THE TECHNICAL UNIVERSITY OF ŁÓDŹ
Faculty of Electrical, Electronic, Computer and Control Engineering

Master of Engineering Thesis

DESIGN OF RADIATION TOLERANT INTEGRATED CIRCUITS

Paweł Malinowski

Student's number: 111305

Supervisor:
Grzegorz Jabłoński, PhD

Auxiliary supervisor:
Dariusz Makowski, MSc

Łódź, 2006

1 Introduction **6**
2 Objectives **7**
2.1 Problem description 7
2.2 Previous achievements 8
2.3 Proposed solution 9
3 Radiation effects on electronic devices **10**
3.1 Displacement Damage 11
3.2 Ionizing effects 12
3.3 Single Event Effects 13
3.3.1 Soft Errors 14
3.3.1a Transient Effects 14
3.3.1b Single Event Upsets 16
3.3.1c MBU - Multiple Bit Upset 17
3.3.1d SEFI - Single Event Functional Interrupt 18
3.3.2 Hard Errors 18
3.3.2a SEL - Single Event Latch-Up 18
3.3.2b SES - Single Event Snapback 20
3.3.2c SHE - Single Hard Error 20
3.3.2d SEGR - Single Event Gate Rupture 20
3.3.2e SEBO - Single Event Burn Out 21
4 Mitigation of Radiation Effects **22**
4.1 Hardware Mitigation 22
4.1.1 Process Hardening 23
4.1.1a Wafer Cleaning and Polishing 23
4.1.1b Gate Oxide Growth 23
4.1.1c Oxide Annealing 24
4.1.1d Gate Electrode 24
4.1.1e Silicon On Insulator (SOI) Technology 25
4.1.2 Layout Hardening 25
4.1.2a Enclosed Layout Transistors 26
4.1.2b Guard Rings 28
4.1.2c Trench Isolation 28
4.1.3 System Hardening 29
4.1.3a Parity Bit for Detecting Single Errors 30
4.1.3b Checksum. For Error Detection 31
4.1.3c Cyclic Redundancy Check Codes 31
4.1.3d Hamming Codes 37
4.1.3e Modular Redundancy 42
4.1.3f Partial Replication 45
4.1.3g Periodical Refresh 48
5 Design of a radiation tolerant readout system for a neutron detector .. **49**
5.1 Project Overview 49
5.2 Possible Applications in the Radiation Environment 51
5.3 Used Tools 51
5.4 Readout System Description 54
5.4.1 Design Overview 54

5.4.2 Detector Readout Design.. 55
5.4.3 System Description.. 56
5.5 Design Path... 69
5.5.1 Behavioural Description and Simulations in Aldec Active HDL.............. 70
5.5.2 Synthesis in Cadence BuildGates Physically Knowledgeable Synthesis....... 72
5.5.3 Synthesis in Xilinx ISE version 8.1i... 73
5.5.4 Cadence FirstEncounter Place & Route Environment.......................... 74
5.5.5 Post Layout SDF Simulation.. 83
5.5.6 Cadence Virtuoso Layout Editor.. 84
6 Conclusions and Suggested Improvements.. 86
References... 88
Appendix A Tutorial Presenting Layout Generation from VHDL.................... 91
Appendix B: List of Abbreviations.. 119

Streszczenie

Niniejsza praca magisterska dotyczy projektowania cyfrowych układów scalonych odpornych na promieniowanie. Głównym typem błędów powodowanych przez radiację opisywanych w tej pracy są błędy typu Single Event Effect. Ich efektem może być zmiana stanu tranzystora, co może się objawiać zmianą wartości bitu przechowywanego na przykład w pamięci SRAM lub w przerzutniku. Wraz ze skalowaniem układów i postępem technologii zmniejsza się ładunek potrzebny do wygenerowania takiego błędu. Potrzeba zabezpieczania układów przed promieniowaniem występuje nie tylko dla systemów stosowanych w środowiskach z dużymi dawkami promieniowania, jak w zastosowaniach kosmicznych czy w akceleratorach cząstek elementarnych, ale również w systemach elektronicznych codziennego użytku.

Teoretyczna część niniejszej pracy opisuje główne efekty promieniowania neutronowego na systemy elektroniczne (zaburzenia siatki krystalicznej, efekty jonizacji oraz Single Event Effects). Są tu również przedstawione powszechnie stosowane metody zabezpieczania układów przed efektami radiacji na różnych poziomach projektowania. Opisane są metody uwzględniające zmiany technologiczne, specjalne metody optymalizacji topografii oraz techniki implementowane na poziomie systemowym. Główny nacisk jest położony na trzecią grupę, w której opisano różne metody kodowania. Oprócz tego przedstawione zostały metody podwójnej i potrójnej redundancji, powszechnie stosowane w układach programowalnych oraz techniki odświeżania systemu i częściowej replikacji zasobów w rodzinach układów logicznych.

Część praktyczna niniejszej pracy opisuje pełen proces projektowania układów cyfrowych uodpornionych na promieniowanie na poziomie systemowym na przykładzie systemu sterującego do detektora neutronów opartego na asymetrycznej pamięci SRAM. Projektowany kontroler został opisany przy użyciu języka opisu sprzętu VHDL.

Zastosowany detektor jest asymetryczną pamięcią SRAM zaprojektowaną do przechowywania logicznej '1'. Następnie pamięć ta jest zapisywana wyłącznie wartościami logicznego '0', co powoduje, że jest szczególnie wrażliwa na przestawianie bitów (bit flips) w wyniku promieniowania neutronów. System kontrolujący jest oparty na maszynie stanów, której zadaniem jest odczytywanie zawartości pamięci i zliczanie liczby logicznych '1', które w założeniu są efektem występowania zjawisk Single Event Effects. Następnie suma odczytanych błędów jest wysyłana z kontrolera używając portu szeregowego w ramach standardu EIA-232. Za każdym razem, kiedy w pamięci wystąpią błędy, jest ona zerowana przez system sterujący.

Odporność na zjawiska Single Event Upsets została osiągnieta przez zastosowanie kodów Hamminga do zabezpieczania rejestrów stanów w maszynach stanowych. Oprócz tego wszystkie używane stany zostały zakodowane przy użyciu kodów Graya. Dodatkowo ramki wysyłane z kontrolera są kodowane za pomocą CRC32 (32-bitowy Cyclic Redundancy Check). Skuteczność zastosowanych metod została sprawdzona przez wymuszanie błędów w behawioralnym opisie VHDL. Sprawdzono, że układ wykrywa i poprawia pojedyncze błędy w rejestrach stanów, a maszyny stanowe funkcjonują prawidłowo, jeżeli taki błąd wystąpi. Dowiedziono, że w dziedzinie błędów generowanych w efekcie promieniowania, zamiany pojedynczych bitów sa najczęstsze. Jednak system został również wyposażony w funkcję detekcji podwójnych błędów. W przypadku wystąpienia takiego błędu maszyna stanów powraca do stanu wyjściowego.

W aneksie do niniejszej pracy został zawarty pełen opis ścieżki projektowej wymaganej do wygenerowania layoutu w technologii krzemowej AMS v3.70 0.35 μm. Punktem wyjścia jest opis behawioralny w języku opisu sprzętu VHDL lub Verilog. Do przeprowadzenia całego procesu używane są następujące programy: ALDEC Active HDL do opisu behawioralnego i symulacji, Cadence BuildGates Physically Knowledgeable Synthesis do syntezy, Cadence Encounter do Place&Route oraz Cadence Hit Kit v3.70 do generacji layoutu używając wymienionej technologii CMOS.

1 Introduction

This thesis concerns aspects of designing radiation tolerant Integrated Circuits together with a detailed description of the complete design path from the behavioural description to the generated technological layout of the final device. Some Single Event Upsets mitigation techniques have been presented using an exemplary circuit supposed to work as a readout system for an SRAM based neutron radiation detector. Such system is targeted to environments with very high levels of radiation, like the particle colliders.

A review of radiation effects on electronic devices is given in Chapter 3. Covering atomic displacement damage, ionization effects and Single Event Effects it provides a brief summary of results of radiation on electronic systems.

Chapter 4 describes the most common radiation mitigation techniques. It covers achieving radiation tolerance on different design levels.

A detailed review of the layout generation process given the behavioural description is presented in Chapter 5. The shown design path can be used for any digital circuit implemented in the considered technology, with some changes necessary for the appropriate system.

Chapter 6 are conclusions and summary of the designed system. Furthermore, some suggestions for the next generations of the considered device are described briefly.

Appendix A provides a tutorial with the thorough description of all design steps needed to generate the layout with the given behavioural description in Hardware Description Language. This chapter is supposed to assist in performing the whole design path in AMS v3.70 0.35 µm CMOS process technology.

2 Objectives

The goal of this thesis was to design a radiation tolerant readout system for an SRAM based neutron radiation detector. The device was supposed to be designed as an Application Specific Integrated Circuit (ASIC). Such solution provides the possibility of integration of the readout with the detector and also minimises the device's silicon area usage. The aim was to elaborate the whole design path from the behavioural description in VHDL (Very-High-Speed Integrated Circuit Hardware Description Language), through synthesis to layout generation and post-layout simulations.

This chapter describes the problem, reviews some previous solutions and presents the proposed implementation of the system.

2.1. Problem Description

Malfunctions of electronic devices due to Single Event Effects being an effect of radiation are observed not only in cosmic and airborne equipment, but also in mainstream applications. Together with the progressing integration and scaling of the electronic chips their susceptibility to errors increases. Thus, there is a need for both radiation detection and also for hardening of the designs against radiation.

In particle accelerator locations, as for example in the International Linear Collider (ILC) tunnel, monitoring of radiation is crucial. However, the environment of particle colliders and accelerators is especially tough for electronic circuits, since the radiation doses are extreme. Thus, designing a control system for radiation detectors working in such environments presents a challenge. The readout from the sensor should be performed reliably and provide means for safe transmission of the results. This is done by using some mitigation techniques. To obtain large scale of integration of the readout system at a

reasonable cost, a commercial silicon technology is used. However, a disadvantage of such approach is a limitation of mitigation strategies to the system level only.

2.2 Previous Achievements

Several projects concerning radiaiton mitigation techniques have been described. A very straightforward approach is using Hamming codes to detect and correct faults caused by Single Event Upsets (SEUs), which was presented in many publications. [Lima] proposes a rad-hard version of a 8051 microcontroller, where Hamming codes have been used to protect memory and registers. Such approach was also implemented in this project. Another broadly discussed solution are modular redundancy techniques. Such approach concerns duplicating (DMR, Double Modular Redundancy) or triplicating (TMR, Triple Modular Redundancy) the crucial modules of the system. A comparison between this method and Hamming coding was presented in [Hentschke 02]. A combined solution is described in [Mielczarek 05], where a similar system to the one presented in this thesis was designed. The difference was using FPGA platform for the system implementation. Other concepts of protecting FPGA circuits include projects described in [Andraka], [Baloch 06], [Quicklogic 03], [Katz 97], [Lima 03], [Wirthlin], [Bezerra] and [Srinivasan 04]. The most common approaches are based on DMR and TMR with voting circuits to detect and correct errors. Radiation mitigation methods are described in Chapter 4 of this thesis. Not only system level solutions are presented but also process and layout approaches, to give a more complete view on different design techniques used in radiation hardening.

2.3 Proposed Solution

This thesis proposes a readout system which is supposed to work as a control unit for an integrated neutron radiation detector. The design is based on a Finite State Machine (FSM) and supports serial communication according to the EIA-232 standard. The assumption for the project is that a detector based on a Static Random Access Memory

(SRAM) is used. The memory is programmed with a pattern. The state machine first reads the memory contents, then compares the read values with the predefined pattern and based on the results of the comparison calculates the number of differing bits. Since Single Event Upsets (SEUs) are known to cause bit flips in SRAM, the number of such bit flips is corresponding to the number of SEU occurences in the memory. After reading, the memory is reprogrammed and the number of SEUs is transmitted using serial transmission. Apart from that there is a possibility of extending the system to communicate with a temperature sensor and a gamma radiation detector based on the special RadFET transistor (Field Effect Transistor) [Makowski 06], [Holmes-Siedle 02].

Radiation tolerance in the system is achieved through implementation of Hamming codes to protect the state registers. In this way some basic Error Detection And Correction (EDAC) functionality is provided. Used codes are capable of detecting double errors and, what is the most important, detecting and correcting single errors, which are the most common types of errors concerning the radiation effects. According to [Makowski 04], the probability of a single error (for configuration bits in FPGA placed in collider environment) is approximately 1 error per hour (for a 128 KB memory), whereas of a double error is about 1 error in 10^5 years. Thus, it is sufficient to provide functionality for correcting single errors. Additional CRC coding is applied also to the transmitted message with calculated SEUs. In this way it can be verified if there have been transmission errors when reading data sent from the controller.

Another purpose of this work was to determine a design path. Starting with the behavioural description in VHDL, the goal was to generate a ready layout. Furthermore, the layout was supposed to be verified in order to be sure if it can be manufactured in the specified technology. The aim of this thesis was to show how to protect a digital circuit from Single Event Upsets and to define a way for preparing it for production.

