this is specified in the PCM.

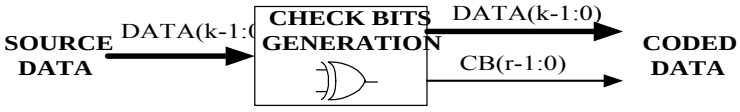

Figure 8.1.1-3 - Check bit generation for Hamming codes

4. The single error correction consists in calculating the r bits of the syndrome and comparing it to the columns of the PCM. As for the check bits, each syndrome bit is represented by a row in the PCM, and it is built by calculating the parity over the selected bits (including the corresponding check bit).

   Syndrome bit 0 = parity over information bits 10, 8, 6, 4, 3, 1, 0 and check bit 0

   Syndrome bit 1 = parity over information bits 10, 9, 6, 5, 3, 2, 0 and check bit 1

   Syndrome bit 2 = parity over information bits 10, 9, 8, 7, 3, 2, 1 and check bit 2

   Syndrome bit 3 = parity over information bits 10, 9, 8, 7, 6, 5, 4 and check bit 3

If odd and even parities are required for different syndrome bits of the same code, this is specified in the PCM.

If the calculated syndrome is equal to all zeroes, there is no error. If the syndrome is equal to one column of the PCM, the bit that corresponds to this column is erroneous and shall be inverted. An error signal can be output by comparing the calculated syndrome with the "all zeroes" value.

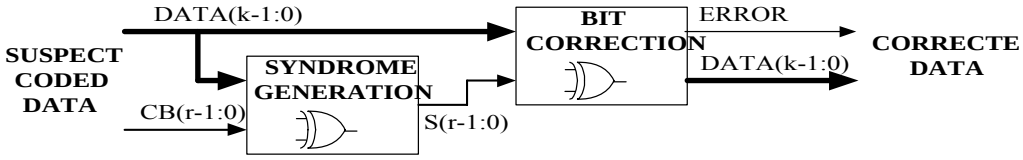

Figure 8.1.1-4 - Single error correction for Hamming codes

There are some tricks for the physical implementation, in order to simplify the logic synthesis, to reduce the critical path and the gate count. Generally, the number of "1" in each row of the PCM is minimised for having less XOR circuitry and thus lower gate count and higher speed.

This method of error detection and correction allows double error detection. If a double error occurs, the syndrome will be different from all zeroes, but there are no means to distinguish between single and double errors and the circuit will correct this error as if it was a single error.

8.1.2 SEC and DED modified Hamming code

The SEC and DED modified Hamming code allows the distinction between the single errors and the double errors, by using an extra check bit.

The following table shows the check bits count r for a given number of data bits k.

Data bit k	Check bits r	Code word bit n=k+r
k = 1	3	n = 4
2 ≤ k ≤ 4	4	6 ≤ n ≤ 8
5 ≤ k ≤ 11	5	10 ≤ n ≤ 16
12 ≤ k ≤ 26	6	18 ≤ n ≤ 32
27 ≤ k ≤ 57	7	34 ≤ n ≤ 64
58 ≤ k ≤ 120	8	66 ≤ n ≤ 128
121 ≤ k ≤ 247	9	130 ≤ n ≤ 256
248 ≤ k ≤ 502	10	258 ≤ n ≤ 512
503 ≤ k ≤ 1013	11	514 ≤ n ≤ 1024

Table 8.1.2-1 - SEC and DED modified Hamming code : check bits count versus data bit count

The methodology for building a SEC and DED (n,k) modified Hamming code is almost the same as for the SEC or DED Hamming code:

1. The number r of check bits is extracted from the table, according to k (number of data bits to protect).

2. Then the r rows and n columns Parity Check Matrix (PCM) is built:

   Each column corresponds to one bit of the code word, corresponding to a different code made with r bits. The only rule is *to generate linearly independent columns*, and *to have an odd number of "1" in each column (for double error detection)*. Among the 2^r possible codes :

   - one code is reserved for "no error" (all zeroes),
   - k codes are made with the r bits syndrome that should be generated in case of error on one bit of the data word. Choosing syndromes with odd number of "1" guarantees that a double error will be detected.
   - r codes are reserved for the check bits (only one "1").

Each row corresponds to one check bit : the "1" corresponding to the information bits are XORed or XNORed together in order to build the corresponding check bit (odd or even parity of a subset of the information bits).

In the following example, the PCM of the MA31755 EDAC from Gec Plessey Semiconductors ((22,16) modified Hamming code), it can be seen that each column has an odd number of "1".