3 Radiation Effects on Electronic Devices

Highly energetic particles constantly cross the atmosphere and deploy their energy while bombarding the surface. The particles that are attenuated in the smallest degree are neutrons and pions [Ziegler 96]. Since the scope of this work concerns mostly the effects of neutron and gamma radiation on integrated circuits, the following sections will focus on the effects of the interaction of neutrons with matter. Neutrons, being neutral particles, unlike the charged particles, are not affected by the coulombic forces. However, due to the lack of charge and its relatively large mass, comparable to the mass of a proton, a neutron penetrates to large depths and is hard to stop. According to [Anelli 00] we can distinguish three types of neutrons, depending on their energy. These are slow, with energy smaller than 1 eV, intermediate, between 1 eV and 100 keV, and fast, with energy exceeding 100 keV. Neutrons can be attenuated in three different mechanisms:

- absorption - neutron is captured by the atomic nucleus, leaving the atom in excitation. In order to reach the stable level, the nucleus can then emit proton, alpha particle or gamma rays. It can even lead to nuclear fission. This phenomenon can occur for all energy levels but is most probable for the slow neutrons.
- elastic scatter - neutron collides with the nucleus and travels further, changing its trajectory, just like the billiard ball. The transfer of kinetic energy may cause the displacement of the nucleus, which can in turn cause further ionisation or displacement in the lattice. This effect can be triggered by slow neutrons but mostly it is due to fast ones.
- inelastic scatter - similar to elastic scatter, however the nucleus receives also some internal energy, thus is left in an exited state. Upon returning to stability it can emit gamma rays. This behaviour is predominant for neutrons of very high energies.

In the domain of radiation interaction with electronic devices and integrated circuits the most common division of radiation effects is into cumulative effects and single event effects. Main focus of this work is on the latter group. In the former group two main phenomena are distiguished, namely displacement damage and ionization effects.

3.1 Displacement Damage

Since a neutron is a heavy particle, it has a high penetrating power when hitting the surface. After entering the bulk of a device not only surface atoms are displaced but a whole displacement tree is formed. According to [Holmes-Siedle 02] it may be considered as a cascade of Frenkel pairs, which are vacancies together with the interstitial atoms. Atomic displacement is related to non-ionizing energy loss, abbreviated NIEL. The deposited dose is regarded as the non-ionizing KERMA (Kinetic Energy Released in MAtter), so is related to the sum of kinetic energies of particles displaced due to radiation interaction. The first atoms displaced by the incident neutron are called primary knock-on atoms, abbreviated PKA. Their energy, being about 50 keV, and 100 nm range are sufficient to give origin to the formation of a displacement tree, which is formed in just one nanosecond after irradiation. At the end of each branch one can see a so-called terminal cluster, being in fact a collection of vacancies surrounded by interstitial atoms. These clusters begin expansion immediately, due to diffusion of vacancies and interstitials into the lattice. Both types have high mobilities. Vacancies are able to combine either with the interstitials or with impurities. The interstitials can in turn exchange an impurity and take its place in the lattice. The silicon lattice slowly returns to stable state due to annealing, which is based on thermal diffusion processes.

3.2 Ionizing Effects

Several terms concerning the effects of IEL, that corresponds to the ionizing energy loss, are used interchangeably in literature. The most popular are ionization, total ionizing dose (TID) and total dose. These effects are related to degradation or destruction of devices due to electron-hole pair establishment after irradiation. The energy required to create a single electron-hole pair in SiO_2 is only 18 eV [Holmes-Siedle 02]. No momentum transfer to atoms is observed in the case of ionizing effects, contrary to displacement damage. The phenomenon present directly after irradiation is partial recombination. Electrons that do not recombine leave the oxide due to their high mobility, whereas holes are mostly trapped. Such effect is the cause for the positive charge trapping in the oxide and at the Si-SiO_2 interface as well as for the new interface states. Such degradation can be present for days or even years, depending on the dose, oxide parametres and structure of the design. Figure 3.1 presents the trapping zones and their location in the oxide.

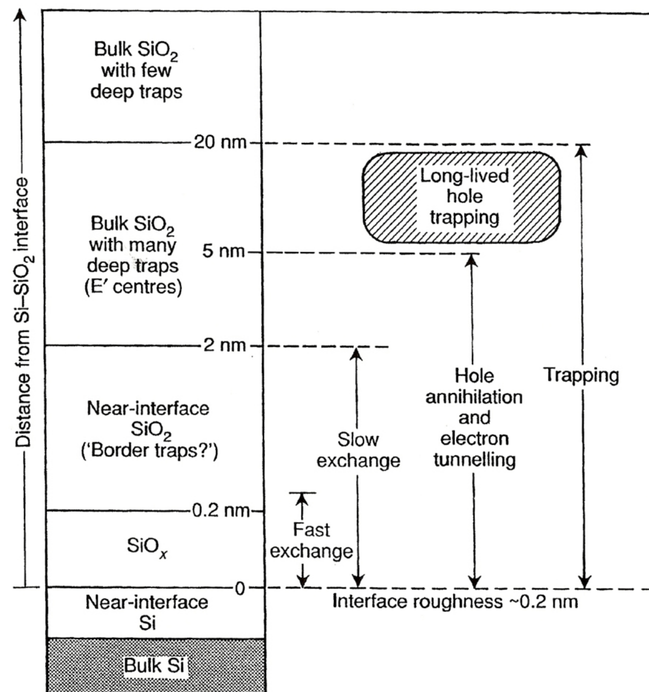

Figure 3.1 Charge sheets structure near the Si-SiO_2 interface ([Holmes-Siedle 02], p. 87).

Holes near the interface undergo electronic exchange and recombine with time because of tunneling effect. Only those deeper in the oxide form a charge sheet, which directly affects the device parameters. Due to the built-in field the charges tend to move away from the trapping centre and the rate of such relaxation is not precisely specified since it is dependent on various factors, like temperature. This phenomenon is referred to as annealing and can mean both strenghening and weakening of trapped charge effects. The charge build-up can temporarily lower the potential barrier, shift the threshold voltage, cause some leakage currents and affect conductivity due to band bending near the Si-SiO_2 interface in MOS devices.

3.3 Single Event Effects

In the late 1970s, with the advent of modern LSI circuits, the radiation effects on electronic devices became the serious issue. As described in [Ziegler 96], IBM performed a series of tests aimed at estimating the results of soft fails on electronics circuits. Since the introduction of the first 16 Kb memory chips it was apparent that the energetic particles coming either from the radioactive decay present in all materials or from the extraterrestial cosmic rays influence the information contained in such devices. Together with the scaling down of technology the consequences of ionized particles bombarding the sensitive nodes of the circuits are of major importance not only in the airborne equipment but also in common applications on the ground.

Electronic devices are affected by Single Event Effects (SEE). SEE is the general name of the phenomena caused by highly energetic particles crossing the integrated circuits [Anelli 00]. Such events trigger immediate malfunctioning of a transistor or, in some cases, of more than one device. These faults can further cause errors in the entire system. We distinguish two main categories of Single Event Effects. These are soft errors, which are

non-destructive and reversible, and hard errors, which are destructive and irreversible. The former are of major concern in this work, together with ways and measures of preventing them.

3.3.1 Soft Errors

Soft Errors do not include damaging of the chip and they can be mitigated simply by resetting the device to the initial state. This kind of errors introduces only functional faults in the system. Also the performance can be worsened. However, soft errors are not dangerous for the device concerning material properties. Single Event Upsets (SEU) are the most common type of soft errors. However, one should not forget the other effects from this category. In this section the most important ones will be explained shortly. Not only are they worth presenting from the theoretical point of view but also the probability of their existence in modern technologies is not negligible.

3.3.1a Transient Effects

Single Event Transients (SET) are responsible for causing malfunctioning of logic circuits. A particle strike can induce creation of electron-hole pairs. For an exemplary pulse of a rate of 10^9 rad/s in silicon the generated concentration can be as high as $10^{18}/cm^3$, which is comparable to the dopant densities in majority of technologies. An abrupt current pulse can be created provided that such charge gathers in a reversed biased junction. A pulse of this kind cannot be distinguished from the correct signals. SETs are observed mostly during the signals transitions, when the device is most susceptible to upsets. Pulse voltage can be higher than normal signals voltages according to the technology. Such voltage burst can travel through the circuit lines and provided advantageous conditions it can reach the outputs of the circuit [Mavis 02], [Massengill 02]. Figure 3.2 illustrates the idea of a Single Event Transient. An example of propagation of an erroneous signal caused by a SET is depicted in Figure 3.3.

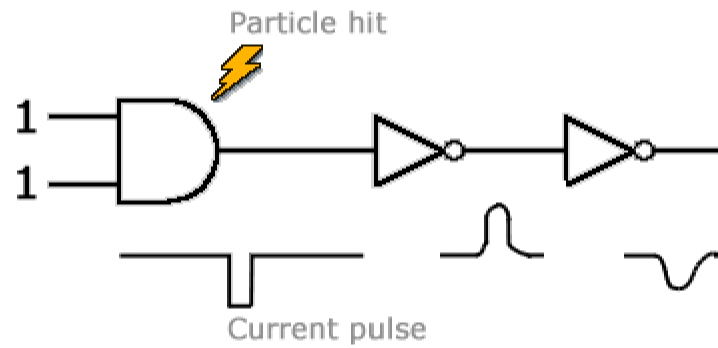

Figure 3.2 The idea of Single Event Transient

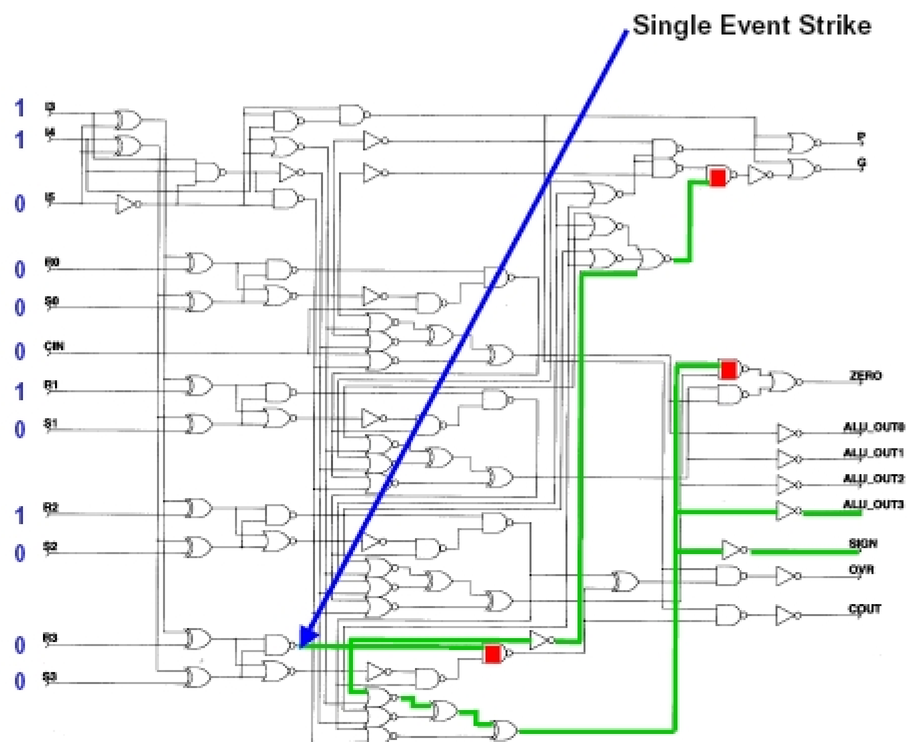

Figure 3.3 Propagation of a Single Event Transient
in a combinational circuit [Massengill 02].

3.3.1b Single Event Upsets

The Single Event Upsets (SEU) are the most common radiation-induced faults affecting the electronic devices. SEU take place when such an highly energetic particle hits a device in an integrated circuit, generating in this way a path of electron-hole pairs. The charges induced in this process are then gathered in the sensitive node of the circuit, resulting in a very short current pulse. In the case of interaction with a transistor it can result in a bit flip and the erroneus information can be propagated further in the system. To predict vulnerability of a circuit element the notion of Linear Energy Transfer (LET) is used. It tells how much energy is transferred to the matter by the incident particles. LET is expressed as [Anelli 00] :

$$LET = 1/\rho \cdot dE/dx, \quad (1)$$

where ρ is the density of material,
dE/dx is the mean energy trasferred to silicon material per unit path length,
$[LET] = J \cdot m^2 \cdot kg^{-1}$ or $[LET] = MeV \cdot cm^2 \cdot mg^{-1}$

LET can be used to calculate the number of generated electron-hole pairs. Knowing LET the total deployed energy can be calculated and then by dividing the result by the energy needed to create a single electron-hole pair the answer is obtained. According to [Holmes-Siedle 02], 3.6 MeV of energy transfer results in 0.16 pC of charge in silicon. LET is connected with two notions:

- Threshold LET, which is the minimum Linear Energy Transfer value needed to result in an upset. It can be compared to the energy depositing the critical charge in the device. This charge in turn corresponds to the volume of a sensitive node, which is affected by SEU, and the degree of charge funnelling, shown in the Figure 3.4.
- Saturated cross-section, which is a situation when all particles that are crossing the

sensitive node are capable of producing upsets and increasing LET does not result in an increase in the upset rate. The cross-section is expressed in cm^2 and depends on the surface area of the sensitive nodes.

Figure 3.4 illustrates the cross section of a MOS transistor when a Single Event Upset occurs.

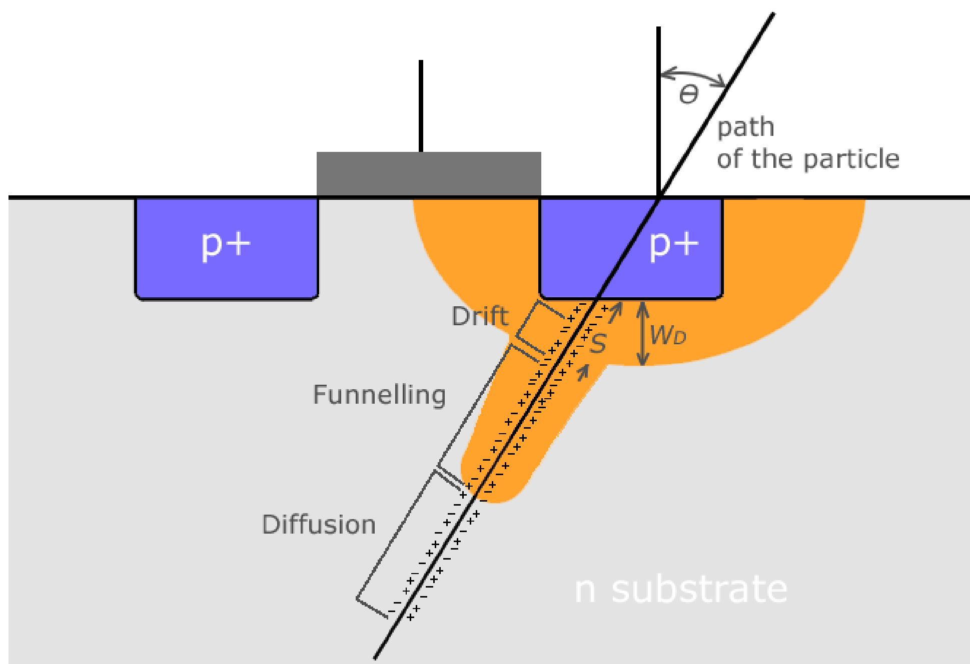

Figure 3.4 A Single Event Upset phenomenon [Holmes-Siedle 02].

SEUs are soft errors, therefore recovering from them can be done by resetting the device.

3.3.1c Multiple Bit Upset (MBU)

MBU is a variation of the Single Event Upset [Anelli 00]. The main difference is that here more than one device is affected after irradiation. Three cases are distinguished, depending on the trajectory of the incident particle. If the angle of incidence equals 90

degrees, the particle can cross the sensitive node of more than one device, because of very close packing of adjacent transistors. Another MBU variation is when the radiation is coming from above the surface of the devices and the particle hits the sensitive nodes at an angle close to 0 degrees. To affect more devices it has to deploy sufficient energy. The third case is just when two particles change the state of two devices. Should they have enough energy, they can cause errors even in adjacent transistors. This however is very rarely encountered [Li 00].

3.3.1d Single Event Functional Interrupt (SEFI)

SEFI is a special case of SEU. It is a regular error with the slight difference that it affects the peripheral modules of the system itself, such as Error Detection and Correction circuits. If an upset is generated in such part of the system it can go undetected, changing the whole functionality [Anelli 00]. Contrary to SEUs, SEFIs are hard to distinguish, since they affect parts of the circuit that cannot be clearly defined, like it is done in the case of memories, for example. Such kind of errors can last for quite long, even as long as the device is turned on. SEFIs can also have more than just a single cause, like the particle hit [Koga].

3.3.2 Hard Errors

Hard Errors are the faults that can cause not only malfunctioning of the system but also even destroy the whole device. They are not that common in the modern technologies thanks to many design advances. However, they are worth describing in general to have a better understanding of the influence of the interaction of energized particles with matter.

3.3.2a Single Event Latch-Up (SEL)

Latch-up is a high-current and low-voltage condition, which occurs in integrated circuits when the power is short-circuitted to the ground. In the CMOS structure it can

happen due to turning on the parasitic thyristor or SCR, the Silicon Controlled Rectifier. Such PNPN structure can be driven into the low impedance (conducting) state inter alia by a pulse of ionizing radiation. It is called the Single Event Latch-Up. The following conditions must be fulfilled for a SEL to occur [Anelli 00], [Holmes-Siedle 02]:

1. first of all the parasitic thyristor must be present, which is ruled out in the Silicon On Insulator technologies;
2. the product of the gains of the two PNP and NPN bipolar transistors producing the PNPN structure must be greater than 1, which allows for generating the positive feedback loop;
3. the two parasitic bipolar transistors must maintain the forward-bias condition in their emitter-base junctions, which enables the injection of excess minority carriers that are generated in the base regions after the radiation effects;
4. power supplies must be high enough to provide a current necessary to maintain a latch in the device.

The latch-up mechanism for a CMOS structure in silicon is illustrated in the Figure 3.5. A SEL is proven to be a long-term effect and if the power is not turned off immediatelly, it may be even destructive for the integrated circuit.

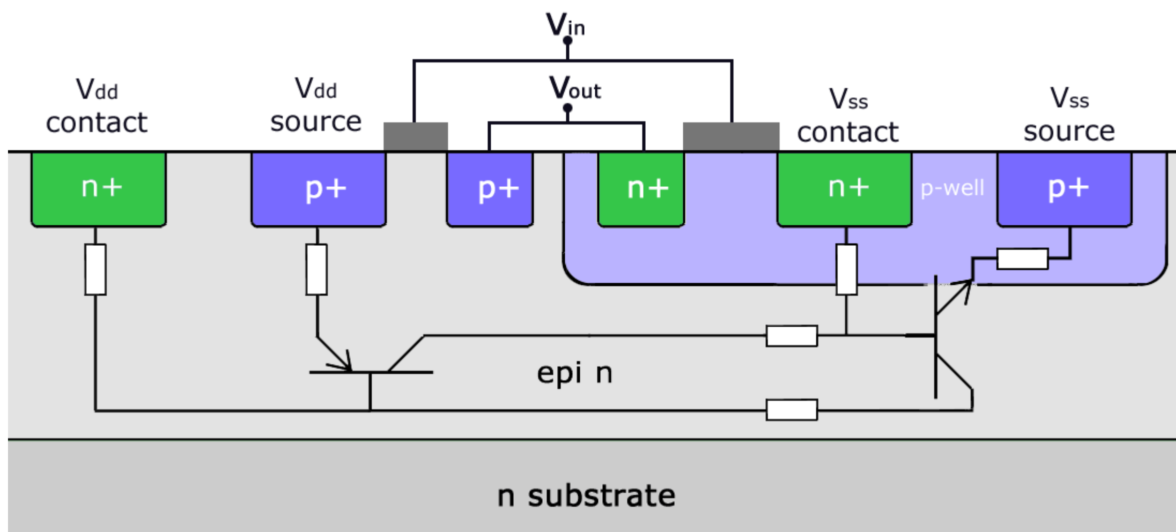

Figure 3.5 Latch-up mechanism in a silicon CMOS structure.

3.3.2b Single Event Snapback (SES)

Single Event Snapback can be observed in n-channel devices, like the n-channel MOS transistor. In this case it does not rely on the parasitic SCR structure but on the parasitic bipolar structure, having its nodes corresponding to the nodes of the MOS device. Emitter, base and collector correspond to source, substrate and drain. When an energized particle hits the transistor the parasitic NPN structure can be turned on by an avalanche mechanism which is then maintained by the injection of electrons from the source to the drain [Anelli 00], [Holmes-Siedle 02]. It can be visualized as turning on the transistor and keeping it on [Wikipedia]. This effect is generally connected to very high supply voltages.

3.3.2c Single Hard Error (SHE)

This kind of error happens when an ionized particle hits the MOS transistor and passes through its gate oxide. In this way even a shift in the threshold voltage may be observed. SHE was more important in the older technologies, where a thick gate oxide was used [Anelli 00].

3.3.2d Single Event Gate Rupture (SEGR)

The Single Event Gate Rupture is an irreversible and destructive effect, found in the devices with a very high values of electric field. Thus, it can be experienced mainly in the Power MOSFET transistors and in EEPROM chips while writing or erasing the memory. After striking the device in the thin gate oxide region, the energized particle generates electron-hole pairs. The charges then start drifting, electrons towards the drain contact and holes towards the Si-SiO_2 interface. Electrons diffuse faster, which is the cause for the gathering of holes. This creates a place with large positive charge, increasing the electric field in the oxide. If this field gets high enough, the oxide can break down, causing the discharge of holes through it, which in turn leads to overheating the region of the oxide

breakdown. This can destroy the whole device. SEGR can be a serious issue in the modern technologies, since we observe constant scaling down of the gate oxide. Experiments have shown however that up to the 0.25 μm technology it is negligible. [Anelli 00], [Allenspach 95]

3.3.2e Single Event Burn Out (SEBO)

The Single Event Burn Out is observed in high current devices, especially in Power MOSFETs and bipolar transitors. A parasitic bipolar structure is necessary to obtain a SEBO. An ionized particle crossing the source of the MOS transistor can induce a current flow in the npn structure in the bulk of the device, which initiates a forward biased second breakdown. Provided sufficiently high current in the parasitic bipolar transistor, the locally dissipated power can lead to the meltdown of the whole device. However, SEBO is important mainly for the designers of power transistors [Anelli 00], [Foutz 92], [Normand 97]. See Figure 3.6.

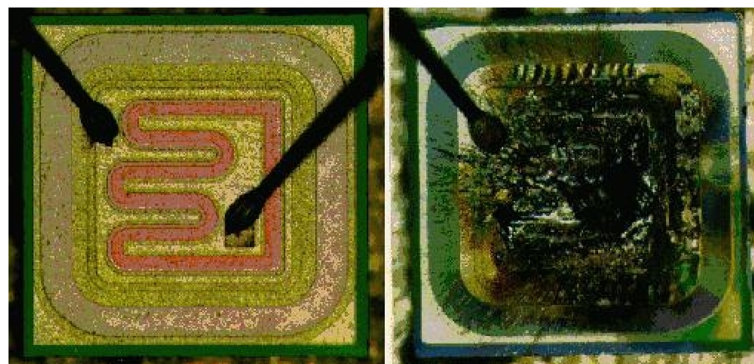

Figure 3.6 Semiconductor device before and after a SEBO [Carson 97].

4 Mitigation of Radiation Effects

Together with advent of modern technologies, with their extremly high circuit densities and shrinking geometries, the effects of radiation are beginning to pose a serious threat not only to cosmic and airborne systems but also to the simplest, common-use devices. As it was discussed earlier, the energized particles can cause malfunctions or even, in extreme cases, destroy the device. This is especially important in systems, where reliability is of major concern, such as medical, avionic, cosmic, automotive or military.

Designers of integrated circuits have a wide spectrum of techniques providing satisfactory protection against different types of radiation. Since the scope of this work is protecting devices against neutron radiation, in the following sections a brief description of various methods will be provided. The first part presents the hardware techniques are presented, so the ones corresponding to the layout and manufacturing technology. The second part, on the other hand, gives some concepts that can be implemented from the software point of view. These are different coding schemes and methods that can be applied not interfering with the technology used.

4.1 Hardware Mitigation

Radiation-hardening can be achieved by adjusting the process and layout rules. Such methods are proven to be very successful, however, they require altering the technology, which is not always possible, as in the case of FPGA systems or in fabless companies. The next sections give an overview of some of the most common techniques for radiation-hardening the integrated circuit technology-wise.

4.1.1 Process Hardening

As it is well known and does not have to be explained in detail, the CMOS process consists of tens or hundreds of steps. It has been shown from various tests that many of these stages during manufacturing a device can significantly affect its vulnerability to Single Event Upsets. The most important elements of the process are those changing the possibility of charge trapping in the oxide or at the oxide interface of a MOS transistor [Holmes-Siedle 02].

4.1.1a Wafer Cleaning and Polishing

It is clear that special attention ought to be paid when preparing the substrate for the integrated ciruits. The dislocations in the lattice together with the presence of impurities can affect not only the performance of the ready device but also its radiation hardness. It has been shown that the (100) orientation for the surface of silicon wafer is advantageous. After careful cleaning and polishing the surface should also be treated with a dilute Hydrogen Fluoride solution. HF is known for a great effectiveness in removing most of the organic and inorganic contaminants. [Holmes-Siedle 02]

4.1.1b Gate Oxide Growth

The greater thickness of gate oxide, which implies larger oxide volume, increases transistor's vulnerability to radiation effects in the way that the number of holes that can be trapped also increases. The threshold voltage is dependent on the gate oxide thickness. The threshold voltage shift depends on the oxide thickness as t_{ox}^{n}, where n is empirically found to be a value between 1 and 3. Experiments have shown that n is affected by the conditions for the oxide growth process, like the used ambient and electric field. The temperature assuring best hardness has been found to be around 850°C for wet oxidation and around 1000°C for dry oxidation. It has to be noted that wet oxidation generates higher values of n than the dry process. Moreover, the use of forming gases added to the oxygen flow can

make a difference. For example argon has proven to be advantageous, whereas nitrogen disadvantagous in achieving radiation hardness in oxide growth process. [Holmes-Siedle 02]

4.1.1c Oxide Annealing

The process step after gate oxide growth is the gate oxide anneal, which is done in order to relieve stress in the SiO_2-Si interface. It also decreases the number of defects in the oxide. In order to achieve radiation hardness, the processes following the growth of the gate oxide should be performed in temperatures as low as possible, also with short times. In [Lenahan 99], where a thorough study of different gate oxides is performed, it is confirmed that hole trapping is determined by the highest processing temperature. The best radiation hardness was achieved with annealing between 850°C and 950°C for times between ten and several tens of minutes. The ambient was chosen to be a forming gas, being 10 % hydrogen and 90 % nitrogen [Holmes-Siedle 02].

4.1.1d Gate Electrode

Modern trend is using polycrystalline silicon as the gate material, because of the self-aligned technology and the enhancing performance of the devices. The problem may be the ion implantation through the oxide mask. This process may introduce oxide damage, which generates the additional trapping centres. Another disadvantage of polysilicon are the high-temperature processes following the gate oxide growth, which determine the radiation hardness. However, modern manufacturing technologies use lower temperatures, which enables achieving similar hardness as is the case for traditional, metal gates. The new concepts include the use of refractory materials, like tungsten and tantalum, and metal silicides as the gate material. Such materials are advantageous for improving the radiation hardness, reducing the interface states in the oxide, toghether with decreasing the charge trapped in this region [Holmes-Siedle 02].