Synd	**Parity**	**Information bits coef. (h_{ij})**																**Check bits coef.**					
bits	**computation**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	**0**	**1**	**2**	**3**	**4**	**5**
0	even			1	1				1	1	1	1	1				1	1					
1	even		1			1	1	1	1		1		1			1			1				
2	odd	1			1			1				1		1	1	1	1			1			
3	odd		1	1			1						1	1	1	1	1				1		
4	even	1				1	1	1	1	1		1			1							1	
5	even	1	1	1	1	1				1	1			1									1

Table 8.1.2-2 Parity Check Matrix for SEC and DED modified Hamming (22,16) code

3. The generation of the check bits is made from the PCM. Each check bit is represented by a row in the PCM, it is built by calculating the parity over the selected information bits. From the previous PCM example:

Check bit 0 = even parity over information bits 13, 12, 8, 7, 6, 5, 4 and 0

Check bit 1 = even parity over information bits 14, 11, 10, 9, 8, 6, 4 and 1

Check bit 2 = odd parity over information bits 15, 12, 9, 5, 3, 2, 1 and 0

Check bit 3 = odd parity over information bits 14, 13, 10, 4, 3, 2, 1 and 0

Check bit 4 = even parity over information bits 15, 11, 10, 9, 8, 7, 5 and 2

Check bit 5 = even parity over information bits 15, 14, 13, 12, 11, 7, 6 and 3

4. The single error correction consists in calculating the r bits of the syndrome and comparing it to the columns of the PCM. As for the check bits, each syndrome bit is represented by a row in the PCM, and it is built by calculating the parity over the selected bits (including the corresponding check bit).

   Syndrome bit 0 = even parity over information bits 13, 12, 8, 7, 6, 5, 4, 0 and check bit 0

   Syndrome bit 1 = even parity over information bits 14, 11, 10, 9, 8, 6, 4, 1 and check bit 1

   Syndrome bit 2 = odd parity over information bits 15, 12, 9, 5, 3, 2, 1, 0 and check bit 2

   Syndrome bit 3 = odd parity over information bits 14, 13, 10, 4, 3, 2, 1, 0 and check bit 3

   Syndrome bit 4 = even parity over information bits 15, 11, 10, 9, 8, 7, 5, 2 and check bit 4

   Syndrome bit 5 = even parity over information bits 15, 14, 13, 12, 11, 7, 6, 3 and check bit 5

If the calculated syndrome is equal to all zeroes, there is no error.

If the syndrome is equal to one column of the PCM, the bit that corresponds to this column is erroneous and shall be inverted.

In the other cases, if the syndrome has an even number of “1”, a double error is detected.

It is possible to output 2 error signals, one for the single corrected errors and the other for the double uncorrected errors.

8.2 IMPLEMENTATION OF REED-SOLOMON ENCODER AND DECODER

8.2.1 Encoder Implementation

A typical architecture of a Reed-Solomon encoder is depicted in Figure 8.2.1-1. It is based on the division of the input polynomial by the coefficient of the polynomial generator. It is mainly composed of:

- an n-stage 8 bit shift register
- n sets of 8 xor gates to make the additions in the finite field
- n 8 bits multipliers in the finite field

The message is input in the shift register, the first k bytes of the output are the same as the input, the last n CRC bytes are output from the shift registers after the computation of the first k bytes.

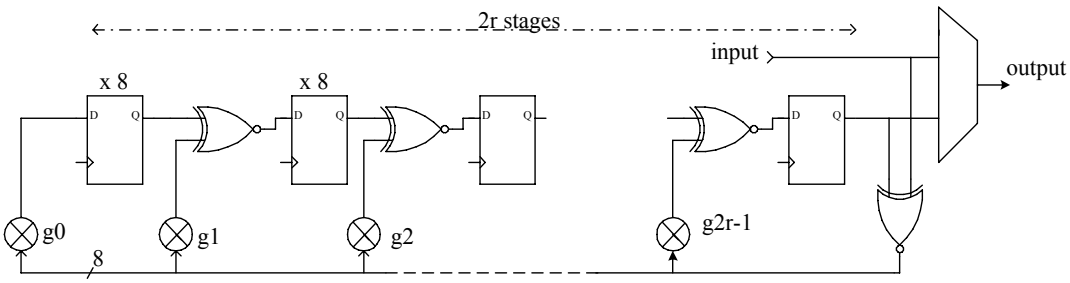

Figure 8.2.1-1 Implementation of a Reed-Solomon Encoder

The following table provides practical results of implementations of Reed-Solomon decoders in FPGA or ASICs ([BOW], ESA and MMS sources). As it can be seen, a Reed-Solomon encoder can be easily included in a space ASIC manufactured in a 1 or 0.6 µm technology. It is even compliant with an ACTEL implementation, if the number of stage (i.e. errors to correct) is reduced and if the clock frequency is limited.

The complexity of the fixed codeword encoder in terms of gates is generally linear in the number of check bits (or errors to correct), and independent in n the code length.

Code	Hardware	Gates/module	Rate	reference
N=255, 4 errors	1 ACTEL 1020	448 of 548 modules	?	Joe Keith - Lockheed
N=255, 8 errors	ASIC LSI 1µm	2000 gates	40 Mbyte/s	LSI design [TONG90] (commercial conditions)
N=255,16 errors	ASIC MHS 1 µm	4900 gates	20 Mbyte/s	MMS design (space RT conditions)
N=255,16 errors with interleave of 1 to 5	RESCUE chip	about 5000 gates	2.5 Mbyte/s	ESA design (space conditions)

Table 8.2.1-2 Hardware implementation of Reed-Solomon encoders

The design of a fixed codeword length Reed-Solomon encoder without interleave is easily achievable. A basic knowledge of Gallois finite field and cyclic code is necessary. A pre-optimisation of the constant finite field multipliers is recommended to help the synthesis tool. It can be made with a software as Mathematica that can directly generate a VHDL code easier to synthesise.