4.1.1e Silicon On Insulator (SOI) Technology

When discussing hardening by process one should not forget the Silicon-On-Insulator (SOI) alternative technology. Although such approach is much more expensive, it gives a full protection against Single Event Latch-Up, since there is no parasitic thyristor. The vulnerability to Single Event Upsets in SOI technology is also diminished. Also leakage between devices is eliminated. SOI technology is worth considering in designs especially prone to SEL [Anelli 00]. Silicon On Insulator is also advantageous because the sensitive volume is smaller because of shallow insulation layer. The cross-section of an exemplary device manufactured in SOI technology is shown in the Figure 4.1.

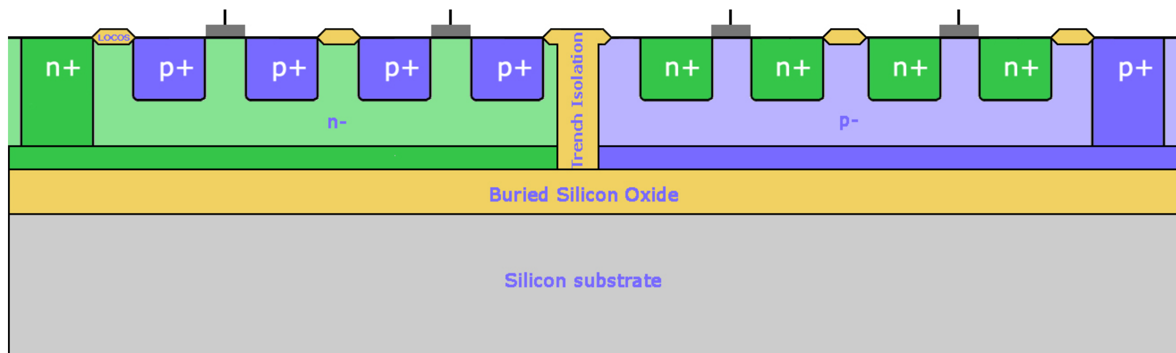

Figure 4.1 SOI technology [Anghinolfi 00]

4.1.2 Layout Hardening

This section presents some layout hardening techniques. Presented solutions have been proven successful, however they require a serious topography change, which is not always allowed. They can be used for example in the full custom designs.

4.1.2a Enclosed Layout Transistors

Standard transistor layout does not prevent the occurence of the total ionizing dose-induced leakage currents in n-channel MOS devices. The leakage paths are shown in the Figure 4.2. In order to overcome this problem, some special layout strategies were designed. [Anelli 00] describes three alternatives. First, illustrated in the Figure 4.3 is a MOS transistor with a modified gate, so that the leakage path length is increased significantly. Such design is rather simple and thus not very much effective. The parasitic devices are still present, however it is less likely that they can be turned on. Figure 4.4 presents the second solution, where a p+ diffusion is added around the original n+ drain and source regions. In this case the thin oxide is defined within the limits of the p+ diffusion. This alternative increases the leakage path even more. However, the parasitic currents may still exist provided high enough total ionizing dose. Another problem with this approach is violation of design rules for some technologies. The most effective solution is presented in the Figure 4.5. The Enclosed Layout Transistors provide total elimination of the parasitic paths since the drain region is placed inside the gate. There are however some disadvantages of the edgeless transistors. The most striking aspect of this technology is the area usage, significantly larger than in the standard approach. Another problem is the difficulty in modelling such transitor, since the W/L ratio is not straightforward. Also the gate and drain or source capacitances are larger than in the standard transistors. However, ELT is the best solution for some most demanding radiation environments. Provided careful modelling, a compromise between high density and radiation hardness can be achieved.

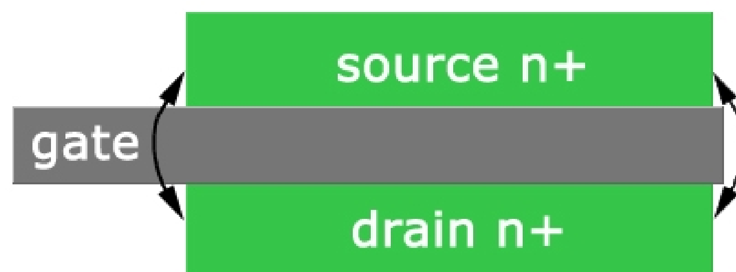

Figure 4.2 A standard self-aligned n-channel MOS transistor layout [Anelli 00]

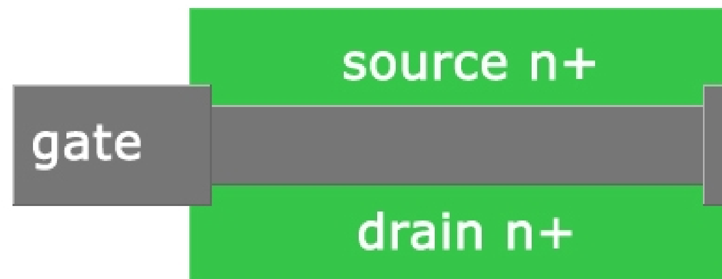

Figure 4.3 MOS transistor with modified gate to increase the leakage paths [Anelli 00]

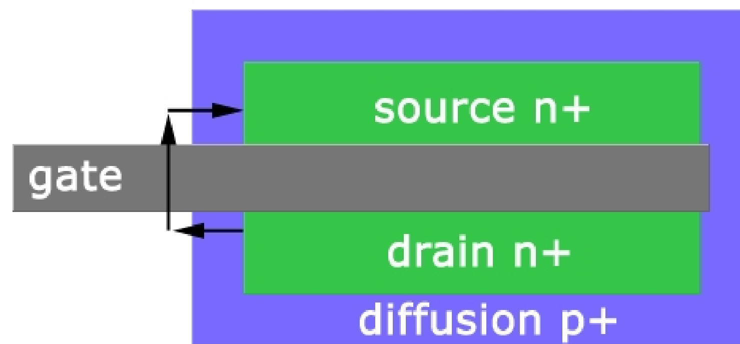

Figure 4.4 MOS transistor with the thin oxide region defined by the p+ diffusion [Anelli 00]

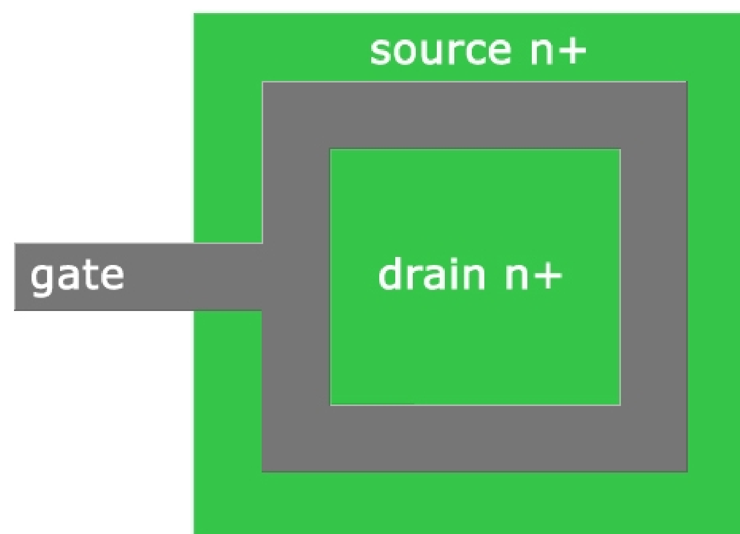

Figure 4.5 Layout of a rad-hard MOS transistor designed according to the Enclosed Layout Transistors topography [Anelli 00]

4.1.2b Guard Rings

Leakage between adjacent devices is a really important issue in the CMOS technology. According to [Anelli 00], surrounding each n-channel device with a p+ guard ring and, conversely, each p-channel device with an n+ guard ring is very effective. Although area consuming, it provides means for eliminating the parasitic leakage paths. Apart from that such structures are also means for protecting devices from Single Event Latch-Up, since the gain of the parasitic NPN transistor is significantly decreased by the p+ guard ring and, respectively, the gain of the PNP bipolar by the n+ guard ring. The use of guard rings together with the devices in ELT technology is shown in the Figure 4.6. in a radiation tolerant CMOS inverter layout.

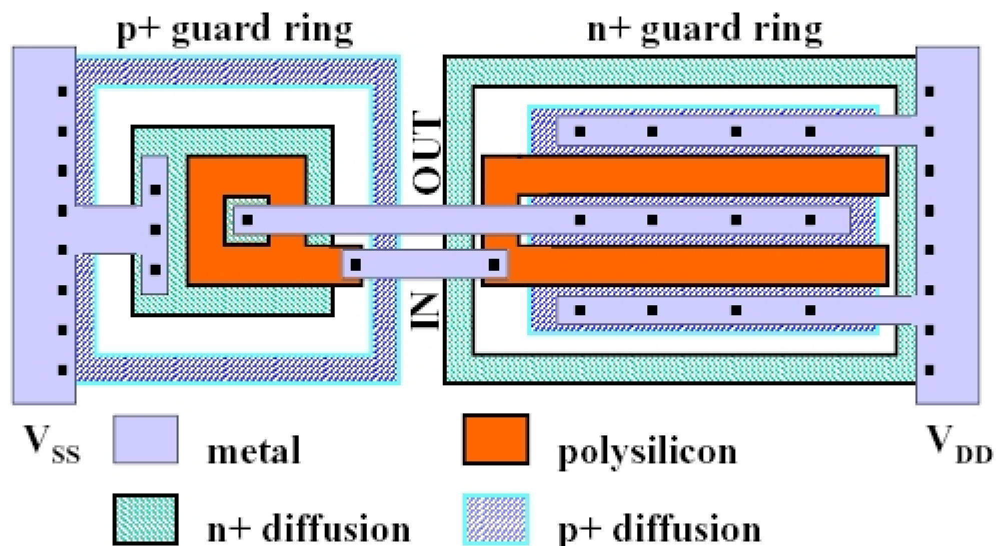

Figure 4.6 Use of ELTs and guard rings in a radiation tolerant CMOS inverter layout [Anelli 00]

4.1.2c Trench Isolation

Another way of preventing leakage currents between devices is using trench

isolation. Deep trench made by deposition of a vertical insulation layer effectively protects carriers from flowing along the parasitic paths between adjacent transistors [Shaneyfelt 98]. This concept can be seen also in the SOI technology, as presented in the Figure 4.1.

4.1.3 System Hardening

Using different process and layout techniques has proven to give very good protection against faults caused by radiation. However, designers usually do not have the possibility to influence the technological aspects and are forced to stick to the given specifications. In such case, which is the most common for the smaller-scale projects and commercial applications it is practical to use some techniques for error mitigation on the system level. The easiest method is to use some periodical refresh method, which just reconfigures the whole system, no matter if any SEUs occured or not. The other solution is to use some Error Detection And Correction (EDAC) schemes. This is a quite simple way. With Error Detection one can detect errors and just refresh the system when a fault occurs. Such approach may be used for example in the cheap mainstream projects, where reliability is not of the major concern and when altering the technology is not possible. In mission critical applications, such as medical, banking or space ones, designer can implement more complicated Error Detection And Correction solutions. These include coding schemes, such as Hamming or Reed-Solomon codes. The other concept is to use one of the redundancy schemes, double or triple modular redundancy. These methods are based on replicating some modules of the system and using an additional voting module to distinguish the correct information. Such approach requires however a large overhead, which may not be suitable for the commercial applications. All of these concepts are treated in more detail in the following sections.

Coding Schemes

One of very effective ways of protecting data against errors, let them be caused by radiation or by any noise on the communication line, are various types of redundancy

codes. We can distinguish error correcting and detecting algorithms. Their strength and effectiveness is usually directly connected with the associated overhead. This is related to the concept of redundancy check, which is all about adding some additional data to the original code word before transmitting the message through the communication channel. In the following sections there are presented codes starting with the simple checksums, through CRC and Hamming codes. Since the scope of this work is dealing with SEUs, that is, single errors caused by radiation, the main focus in on the codes' capability to detect and correct such errors. Dealing with double and burst errors are omitted or mentioned just as a reference.

4.1.3a Parity Bit for Detecting Single Errors

Adding a parity bit is the simplest method of error detection. Parity bit indicates if there is an even or odd number of ones in the code word. The most common parity scheme is the even parity, where the total number of bits, together with the appended one, is even. This means that if the number of ones in the original message is even, the parity bit will become a zero, and if the number of ones is odd, it will become a one. To generate a code word with the parity bit it is sufficient to use a XOR gate to calculate parity and a shift register to append the parity bit to the message.

The concept of parity is used in many more sophisticated coding schemes. Itself it has rather limited effectiveness, since it can only detect odd number of errors. However, it can be useful for detection of single errors, which are the most common in many transmission systems. Also in the case of radiation tolerance, single errors are the most probable. The limitation of this coding method is that no errors can be corrected and the whole code word has to be retransmitted upon detection of an error. Even parity can be considered the easiest form of CRC, where the x+1 polynomial is used to generate a single parity bit.

4.1.3b Checksum for Error Detection

Another simple form of redundancy check is applying the checksum. This method has only a limited error detecting capability. The concept consists of calculating the sum of digits in the original code word and attaching it to the transmitted message. The receiving unit then calculates the sum of digits in the received data word and compares the sum with the received one. This enables distinguishing odd number of errors which results in some sort of negative acknowledge message and retransmission. However, there are many disadvantages of this technique. Firstly, if the bits in the data word get reordered, the checksum remains the same. Secondly, omitting zeroes in transmission does not change the checksum, similarly as it happens if some zeroes are inserted into the message. Apart from that, if multiple errors occur, which result in the same checksum, also the receiver treats such bitstream as a valid message. Thus, checksum can be useful when there is a large probability that only single errors occur during transferring the message, such as errors caused by radiation, or in not very critical applications.

4.1.3c Cyclic Redundancy Check Codes

The Cyclic Redundancy Check (CRC) is a very popular and powerful method of error detection. In this case the redundancy appended to the message is actually the remainder of the divison of the original message, treated just as a binary number, by a generator number. The feature of divison is that when the dividend changes by a very small value, the quotient is not affected as much as the remainder. Thus, in CRC, the quotient is not used and the main focus is on the remainder, which reflects even one bit changes in the transmitted message. The divisor, called also the generator, is known to both transmitter and receiver. Upon receiving the message, division is performed again and the transmitted and received remainders are compared, just like in the checksum method. The trademark of CRC is using the polynomial arithmetic, which is described in the following section, since it's the most confusing aspect of this algorithm [CRC 96],[Ritter 86],[Tanenbaum 03],[Olcayto 97],[Wikipedia].

Polynomial Arithmetic

CRC uses the polynomial mod 2 divison in the finite Galois Field of order 2, abbreviated GF(2), so every number is represented as a polynomial with binary coefficients. To illustrate this concept, Table 4.1 shows decimal, hexadecimal, binary and polynomial representation of two exemplary numbers.

Table 4.1 Decimal, hexagonal, binary and polynomial representation of numbers

dec	hex	bin	poly
20	14	10100	$1 \cdot x^4 + 0 \cdot x^3 + 1 \cdot x^2 + 0 \cdot x^1 + 0 \cdot x^0 = x^4 + x^2$
17	11	10001	$1 \cdot x^4 + 0 \cdot x^3 + 0 \cdot x^2 + 0 \cdot x^1 + 1 \cdot x^0 = x^4 + 1$

Mod 2 arithmetic for polynomials means that x can only have the value of 0 or 1. This also means that the mathematical operations are always performed with no carry and each digit is computed separately. The reason for such a change from normal arithmetic is that each coefficient is related in no way with its neighbouring coefficients. Thus, addition and subtraction as well as multiplication and division are done bit by bit. It is very convenient, since in implementation there is no complicated circuitry usually associated with the carry operation, which is always a large design area factor. To facilitate the understanding of arithmetic with no carry, Figure 4.7 gives a comparison between traditional and mod 2 basic mathematical operations on exemplary binary vectors.

a = 1101 = d'13
b = 1011 = d'11

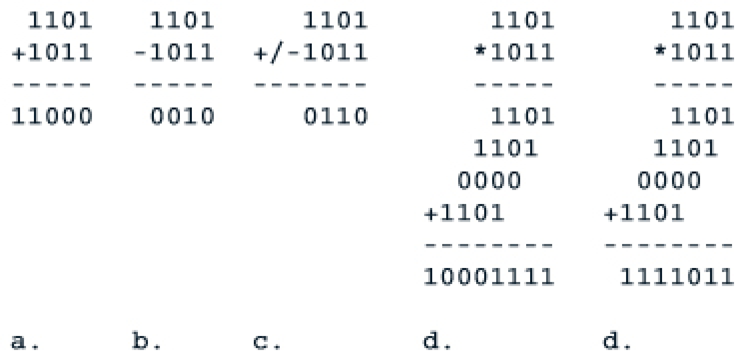

Figure 4.7 Binary vector mathematical operations: a. addition, b. subtraction, c. addition/subtraction with no carry, d. multiplication, e. multiplication with no carry.

Examples a. and b. in Figure 4.7 show the typical approach to binary operations, whereas example c. presents the same with no carry. It is clear that performing these operations bit by bit is equivalent to applying a bitwise XOR function to the vectors *a* and *b*, and the following is valid:

$0 + 0 = 0$ $\qquad$ $0 - 0 = 0$

$0 + 1 = 1$ $\qquad$ $0 - 1 = 1$

$1 + 0 = 1$ $\qquad$ $1 - 0 = 1$

$1 + 1 = 0$ $\qquad$ $1 - 1 = 0$

Addition is identical as subtraction, since it is known that in ordinary arithmetic subtracting a number is achieved by adding the additive inverse of that number. Thus, in mod 2 arithmetic, since the additive inverse of both 1 and 0 are those numbers themselves, addition and subtraction are in fact the same. Turning to example d. in Figure 4.7 one can see that when using the polynomial representation the answer becomes

$$x^7 + x^3 + x^2 + x^1 + x^0,$$

which is in accordance with the answer when the multiplication is done using polynomials,

since:

$a = 1101 = x^3 + x^2 + x^0$
$b = 1011 = x^3 + x^1 + x^0$,

so:

$$(x^3+x^2+x^0)\cdot(x^3+x^1+x^0) = x^6+x^4+x^3+x^5+x^3+x^2+x^3+x^1+x^0 =$$
$$= x^6+x^5+x^4+3\cdot x^3+x^2+x^1+x^0$$

supposing, that x=2, the coefficients propagate, as in a carry system, so the answer becomes:

$$x^7 + x^3 + x^2 + x^1 + x^0 = 10001111.$$

Example e. shows that in mod 2 arithmetic the answer is just the result of applying a XOR function to the vectors bit by bit when adding. Since XOR is a 1 when there is an odd number of 1s, the coefficient 3, seen in the polynomial multiplication results in a 1. Again, the answers here are equivalent.

To illustrate binary division, which is in fact the basis for the CRC algorithm, Figure 4.8 is used, where:

$a = 111011$ - the dividend
$b = 11 = x + 1 = x^1 + x^0$ - the generator polynomial

```
         10011                    101101
   11 | 111011              11 | 1110110
       -11                        11
       ===101                     =01
          -11                      00
          =101                     =10
            -11                      11
            =10                      =11
                                       11
                                       =01
                                         00
                                         =10
                                           11
                                           =1

   a.                          b.
```

Figure 4.8 Binary division: a. ordinary way compared to b. no carry method

In a. the answer is clear, 111011 / 11 = 10011 r 10, which is equivalent to: 59 / 3 = 19 r 2. However in b. the concept of binary division with no carry is presented. In order to do this operation one must append a vector of 0 bits to the dividend. The number of zeros is equal to the width of the divisor, so the generator polynomial, which is exactly the number of the highest power of x. In this case it is just one. In the division all the subtractions are performed mod 2. The divisor is subtracted by comparing to the actual bits from the dividend. Since in mod 2 arithmetic there is no concept of order, the number is considered greater if the position of the highest bit in one vector is greater or equal to the position of the highest bit in the other vector. Using these rules the remainder appears to be 1. Since, as it was said earlier, the remainder is what changes even when the dividend changes by a single bit, the quotient is discarded and the remainder is what counts. If the vector *a* from this example was to be transmitted as a CRC code, the remainder would substitute the appended zeros in the original message. To see the CRC algorithm working some simple vectors will be used to illustrate the concept. Suppose that the message that is to be transmitted is an 8-bit vector x = 11011011. The generator polynomial used is $x^3 + x^1 + x^0$, which is represented by a 4-bit vector y = 1011. After appending 3 zeros to x, which is the width of generator y, one obtains a vector 11011011000. Performing division yields quotient z = 11111010, which is discarded since it plays no role in the CRC algorithm, and

remainder r = 110. The message ready to be sent is the original vector x with remainder r appended, giving the vector 11011011110. This is illustrated in Figure 4.9.

```
original message:       x = 11011011
x with appended zeros: x'= 11011011000
generator polynomial:  y = 1011
quotient:               z = 11111010
remainder:              r = 110
transmitted message:    m = x + r = 11011011110
```

Figure 4.9 Constructing a CRC message with generator polynomial $x^3 + x^1 + x^0$.

What is interesting is how the receiver deals with such message. In fact, there are two ways of validating it. First is to discard the appended checksum, append zeros again to the message, divide it by the generator polynomial and compare the received remainder with the calculated one. Second way is to divide the whole vector m by the generator polynomial again and check if the resulting remainder is a zero. The latter method is preferred, since it is considered cleaner from the mathematical point of view.

The most significant part of constructing a CRC algorithm is choosing the most suitable generator polynomial. Different polynomials are more effective for detecting different kinds of errors. For the most common, single bit errors, it is sufficient if the generator polynomial has at least two bits set to one. Assuming that the receiving unit aquires a message $m + e$, where e is the error vector which inverts all bits in m on positions where e is set to one, the division $(m + e) / y$ is performed. Since dividing m by the generator y yields zero, the result of such division becomes e / y. Now suppose that e is a single error, so $e = x^i$, where i represents the position of the error. It is clear that if the generator y is two or more bits long, it cannot divide the error vector e, thus, the single error will always be detected.

Various tests and implementations led to choosing some standard generator polynomials, which are used in computer networks and telecommunications applications. Some common CRCs are presented in Table 4.2.

Table 4.2 Some standard CRC polynomials with their hexadecimal representations.

Standard	Polynomial	Hexadecimal	Usage
CRC-1	$x + 1$	0x01	Parity bit
CRC-5	$x^5 + x^2 + 1$	0x05	USB token
CRC-8	$x^8 + x^2 + x^1 + 1$	0x07	ATM HEC
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^1 + 1$	0x233	
CRC-16-CCITT	$x^{16} + x^{12} + x^5 + 1$	0x1021	X25
CRC-16-IBM	$x^{16} + x^{15} + x^2 + 1$	0x8005	USB Data
CRC-32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+$ $+x^7+x^5+x^4+x^2+x+1$	0x04C11DB7	IEEE 802.3 Ethernet standard, AAL5

Speaking of efficiency, for example the CRC-16-CCITT algorithm provides 99,998 % error detection capability for all possible errors with just 2 bytes of overhead. In the case of mitigation of SEUs, the CRC codes can provide means for detecting bit flips in a transmitted codeword. In this way there is a possibility of retransmission if due to radiation some errors have occurred.

4.1.3d Hamming Codes

Error detecting codes are not always suitable, since they require retransmission every time an error occurs. Sometimes this is not enough and error correcting codes have to be implemented. One of the most popular and efficient coding schemes was invented by Richard Hamming at Bell Labs in the 1950s and is since known as the Hamming code.

Before delving deeper into the coding scheme itself, some basic notions have to be explained in more detail. Firstly, one should note again, that a codeword is usually described in the form (n,d), where n is the total number of bits in the codeword and d is the number of bits of the original message that is to be transmitted on the channel. Secondly,

the information rate is the number of data bits d divided by total number of transmitted bits n. This shows how much overhead there is to each data frame. Another very important concept is the Hamming distance. It determines how many bits differ in two codewords. Calculating the Hamming distance for a given set of binary vectors is just applying a XOR function to them and counting all the ones in the resulting binary vector. This is illustrated in Figure 4.10.

```
  a    = 11011011
  b    = 10111101
aXORb = 01100110 -> d = 4
```

Figure 4.10 Calculating the Hamming distance

Hamming distance of 4 means that it requires 4 bit flips or 4 single bit errors to change one valid codeword into another valid codeword. If there is a set of legal codewords always the minimum distance between two valid codewords is considered the Hamming distance of the complete set of valid codewords of the code. Hamming distance is so significant because it determines both error detection and error correction capability of the given code. As it was mentioned previously, to change one legal message to another one d single bit errors are needed. Thus, a distance d + 1 of the code is sufficient to obtain a code detecting d errors. In order to correct d errors, the distance that is necessary boosts to 2d + 1. This is due to the fact that even if d bits are flipped, the original term is still closer to the erroneus, transmitted message than the next codeword. Such message differs in less digits from the original one than from any of neighbouring messages. Let the previously described parity bit serve as an example. It is known to be able to detect single bit errors. This confirms the Hamming distance principles, since a change of one single bit results in a vector with a wrong parity. Thus, it shows that the Hamming distance of 2 is sufficient to detect single bit errors in the message [Tanenbaum 03],[Olcayto 97],[Wikipedia].

Constructing a Hamming Code

The key to Hamming codes is attaching some redundant bits to each message, just like in other codes. Here, however, parity bits are calculated for some sets of data bits and those parity bits are constructed so that they overlap. In this way some data bits may be checked by more than one parity bits. In the Hamming code the redundant part of the codeword is not appended at the least significant positions of the message. The convention is to use the bit positions that are powers of 2 as the check bits, so the bits 1, 2, 4, 8, 16 and so on. All the other bits, namely bit 3, 5, 6, 7, 9 and so on, are considered data bits. Always the bit in the ith position is included in parity check computations of the check bits whose positions, being as was previously explained the powers of 2, sum up to i. This means that for example data bit in position 3 is included in checks for parity bits in positions 1 and 2, as 3 = 1 + 2. The same applies for data bit in position 11, being checked by parity bits in positions 1, 2 and 8, as 11 = 1 + 2 + 8. Looking closer at parity bits, all they do is set the parity of a set of data bits. This means that a XOR function can be applied here, giving the even parity. Choosing which bits are included in the calculation can be considered as a pattern of bits that are alternately checked and skipped, counting from the position of the parity bit [Tanenbaum 03]. This is illustrated in Table 4.3.

Table 4.3 Constructing parity check bits in a Hamming code

Position	Number of checked and skipped bits				
n	check n-1	skip n	check n	skip n	check n
1	0	1	1	1	1
2	1	2	2	2	2
4	3	4	4	4	4
8	7	8	8	8	8
16	15	16	16	16	16

To achieve the capability of detection of double errors an extra parity bit can be used. Such check just XORs all the bits from the coded message together. This method does not affect the single bit error correcting capability though.

An example for a (13,8) Hamming code is given in the Table 4.4 below. It should be noted that the bit positions in the codeword n are counted here from the left. Depending on the encoder and decoder design position 1 can be transmitted either as the most significant or the least significant bit.

Table 4.4 Constructing a (13,8) Hamming code

Codeword:	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}	n_{11}	n_{12}	n_{13}
Parity and data:	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7	d_8	c
Original data:			✓		✓	✓	✓		✓	✓	✓	✓	Additional parity
p_1			✓		✓		✓		✓		✓		
p_2			✓			✓	✓			✓	✓		
p_3					✓	✓	✓					✓	
p_4									✓	✓	✓	✓	

To illustrate more clearly how Hamming codes work the following figures provide the full encoding and decoding example for the transmission of an 8-bit message with a single error and a double error. Table 4.5 shows the encoding procedure, as was described earlier.