The implementation of the interleave function increases largely the gate count. The RESCUE chip includes an interleave of 1 to 5. This limit was chosen to design a chip having a limited gate count. An interleave of 8 would have doubled the gate count. Nevertheless, 2 RESCUE chips can be connected to make it, without any additional logic.

8.2.2 Decoder Implementation

The design of a Reed-Solomon decoder is much more complex than the design of the encoder. It requires a good knowledge of the Reed-Solomon theory. Many authors studied architectural implementations of the function in order to reduce gate count or latency, or increase throughput. A possible implementation is provided in figure 8.2.2-1. A Reed-Solomon decoder contains always RAM to store the incoming frames during the error computation. When error location is found out, the previously stored frame is read and corrected for being output.

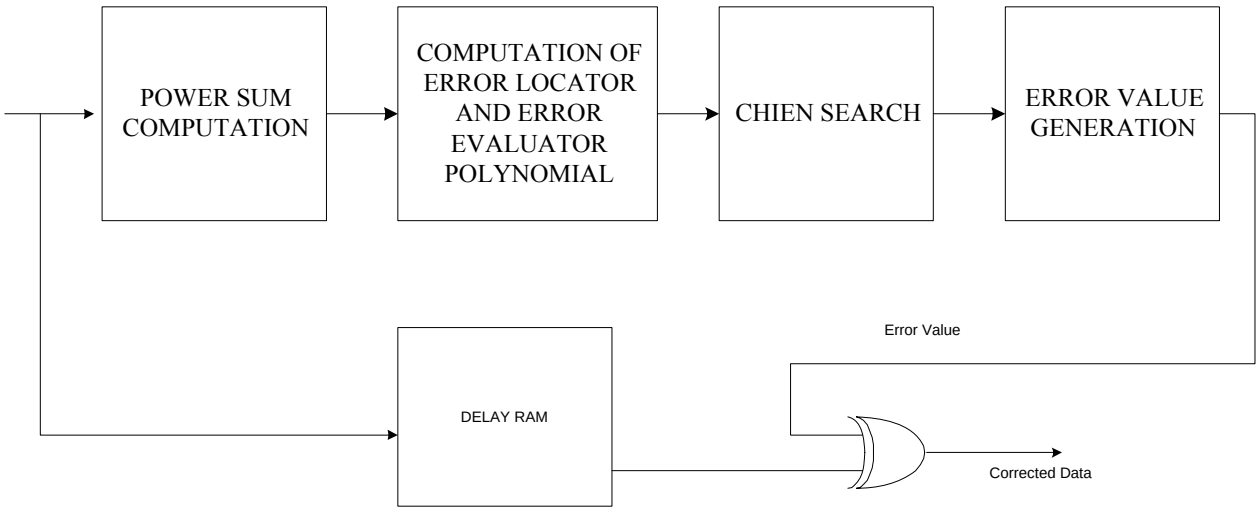

Figure 8.2.2-1 - Possible architecture of a Reed-Solomon decoder

The following table provides the characteristics of Reed-Solomon decoders designed in the industry ([BOW], ESA, MENTOR and MMS sources).

Code	Hardware	Gates/module	Rate	reference
N=255, 8 errors	ASIC LSI 1um	18 000 gates 3K RAM 4K ROM	40 Mbytes/s	LSI design [TONG90] (commercial conditions)
N=255, 8 errors	MHS Core	18000 gates 5000 SRAM bits	25 Mbyte/s target space	MHS design [VANT96]
N=255,16 errors	ASIC MHS 1 µm	30 000 gates 6 256x8 RAM	20 Mbyte/s	MMS design (space RT conditions)
N=255,8 errors	INVENTRA I.P. Core	18 855 gates +RAM and ROM	20 Mbyte/s	Inventra Soft Cores Data Book

Table 8.2.2-2 Hardware implementation of Reed-Solomon decoders

As it can be seen in table 8.2.2-2, a Reed-Solomon decoder able to correct 8 errors has a complexity of about 20 000 gates plus RAM and ROM. If this function has to be included in a complex chip for example for AOCS control, designing it will require manpower and tools that will not be available for the rest of the ASIC. The use of Intellectual Property may be a good opportunity in this case, since this function is offered by many IP vendors (only MENTOR/Inventra is mentioned in table 8.2.2-2), and since this function has a limited number of I/O and can be functionally tested by a limited set of vectors.

8.3 IMPLEMENTATION OF A REED-MULLER CODE PROTECTED COUNTER

Coding technique can be applied in order to implement SEU tolerant counters. Herebelow is described an implementation based on Reed-Muller codes and derived from [REED70]. It uses the following principles:

- the n-bits count words can be encoded in a (2n, n) linear code (n useful bits and n check bits); the code is chosen in such a way that the parity check matrix has the following form:

$$H = \begin{bmatrix} 1 & 1 & 0 & 0 & \cdots & 0 & 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 1 & 0 & \cdots & 0 & 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 1 & \cdots & 0 & 0 & 0 & 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & 0 & 0 & 0 & \cdots & 0 & 1 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