Table 4.5 Encoding a (13,8) Hamming message for a vector d = 11011011

Codeword:	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}	n_{11}	n_{12}	n_{13}
Parity and data:	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7	d_8	c
Original data:			1		1	0	1		1	0	1	1	Additional parity
p_1	1		1		1		1		1		1		
p_2		1	1			0	1			0	1		
p_3				1	1	0	1					1	
p_4								1	1	0	1	1	
Coded message:	1	1	1	1	1	0	1	1	1	0	1	1	0

Having the coded message n = 1111101110110 the desired thing would be to obtain just such bits and decode it into the original message. However, it is important to show the error detecting and error correcting capability of the Hamming code. Table 4.6 presents how the message is corrected after receiving a vector with one bit flipped. In this example the d_8 data bit, which is the 12th bit of the codeword is changed. What should be noted before analyzing this case is the notion of syndrome. After calculating once again the parity for all parity sets p_1, p_2, p_3 and p_4, the resulting vector tells if an error has occured. This vector, where p_4 is the most significant bit, is called the syndrome. When one represents its value as a number, it gives directly the position of the erroneous bit, counting from the left. Flipping this bit results in a correct message. Such vector can be simply decoded to obtain the original message sent.

Table 4.6 Decoding a (13,8) Hamming message for a vector n = 1111101110100

Codeword:	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}	n_{11}	n_{12}	n_{13}	check
Parity and data:	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7	d_8	c	
Received message:	1	1	1	1	1	0	1	1	1	0	1	0	0	1
p_1	1		1		1		1		1		1			0
p_2		1	1			0	1			0	1			0
p_3				1	1	0	1					0		1
p_4								1	1	0	1	0		1
												Syndrome:		s = 1100

The parity check digit c informs that there has been a single error and the syndrome vector s gives the position of the erroneous bit. Converting s = 1100 to decimal, one obtains that s = d'12, so the 12th bit should be flipped. This results in the vector n = 1111101110100 which is decoded into d = 11011011 that in turn is the original, decoded message. This proves the single error correcting capability of the presented Hamming code. Table 4.7 provides an example illustrating the double error detecting capability, using the same exemplary vector but in this case not only the 12th, but also the 3rd bit are changed,

resulting in the transmitted message being n = 11101101110100.

Table 4.7 Decoding a (13,8) Hamming message for a vector n = 1101101110100

Codeword:	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{10}	n_{11}	n_{12}	n_{13}	check
Parity and data:	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	d_6	d_7	d_8	c	
Received message:	1	1	0	1	1	0	1	1	1	0	1	0	0	0
p_1	1		0		1		1		1		1			1
p_2		1	0			0	1			0	1			1
p_3				1	1	0	1					0		1
p_4								1	1	0	1	0		1
												Syndrome:	s = 1111	

As can be seen in the Table 4.7, in the case of a double error the syndrome in also not equal to zero. However, it does not show the position of one of the two erroneous bits. Apart from that one should observe that the additional parity check performed on all transmitted bits results in zero. This, together with the non-zero syndrome is a mark for an occurrence of a double bit flip. It has been shown that a (13,8) Hamming code is able to correct a single bit error and detect a double bit error. Such coding scheme is very powerful and efficient, giving the information rate of 0.62 and overhead of just 38 % [CRC 96],[Ritter 86],[Tanenbaum 03].

4.1.3e Modular Redundancy

Modular redundancy is a very straightforward, yet effective way of SEU mitigation. The idea behind redundancy is replication of a module and using a voter to validate the correct output. In the case of Double Modular Redundancy the module under consideration is duplicated. Whenever outputs from the replicated modules differ, the voting circuit signals an error and the modules are refreshed. This however provides users only with error detecting capability, since the correct output from the module cannot be chosen. Better way, though much more area consuming, is Triple Modular Redundancy. Here a module is triplicated and the voter can distinguish the correct output in case of a single

error in one of the modules. This gives users additional error correcting capability. TMR is not a new idea. It was presented already in 1956 by J. Von Neumann in [Neumann 56], as it is presented in the Figure 4.11.

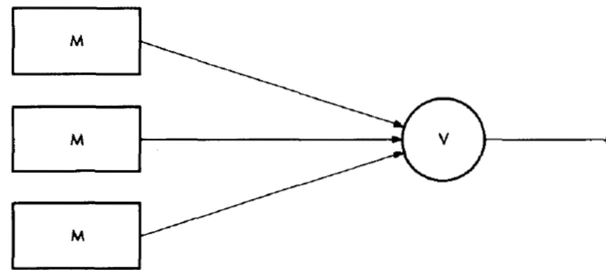

Figure 4.11. Original TMR presented by Von Neumann [Neumann 56], [Lyons 62]

Delving more into the concept of TMR, the voting circuit should be described. The majority voter is a triple OR gate driven by the output signals from three AND gates. Inputs to the AND gates are three different pairs, being the combinations of outputs from the modules. As the best and most common illustration for the TMR is replication of a d-type flip flop, it will serve as an example for explaining the majority voting. It is presented in the Figure 4.12. It is clear that whenever one output differs from the others, the voter output will be in fact the effect of majority voting, that is, the output that is equal for at least two modules. If all flip flops in the design are changed to the shown solution, the design can be considered tolerant for SEUs in the flip flops. However, care has to be taken also of the combinational parts of such circuit, which can also be affected and in turn cause errors in the sequential modules. This will be treated in next sections.

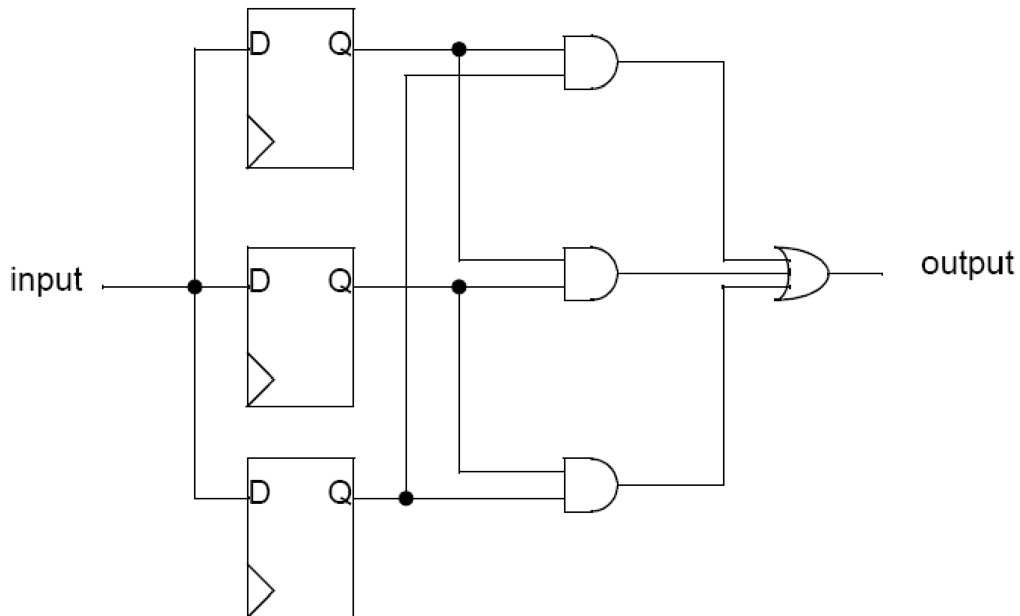

Figure 4.12. TMR with voting circuit for a d-type flip flop [Hanbic 02]

Triple Modular Redundancy can be applied at different levels in the design. One approach is device redundancy, where the whole circuit is triplicated and outputs are compared in the voter. Such solution provides very good protection against faults, however, it is very expensive. With new methods for SEU mitigation, TMR on the device level does not give a good effectiveness, considering the chip area used. A better trade-off between chip area and performance is provided with the module redundancy approach. In this case modules that are most vulnerable to SEUs or crucial for the design's reliability can be chosen for replication. The disadvantage in the presented two approaches is that discovering the error at the output provides sometimes insufficient protection. In state machine based systems if the voter signals a fault, the system most probably goes into an undesired state. This is very difficult to overcome and results in system reset, because it is not possible to define the correct state. A more accurate, yet also more complicated, approach is TMR on the gate level. Figure 4.13a. presents an exemplary circuit fragment, consisting of sequential and combinatorial parts. SEU mitigation in this case is done by triplicating not only the sequential elements but also the combinational ones, which can be seen in the Figure 4.13b. This is based on the fact that in both types there is a risk of SEU initiated faults.

Placing voters between sequential logic elements guarantees synchronisation and thus maintaining the correct state of the whole device. Since voting circuits can also be susceptible to SEUs, Figure 4.13c. shows the complete solution, where they have been replicated as well. Such design allows for immediate detection of errors and feedback paths provide fast correction of the current state. TMR applied in this fashion is much more effective and also more flexible in terms of area consumption than the redundancy on device level.

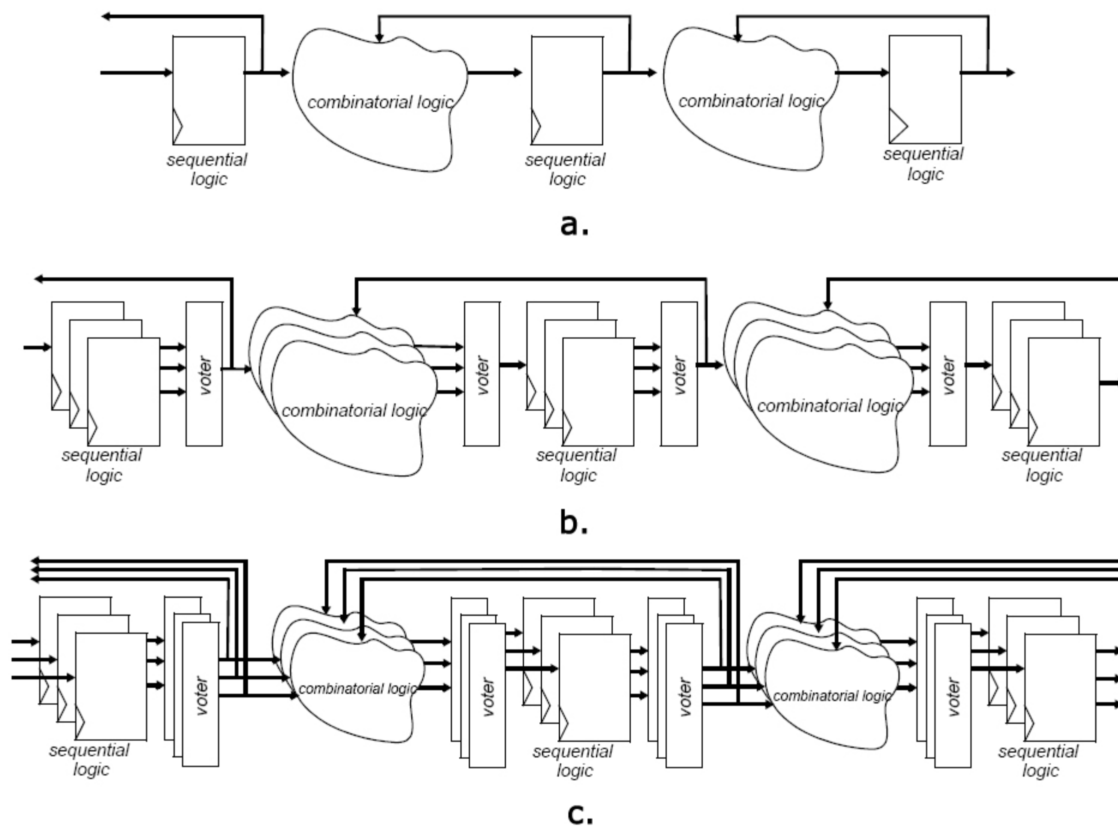

Figure 4.13. Triple Modular Redundancy at gate level: a. original circuit, b. TMR for sequential and combinational logic with feedback paths and c. TMR for sequential, combinational and voter logic [Hanbic 02]

4.1.3f Partial Replication

Although very efficient, Triple or Double Modular Redundancy require very large area overhead, which greatly increases the total system costs. An alternative approach for

logic circuits has been presented in [Mohanram 00], where the concept of partial replication of modules is described. The rationale is simple - keeping satisfactory level of radiation tolerance saving precious space on silicon wafer. This is done by taking advantage of the asymmetric vulnerability of different gates to SEU. By applying an iterative algorithm firstly the nodes that are most susceptible to soft errors are indentified. Then, according to the desired area constraints, the nodes with the highest probability of suffering from a SEU are chosen for replication, just as it is done in the case of DMR or TMR. The results achieved on some test circuits show that, on average, using 20 % area overhead gives 58.1 % reduction of SEU. Generating 50 % area overhead results in even 88.3 % SEU reduction.

Estimating SEU susceptibility of logic nodes

[Mohanram 00] describes a detailed calculation for SEU vulnerability of a node in a circuit. For node *n* soft error susceptibility with respect to latch is given by Formula 2:

$$l = R_{SEU}(n) \cdot P_{sensitized}(n,l) \cdot P_{latched}(n) \quad (2)$$

where:

- $R_{SEU}(n)$ is the rate at which a Single Event Upset deposits enough energy to change the logic state at node *n*. One way of calculating this coefficient is the neutron cross-section method. Neutron SEU cross-section is defined as the probability that a neutron of energy E_x can induce an upset in a device. $R_{SEU}(n)$ is defined in the following way:

$$R_{SEU}(n) = \int_{E_{min}}^{\infty} \sigma_{nSEU}(E_x) \left(\frac{dN}{dE_x} \right) dE_x \quad (3)$$

  where σ_{nSEU} is the neutron SEU cross-section of the device and dN/dE_x is the differential neutron flux.