- for a code-word ($A_1, A_2, A_3, \ldots A_n, B_1, B_2, \ldots B_n$), where the A_i are the useful bits of the counter, and the B_i are the check bits, the check bits can be computed from the matrix:

$$\begin{array}{lll} B_1 = A_1 \oplus A_2 & & A_1 = A_2 \oplus B_1 = A_n \oplus B_n \\ B_2 = A_2 \oplus A_3 & & A_2 = A_3 \oplus B_2 = A_1 \oplus B_1 \\ B_3 = A_3 \oplus A_4 & \text{which is equivalent to} & A_3 = A_4 \oplus B_3 = A_2 \oplus B_2 \\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots\ \ldots & & \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ B_n = A_n \oplus A_1 & & A_n = A_1 \oplus B_n = A_{n-1} \oplus B_{n-1} \end{array}$$

- These relationships show that each useful bit can be issued from three different combinations which give the same value; for instance the first counter bit is either A_1, or $A_2 \oplus B_1$, or $A_n \oplus B_n$. If we state that the output bit of the counter results from a majority voting of the three values, and if one of the values gets wrong due to a SEU, the output bit is kept unchanged.
- Figure 8.3-1 shows an implementation of such a n-bit counter. In this example, the input carry is a strobe which has the pulse width of the general clock (CK), but which repetition rate corresponds to the input frequency of the counter (it can be much smaller than the general clock). The A_i and B_i signals are stored in D Flip-flops; MAJ modules are majority voting modules. The outputs of the different stages are A'_1, A'_2, ..., A'_n signals.

It must be noticed that the A_i and B_i values must be consistent with the above relations within the same clock period. Therefore the logic combinations at the input of the B_i flip-flops must anticipate by one clock period the correct value of the B_i. The combinatorial logic, which sets the correct Bi value before the subsequent clock edge, is given by the following relations:

$$B_1 = B'_1 \oplus not(A'_1)$$
$$B_2 = B'_2 \oplus (A'_1 \cdot not(A'_2))$$
$$B_3 = B'_3 \oplus (A'_1 \cdot A'_2 \cdot not(A'_3))$$
$$B_{n-1} = B'_{n-1} \oplus (A'_1 \cdot A'_2 \cdot \ldots \cdot A'_{n-2} \cdot not(A'_{n-1}))$$
$$\ldots\ldots$$
$$B_n = B'_n \oplus not(A'_1 \cdot A'_2 \cdot \ldots \cdot A'_n)$$

where the B'_i and A'_i are the flip-flops output before the clock edge.

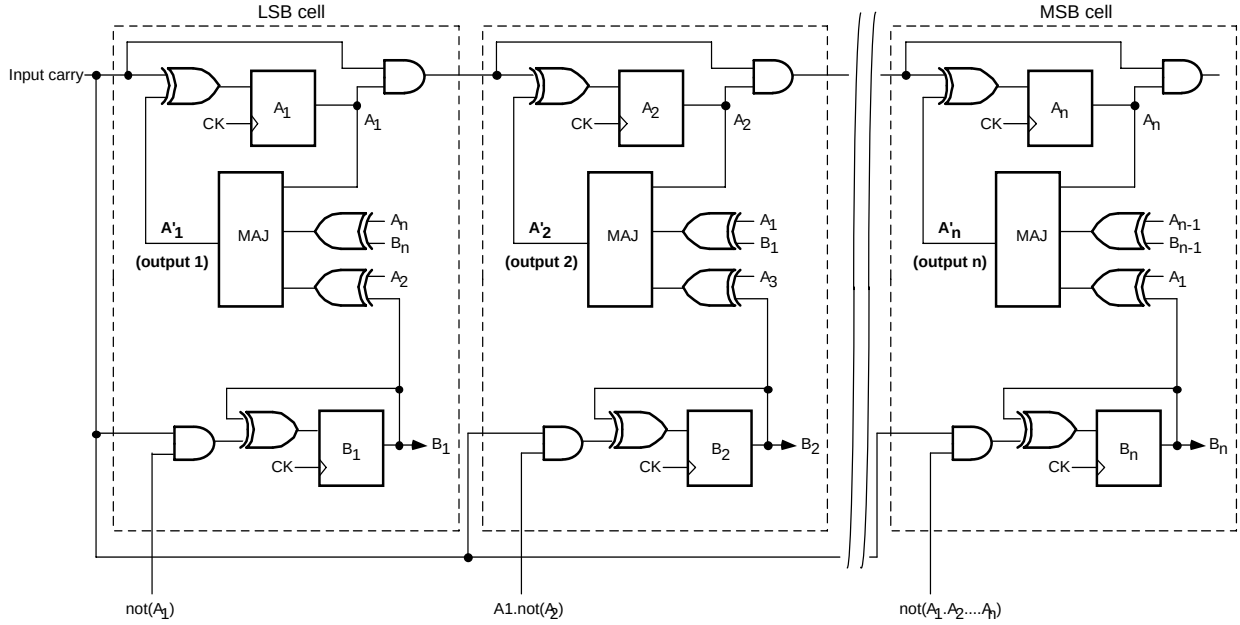

Figure 8.3-1: SEU tolerant n-bit counter by coding

8.4 IMPLEMENTATION OF RESISTIVE HARDENING FOR STORAGE CELLS

The figure below shows typical examples for a hardened RAM cell, a D-latch, a NOR latch and a NAND [DIE83]:

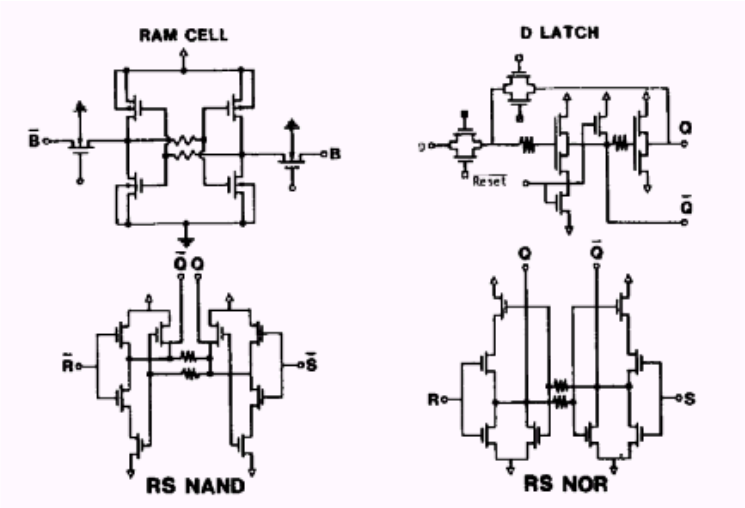

Figure 8.4-1 - Resistive hardening of common storage cells [DIE83]

Figure 8.4-2 shows an alternative resistor hardened RAM cell. A complete discussion of resistor values to be used and the effects on critical charge and local write time can be found in reference [JOHN86]. By configuring this cell correctly, good SEU immunity can be reached with a better performance for write access than the previous configuration.

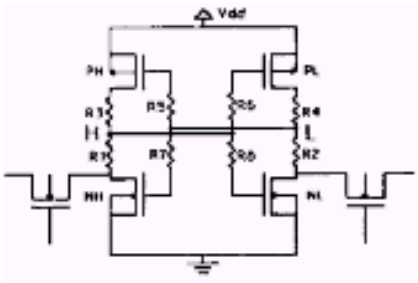

Figure 8.4-2: Isolated CMOS static RAM cell [JOHN86]

Latches can also be hardened using only one resistor in the feedback loop [HAS89]. This way the propagation delay of at least one of the latch outputs remains unaffected which is interesting for speed performance. However setup time and minimum clock pulse width remain affected by the presence of the resistor. Examples for a D-latch and a RS-latch are shown below.

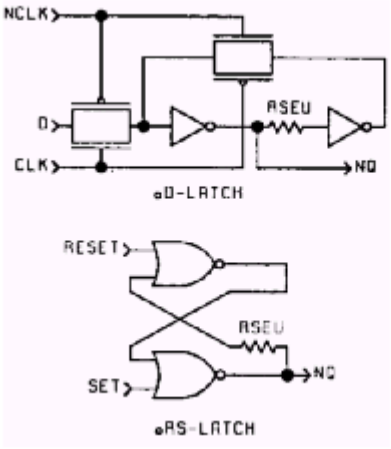

Figure 8.4-3 – SEU hardened latches using only one resistor [HAS89]

An alternative resistive hardening technique consists in the use of gated (or active) resistors instead of passive resistors [ROC92]. The gated resistors, clocked by the data latch write clock, provide the required SEU hardness with minimal speed and area penalty.

Gated resistors are actively clocked polysilicon resistors that are used to provide SEU hardness. They are placed in the cross-coupled segments of SRAM cells, similar to designs using passive polysilicon resistors. The high-resistance OFF-state of the gated resistors protects the stored cell data from SEUs. However, during write-cell cycles, the gated resistors are clocked into a low-resistance ON-state by the word-line clock signal. Thus the fast write response of the cell is preserved. Gated resistor hardening imposes no circuit density penalties because the resistors can be processed using a second level of polysilicon separated by a thin inter-level thermal oxide layer from the first level of polysilicon. Some additional processing steps are required.

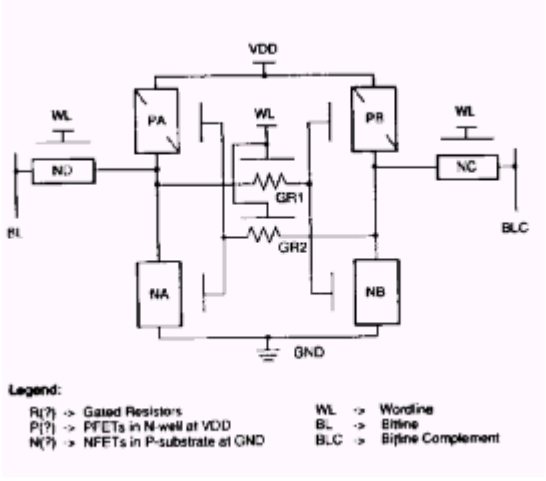

Figure 8.4-4 – gated resistor hardened CMOS SRAM cell [ROC92]

8.5 IMPLEMENTATION OF GLITCH FILTERING FOR STORAGE CELLS

In order to have a latch design to be insensitive to transients of maximum width D, a setup time T_{su} is to be respected that is equal to D multiplied by a security margin S, with S > 2 [BLA87]. The transients may occur randomly in time and S is a worst-case value valid for all possible occurrences of the transient during the time period T_{su}.