- $P_{sensitized}(n,l)$ is the probability that node *n* is functionally sensitized to latch *l*, which depends on the pattern for primary inputs.

- $P_{latched}(n)$ is the probability that the SEU is captured in latch *l*. Since the current pulse can be latched only when it arrives at the latch during the latching window, this probability depends on the width of the pulse. It can be computed comparing the width of the pulse for different particle energies with the total clock period used in the device.

A sample circuit schematic by [Mohanram 00] with the gates in bold chosen for replication is shown in the Figure 4.14.

Figure 4.14. An example of partial replication scheme for a sample logic circuit [Mohanram 00]

4.1.3g Periodical Refresh

A very simple approach to radiation tolerance from a system's point of view is applying a periodical refresh mechanism. As described in [Bezerra], one can mitigate SEUs if a counter is added to a design, which determines reprogramming of a device. This idea was implemented for an FPGA system in the work mentioned above. The idea is to choose the appropriate frequency for refresh and reset the system every time the counter has the zero value. Refresh is performed no matter if any SEUs have occured or not but it helps maintaining proper functioning of the device. Such approach enables for example going back to a legal state for a state machine. This concept is also very economical considering the area usage on the manufactured chip.

5 Design of a Radiation Tolerant Readout System for a Neutron Detector

5.1 Project Overview

The practical part of this thesis was to design, implement and test an integrated radiation tolerant controller which would act as a readout system for an integrated neutron radiation detector. The detector was implemented as an asymmetric SRAM memory programmed with a pattern. The idea behind such design is that SRAM memories are proved to be vulnerable to Single Event Upsets (SEUs). Single Events can cause a bit flip. If the memory is programmed with a predefined bit pattern, each change from this pattern can be considered as an occurence of a SEU. In this way radiation dose can be estimated. The readout system was implemented as a finite state machine (FSM). Its aim is counting the number of ones in the bit stream read from SRAM memory which express the number of SEU occurences. After reading the whole memory the number of SEUs is sent in a special frame, the SEU counter is reset and memory readout begins again. The whole system can be connected to a PC using serial transmission. Such approach is simple and does not require as much resources as more complicated solutions, for example based on microcontrollers or microprocessors. The special feature of the designed readout system was an implementation of some Single Event Upsets mitigation techniques. To improve the reliability of the state machine Gray codes were used to explicitly define the states. Moreover, Hamming codes were implemented to protect the state register. This ensured that the FSM does not get stuck in an illegal state and provided some basic Error Detection And Correction (EDAC) capability. With the used Hamming codes all single errors in the state register can be corrected and the FSM is refreshed whenever a double error occurs in the state register. The block diagram of the complete system is presented in the Figure 5.1. It can be seen that the controller part connects with the SRAM based neutron detector with two bidirectional buses, one for data and the other one for memory address. Apart from that, the *WR* (Write) and *OE* (Output Enable) signals are sent from the readout part.

Additionally, two SPI (Simple Peripheral Interface) blocks are connected. One of them is a temperature sensor, connected with the following signals: *TempSPI_Clk* (Clock signal), *TempSPI_In* (data signal) and *TempSPI_CS* (Chip Select signal). The other block is the ADC (Analog to Digital Converter), which drives the special RadFET (Field Emission Transistor) device, serving as a gamma radiation detector. Here the following signals are used: *SPI_Clk* (Clock signal), *SPI_In* (data signal) and *SPI_CS* (Chip Select signal). Furthermore, the *RadFETSwitch* signal is needed in order to switch the transistor on and off, since it needs some time to accumulate the charges generated due to radiation. Also some LED diodes are connected to the readout module for monitoring the state of the system. All the signals are described in more detail in further sections of this chapter.

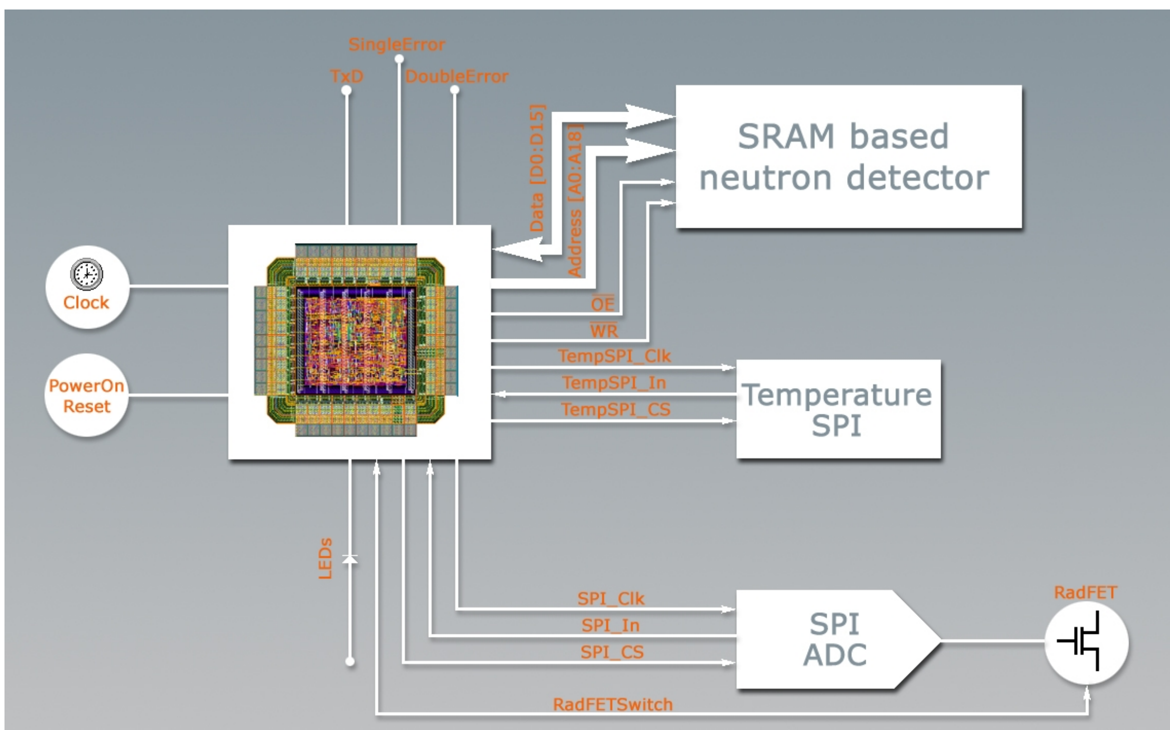

Figure 5.1. Block diagram of the complete system

The advantage of the proposed solution is that it is targeted as an Application Specific Integrated Circuit (ASIC) implementation. This enables integration with the radiation detector on a single chip. Thanks to used system level radiation tolerance schemes, SEU protection is achieved with no interference with the technology.

5.2 Possible Applications in the Radiation Environment

The radiation tolerant finite state machine is supposed to serve as a controller for an integrated neutron radiation detector, manufactured in the same silicon technology. Both circuits are designed to work together on one chip. The detector is a special SRAM memory, very sensitive to SEU, which is programmed with only zeros or only ones, depending on the approach. The controlling part reads the memory continuously and counts the number of bits that differ from the original pattern. Such bits are considered flipped by a SEU. By counting these occurences the Single Event Upsets' rate can be obtained. Making the controller radiation tolerant makes possible the implementation of both parts on a single silicon chip, which significantly reduces the area usage and thus total costs of the project. The implementation was written using VHSIC Hardware Description Language (VHDL).

5.3 Used Tools

Several software applications were used throughout the project design flow. Firstly, behavioural description in VHDL was opened in Aldec Active HDL. This program served as a compiler to verify the VHDL code and as a simulation tool to test the funcionality of the state machine before and after layout (pre-layout and post-layout simulations). Later the code was transferred to Xilinx' ISE 8.1 environment in order to perform an exemplary synthesis process. Although the state machine design was targeted as an ASIC implementation, different versions of the code were tested for synthesizability for the Xilinx Spartan-3E FPGA platform. This served very well for comparison of area usage for consecutive versions of the design.

Implementation of the design after it has been tested and simulated was performed using Cadence design environment. Firstly, BuildGates software was used in order to synthesise the project again. However, here the proper ASIC technology was used, namely version 3.70 of AMS 0.35 μm technology for silicon. BuildGates generated a structural Verilog file which was later used for Place&Route process.

Place&Route was done using Cadence Encounter software. After specifying floorplan, rings for power and ground connections were routed. Then the design was placed toghether with the filler cells to avoid empty spaces on the chip. Finally, routing was performed together with connections to pads. Then the GDSII (Graphic Data System) stream file was extracted and further used in Cadence's Virtuoso Layout Editor.

Cadence Virtuoso Layout Editor served to generate the technological layout of the design. Also Design Rule Check (DRC) and Layout Versus Schematic (LVS) was performed using this software. After verification the device can be manufactured and is ready for testing.

A diagram illustrating all tools used in the design path is presented in the Figure 5.2.

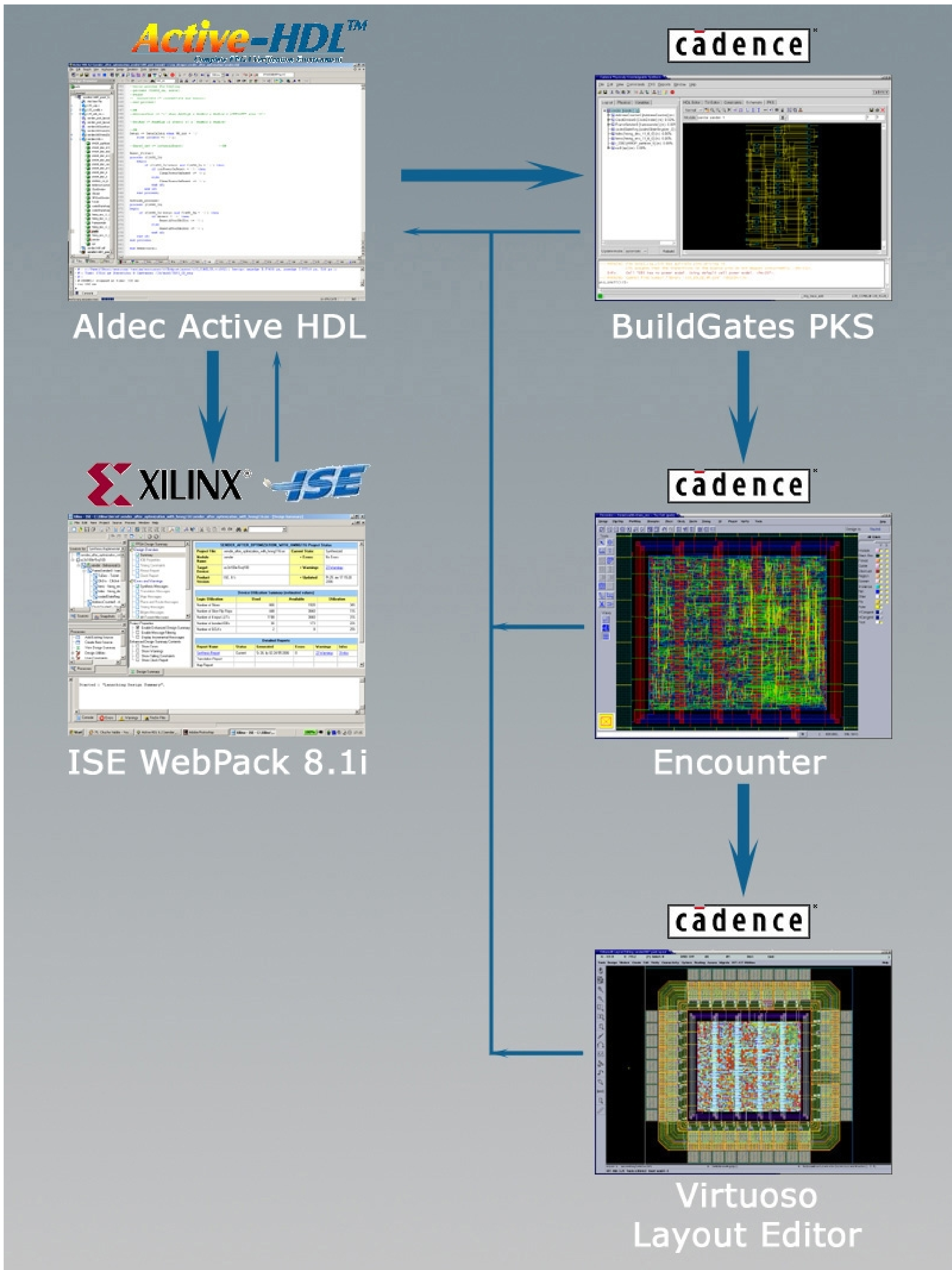

Figure 5.2. Diagram presenting tools used in the design path

5.4 Readout System Description

The basis for radiation detector controller was a finite state machine design which was supposed to be made radiation tolerant. Firstly, the state machine itself will be described. Then applied radiation mitigation techniques will be presented.

5.4.1 Design Overview

The SRAM based radiation detector is controlled by a radiation tolerant readout system. This project is concerned with designing a simple module being capable of reading the memory contents and reprogramming this memory. Since the detector is designed as an asymmetric SRAM being programmed with only zeroes, being sensitized to bit flips caused by Single Event Upsets, the controller counts all occurences of 1 in SRAM. Asymmetric Static Random Access Memory (ASRAM) can be designed in one of two ways, with an easier transition either from high to low state or conversly. In this project it is designed as ASRAM-1, which is optimized for storing a logic '1'. In such an approach it is more reliable to store a high state and the device is vulnerable to SEUs when it is in a low state. This means that if such a memory is programmed with a logic '0', it is very likely that a SEU generates a bit flip. In this way it is easier to detect radiation than with a symmetric memory. All such bit flips are considered a direct effect of a SEU, thus the controller unit outputs the number of SEUs over some period of time. The readout system was supposed to be designed from behavioral description to ASIC layout ready to be manufactured.