In order to implement a glitch-filter in the memory device, resistive or capacitive hardening techniques are used. Also dedicated implementations that combine 'internal' and 'external' sensitivity have been proposed in literature. These implementations try to minimise S at the cost of area or vice versa [BLA87]. The figure below shows a solution with S = 3.48 and an area overhead of 50% compared to a standard latch (the values for W/L are indicated in the figure for a 5 μm process but can directly be scaled to a 0.5μm process by dividing all dimensions by 10).

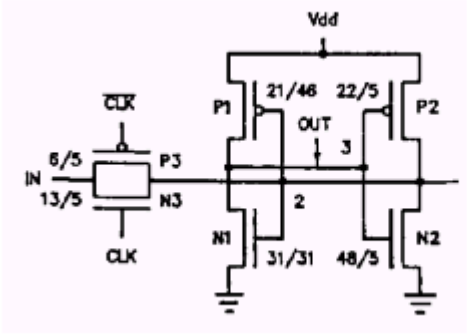

Figure 8.5-1 – latch with embedded glitch-filter [BLA87]

As transients are expected to last up to 1 ns, a setup time of 3.48 ns has to be respected for this particular latch to be glitch-insensitive. This contrasts with setup times of nowadays technologies that are in the range of 0.1 ns to 0.5 ns.

Another implementation that handles at the same time 'internal' and 'external' sensitivity is shown below in Figure 8.5-2. The 'internal' sensitivity is handled here by the transistor hardening technique which, in practice, ensures an almost 100% insensitivity to SEU (refer to section 5.3.1.5). In order to handle the 'external' sensitivity of the cell, transistor dimensions are chosen in such a way that the cell exhibits glitch-filtering characteristics. In concreto, for this type of cell, the glitch-filtering characteristics can be determined mainly by tailoring the ratio of the drive strengths of the transmission gate on one hand and the "weak" inverter stage of the memory loop on the other hand. It is clear that a 'strong' transmission gate combined with a 'weak' 'weak inverter stage' results in a rather glitch-sensitive configuration whereas the opposite will result in a cell with fairly good glitch-filtering characteristics. The exact implementation must be determined by simulation.

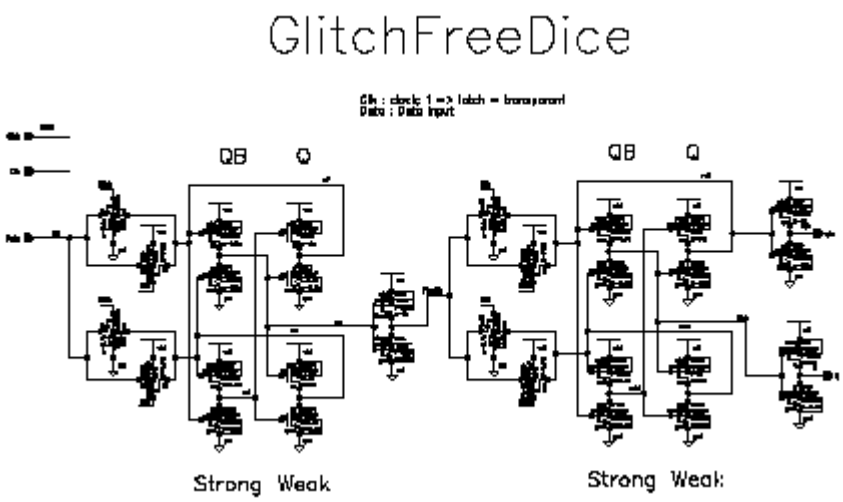

Figure 8.5-2: the DICE register with embedded glitch-filter

T_{su} is to be determined by extensive iterative simulations. Hereby a waveform is applied to the data input of the memory device that:

- exactly becomes stable at T_{su} before the relevant clock edge
- is corrupted by a transient of width D during T_{su} . Independently of the arrival time of the transient within the T_{su} period, correct behaviour (no transient latching) of the latch must be obtained.

The concept of 'extended setup time' is generally applicable for all types of memory devices hardened with all kinds of techniques. If, for example, a cell has been 'internally' hardened using the resistive technique, T_{su} can be determined by simulation to verify the glitch-filtering properties of that cell. If necessary, the cell must be further adapted to meet the 1 ns glitch constraint.

8.6 IMPLEMENTATION OF TRANSISTOR HARDENING FOR STORAGE CELLS

8.6.1 HIT cell

The HIT (Heavy Ion Tolerant cell) cell has been designed for fast recovery after upset, low static power consumption and no speed performance degradation. It is composed of 12 transistors organized as two storage structures interconnected by feedback paths. For read/write a single phase clock CK is needed and differential data inputs D and D'. The schematic of the HIT cell is depicted in the figure below.

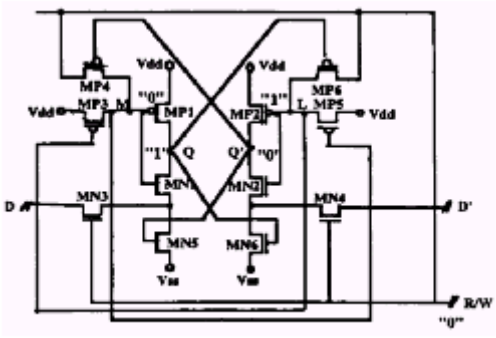

Figure 8.6.1-1 - the basic HIT memory cell [BESS93]

The HIT cell is immune for hits on single nodes Q, Q', L and M and for a simultaneous hit on the tuple Q and Q'. A simultaneous hit on other combinations of the nodes Q, Q', L and M causes the cell to upset.

For a detailed discussion of the HIT cell the reader is referred to the literature [BESS93]. Below, normal operation, upset recovery and hardening are discussed in short.

Read Operation

For read operation the data lines D and D' must first be precharged to VDD. As the read/write signal R/W goes active (high), D remains at VDD while D' is pulled down by node Q' via transistors MN4 and MN6 to GND (starting from the logical state indicated in the circuit diagram).

Write operation

To write the HIT cell the read/write signal R/W must be high while new opposite data values are presented to the cell on the D & D' inputs. Starting from the logical state indicated in the circuit diagram and applying values 0 and 1 to respectively D and D', first transistor MP4 will pull node M to 1. Because of this, input D can freely pull down node Q to 0. MN6 now is turned off and D' is connected to Q', pulling it to 1. MN5 turns on, reinforcing the state of Q and bringing node M to high impedance. When R/W is set to inactive (0) again, node L becomes 0 reinforcing Q' to 1 and pulling M to 1.

Hardening

Four nodes (Q, Q', L, M) are used to store data in the cell. To cope with SEU, specific transistor ratios have been used. In the schematic above it is essential for upset recovery on nodes Q and Q' that the drive strength of MP3 and MP5 is higher than the drive strength of MP4 and MP6 respectively.

Upset recovery

As an illustration, the recovery from a hit on node Q is discussed. Given the initial state of the cell as indicated in the figure, a hit on the drain of MN1 causes a voltage drop on node Q. Because of this MN6 becomes high-impedant (no influence on logical state of cell) and MP6 turns ON. Because MP5 has a bigger drive strength that MP6 the logical state of node L remains at '1', preserving the state of node Q'. As node M remains unaffected at '0', MP1 will restore node Q to the original logic state '1'.

Other topologies based on the same principles as the HIT cell have been proposed by Rockett and Liu (further referred to as ROCKETT cell and LIU cell). In order to be able to make a comparison between HIT, ROCKETT, LIU cell and an unhardened cell UMC, these cells have been designed in a double 1.2 μ N-well CMOS process and their characteristics have been evaluated and compared for static power consumption, propagation delay, area and SEU sensitivity. For comparable propagation delays, static power consumption and error-rate (except UMC cell of course) the cell area of the HIT cell is only 7% bigger than the UMC cell area whereas ROCKETT and LIU cells are 52% and 77% bigger than the UMC cell. Below the schematic diagrams for ROCKETT and LIU cell are shown. For a detailed discussion of their functionality the reader is referred to the literature [ROC88], [LIU92].

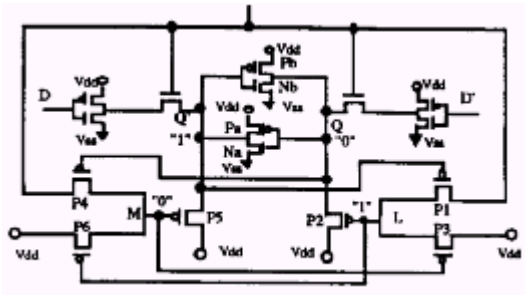

Figure 8.6.1–2 - The ROCKETT memory cell [ROC88]

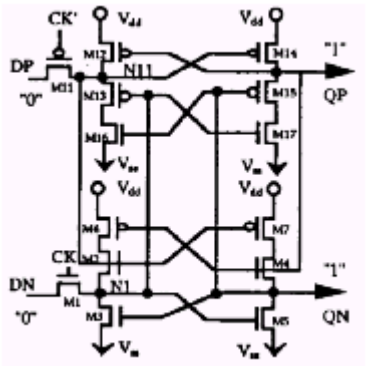

Figure 8.6.1-3 - The LIU memory cell [LIU92]

The HIT, ROCKETT and LIU cells can be used to construct latches and flip-flops if inverters for clock and/or data are added. The figure below shows the schematic to construct a flip-flop out of the HIT cell.

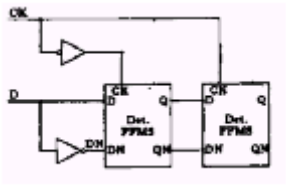

Figure 8.6.1-4 - SEU hardened D-flip-flop starting from HIT cell

8.6.2 DICE cell

The DICE (Dual Interlocked Storage Cell) cell is also conceived according to the general principles defined in paragraph 5.3.1.5 except that no ratioing of transistors is performed and that no degraded logic levels occur. A new principle, 'dual node feedback control', has been applied. The design of the cell relies entirely on feedback loops within a dedicated latch architecture.

The DICE cell is immune to hits on single nodes. If two simultaneously sensitive nodes of the cell, which store the *same* logic state (i.e., either nodes X0-X2 or nodes X1-X3) are hit by one ion that affects both sensitive nodes, the immunity is lost and the cell is upset.

For a detailed discussion of the DICE cell the reader is referred to the literature [CAL96].

Below, normal operation and upset recovery and possible implementations are discussed in short. Figure 8.6.2-1 shows the principle of the Dual Interlocked Storage Cell.

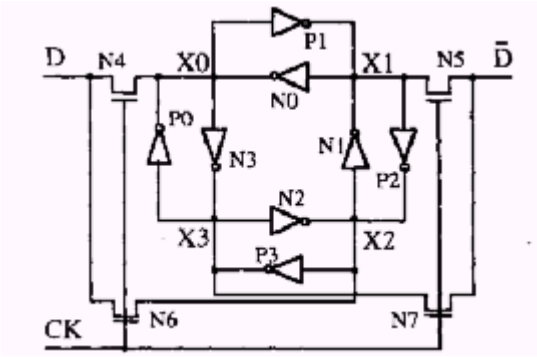

Figure 8.6.2–1 - Principle of the Dual Interlocked Storage Cell [CAL96]

Normal operation

The cell uses a 4-node redundant structure. Two horizontal conventional cross-coupled inverter latch structures are connected by two vertical conventional cross-coupled inverter latch structures. The four nodes $X_0 \ldots X_3$ store the data as two pairs of complementary values (i.e. 0101 or 1010). These nodes are simultaneously accessed for write and read operation. The principle of 'dual node feedback control' means that the value of each node is controlled by the adjacent nodes located on the opposite diagonal.

For instance, the value of node X_1 is controlled by the values of nodes X_0 and X_2. In practice the inverters in the diagram are realised by single nmos or pmos transistors. Suppose that the cell is in state $X_0 \ldots X_3 = 0101$

then the horizontal pairs of transistors are conducting, determining the logic values of the four nodes. At the same time the vertical transistor pairs are turned off, performing a 'interlock' function by isolating the two horizontal latches. For the opposite state of the cell the roles of the transistor pairs are switched.

Upset recovery

An upset at a single node X_i affects at most 2 nodes, the other 2 nodes remaining unaffected. If we assume, for instance, the state $X_0 \ldots X_3$ to be 0101 and node X1 to be hit by a negative upset pulse, only nodes X1 and X2 will be affected. Indeed the negative pulse on X_1 turns on transistor P2 which results in a positive transition of node X2. This positive transition on X2 however, is not propagated any more to node X3 because the only effect is that P3 turns off (temporarily). Because of the same reason, the negative pulse on X1 has neither an effect on node X0 On the other side. The perturbations on nodes X1 and X2 are removed due to the state-reinforcing feedback ensured by the two unaffected nodes X0 and X3. From the explanation above it can be understood that the DICE cell is insensitive to SEU and partially sensitive to double event upsets. Indeed, the cell will upset when simultaneous upsets occur at e.g. nodes X0 and X2 but not when simultaneous upsets occur at e.g. nodes X1 and X2.

Implementations

The DICE cell used as a RAM cell has an overhead of 100% compared to the six-transistor RAM cell. Fabricated in a high interconnect density technology with three metal layers the overhead can be limited to 70%. The circuit diagram is depicted below.

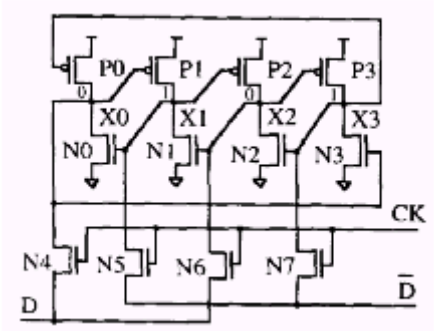

Figure 8.6.2–2 - The DICE memory cell [CAL96]

The DICE cell can also be used to build latches and flip-flops. The figures below show the transistor diagrams for a transmission gate latch and a clocked inverter latch.

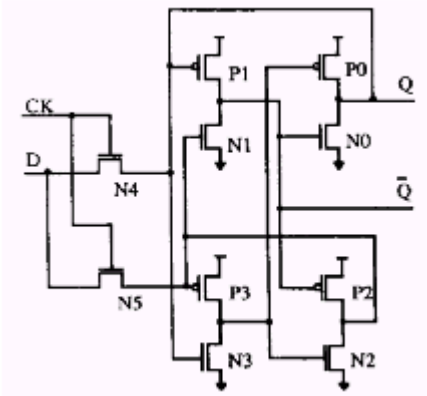

Figure 8.6.2–3 - Transmission gate latch using the DICE cell [CAL96]

N1-P1 and N3-P3 are weak feedback inverters to reduce dynamic power consumption.

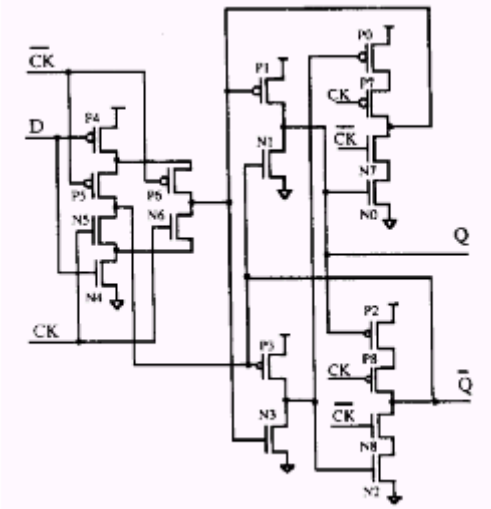

Figure 8.6.2–4 - Clocked inverter latch using the DICE cell [CAL96]

This topology further reduces dynamic power consumption.

---

i NOOA-USAF Space Weather Operations, "Preliminary report and forecast of solar geophysical data", SWO PRF 1079, 7 May 1996.