5.4.2 Detector Readout Design

Readout system has been implemented as a finite state machine. Its operation is based on 28 states used for reading the memory contents of the SRAM detector, counting the SEU occurrences and reprogramming the memory if needed. Moreover, it transmits the gathered data using EIA-232 serial transmission. Additionaly, the data is sent with a calculated Cyclic Redundancy Check checksum. In order to achieve radiation tolerance all the states are defined explicitly and encoded with Gray code. Furthermore, Hamming codes are used to protect the state registers, not only in the top modules but in the modules lower in the hierarchy that use state machines.

First step in designing the readout system was writing the behavioural description in VHDL using Aldec Active HDL environment. This software served as a platform for coding, debugging and simulating the design. After verification of the behavioural model the project was moved to synthesis tools. Xilinx ISE and Cadence BuildGates were used. The former was used to verify the Real Time Logic schematic and to estimate the area usage in FPGA, in this case in Xilinx Spartan 3E module. The latter served as a major synthesis tool for further design stages. Using BuildGates, the structural Verilog file was created and the system was verified for correct connectivity. After assuring that all modules have been synthesized properly, the design was moved to Cadence Encounter platform, which served as a tool for the Place & Route process. Here the floorplan was specified together with power planning and pads placing. After placing all the design's modules routing was performed. Then SDF file was extracted for post-layout timing simulations. Moreover, GDS file was created, containing information for the final layout in the next step. The next design stage was done using Cadence Virtuoso Layout Editor. Here the GDS file was imported in order to obtain the technological layout of the system. Additionaly, verification of design rules was performed, together with Layout Versus Schematic analysis, using imported structural Verilog file for the schematic generation.

5.4.3 System Description

This section describes the individual modules of the readout system starting from the top module, which is sender.vhd, and then the ones lower in the design hierarchy. The project's structure is illustrated in the Figure 5.3, which is a screenshot from the simulation view in Active HDL environment. Table 5.1 presents short descriptions of all modules.

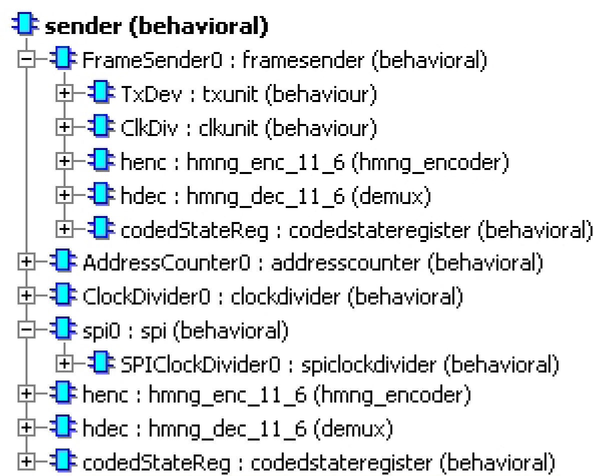

Figure 5.3. Design structure as seen in Active HDL environment

Table 5.1. States description

sender	Top module, includes state machine for memory readout, SEU counter and sender of frames with the counted SEUs
framesender	Module for sending frames
TxUnit	Transmit unit, sends given data bit by bit
AddressCounter	Module for determining the memory address for reading and writing
ClkUnit	Module for dividing clock signal, used by framesender module
ClockDivider	Module for dividing clock signal, used by sender module
hmng_enc_11_6	(11,6) Hamming code encoder, used by framesender and sender modules
hmng_dec_11_6	(11,6) Hamming code decoder, used by framesender and sender modules
codedStateRegister	Register for storing the encoded states

SPI	Additional module for reading SPI data
spiclockdivider	Clock divider for the SPI module
uart_lib	UART library package
pck_CRC32_D8	CRC coding package for generating 32 bit CRC checksum from an 8 bit codeword

Top module: sender.vhd

The top module is in fact the state machine used for reading the memory contents, checking it for SEU occurences and sending data frames with the obtained information. It has got 4 input ports, 10 output ports and 2 inout ports, which is illustrated in the Figure 5.4.

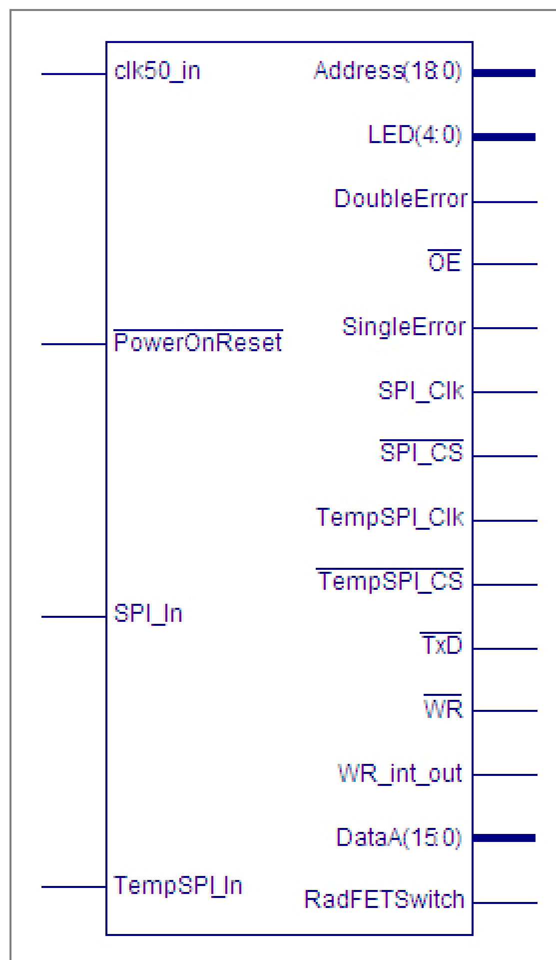

Figure 5.4. Pins in the top module, view from Xilinx ISE 8.1.

Module sender implements 7 modules: ClockDivider, framesender, spi, hmng_enc_11_6, hmng_dec_11_6 and codedStateRegister. Functionality of all those modules is used in the main state machine of the system. The used FSM has 28 states which are described using the Table 5.2.

Table 5.2. State description for the main module of FSM

State name	State function
ST0	reset, initial state
ST1	idle state
ST2 ST3	sending first frame and waiting for acknowledgement of frame sending
ST4delay ST4 ST5 ST6 ST7 ST8	writing memory with the agreed pattern and checking if the whole memory has been written
ST9 ST10	sending second frame and waiting for acknowledgement
ST11	reading data from memory in 3 clock cycles
ST12	data from the inout port DataA is stored
STCheckifSEU	SEUCounter signal is incremented whenever any bit read from memory is 1
	if there were any SEUs detected, FSM goes into ST13a, where the memory is programmed again
ST13 ST14 ST15	checking if whole memory has been read and if not, going back to ST11
ST16 ST17	checking whether periodic information should be sent or not, depending on the additional clocks of the low frequency. Additionaly, determination if a frame with counted SEUs should be sent or not
	if SEUs have been detected: ST18, where number of SEUs is transmitted
	otherwise: ST_SPI1 and ST_SPI2
ST19 ST20 ST21 ST22	waiting for the send acknowledgement and going back to ST11 for further readout of the memory
OTHERS	ST0

The main reset signal of the system is driven by *PowerOnReset* port. *PowerOnReset* is active low and it is connected with the *Reset* signal refreshing the state machine by a Reset_filter process. The *Reset* signal is active high and resets all signals in the state machine. In such case the state is set to ST0. Additionally, reset of the system may be performed after a double error in the state register has occured. This will be explained in the further sections, concerning the radiation tolerance of the design.

Clock signal is connected through *clk50_in* pin. All lower modules that are driven by a clock are connected to this signal. *clk50_in* is used also for generation of additional clock signals in the ClockDivider module. Here the *Clk50Hz*, *Clk9m30* and *Clk10min* signals are generated by some simple clock division processes. They are used in the top module for sending some periodic information in special frame formats. In all Active HDL behavioural simulations *clk50_in* frequency of 10 MHz was used, thus clock had a period of 100 ns.

Counting SEUs in the SRAM memory is done in the STCheckifSEU state. The 16 bit data vector read from the inout *DataA* port is checked bit by bit and the value of *SEUCounter* is a summation of all the '1' bits incoming from *DataA*. Thus, *SEUCounter* counts all ones in the memory. After checking the whole memory, the *SEUCounter* value is transmitted in the special frame on the *TxD* pin.

Module for sending frames: framesender.vhd

Framesender module is responsible for preparing and sending frames with data. It is constructed as a state machine operating on 8 states. Short states description is given in the Table 5.3.

Table 5.3. State description for framesender module.

State	Description
ST0	Idle state, waiting for a signal to begin sending data
ST1	Loading data into buffer
ST2	Waiting for transfer of data from buffer to the sending register
ST3	Sending data
ST4	Checking which part of the frame should be sent, depending on the FrameCounter signal
ST5	Checking which part of the frame is being sent and determination if the CRC should be computed
ST6	Checking if all frame components have been sent and going back to ST1 if necessary
ST7	Checking if whole frame has been sent, if yes going back to ST0
OTHERS	ST0

A frame consists of 12 bytes. It's structure is given below:

```
|Start|FrameLen|FCounter|FrameType|VData 2B|TData 2B|CRC32 4B|
```

where:

- `Start` – byte indicating start of a frame, equal always to x"55" in hexadecimal (01010101 in binary);
- `FrameLen` – length of the frame, by default set to x"06";
- `Fcounter` – number of current frame, 8 bits;
- `FrameType` – indication of the contents of current frame, 8 bits, the following values are used:
  - x"01" – introductory frame, sent at the beginning of transmission, set in ST2;
  - x"02" – data frame, set in ST9;
  - x"03" – frame with counted SEUs, set in ST17;
  - x"04" – live frame, indicating that no SEUs were detected and that the frame contains data from the SPI sensors, set in ST_SPI2;
- `VData` – Voltage Data, obtained from the RadFET radiation detector, 2 bytes;

- `TData` – Temperature Data, obtained from the SPI temperature sensor, 2 bytes;
- `CRC32` – 4 bytes of calculated CRC32 codeword;

4-bit *FrameCounter* signal is used to distinguish which byte should be sent. Reaching value of 11 causes sending a *SendAck* signal, which means completing the transmission. Data bits are sent using 1-bit serial output *TxD* pin, conforming to the EIA-232 standard. Figure 5.5 presents a screenshot from a simulation performed using Aldec Active HDL simulator. Some exemplary stimuli have been used to illustrate frame sending process in framesender module.

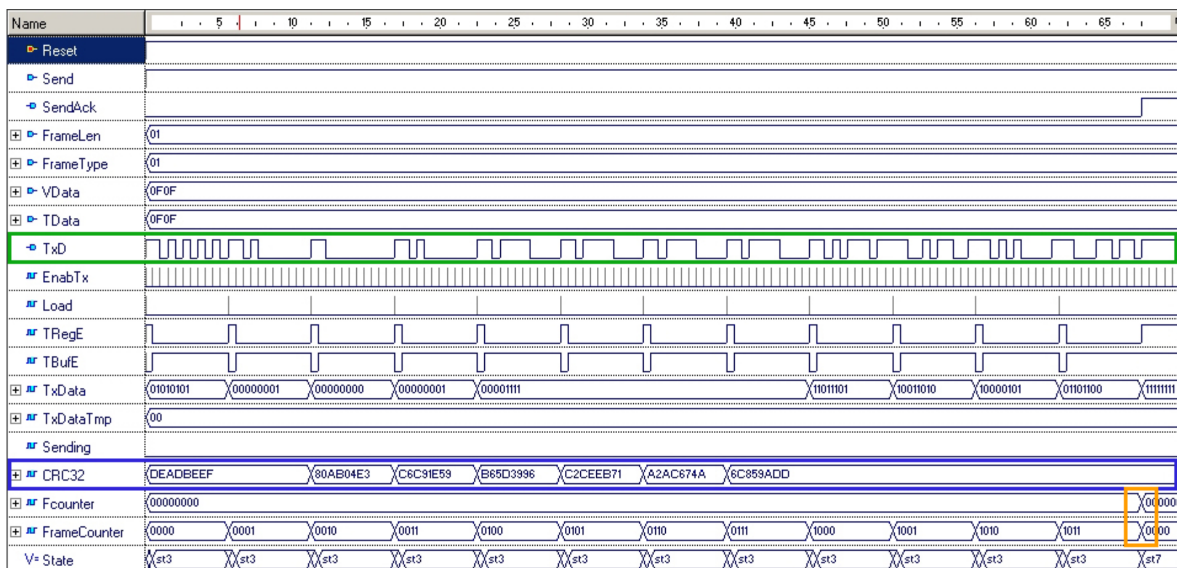

Figure 5.5. Simulation results for framesender module. Green marks the TxD signal, which is the sent frame bit by bit. Blue marks the current values of the CRC32 calculation. Orange marks the counter value which indicates completing the whole frame.

Transmit module: TxUnit.vhd

TxUnit is the transmitter module that is responsible for the actual transmission on the *TxD* serial pin. Whenever the *Enable* signal goes high TxUnit is ready for transmitting a byte of data. Setting *Load* signal to '1' loads data from *DataO* port to transmit buffer and

then to transmit register. After this the loaded data byte is sent bit by bit at every rising edge of the clock. When eight bits, counted by *BitCnt* signal, are sent, an acknowledge signal is sent on port *TRegE*, indicating an empty transmit register. Simulation results from Aldec's Active HDL can be observed in the Figure 5.6.

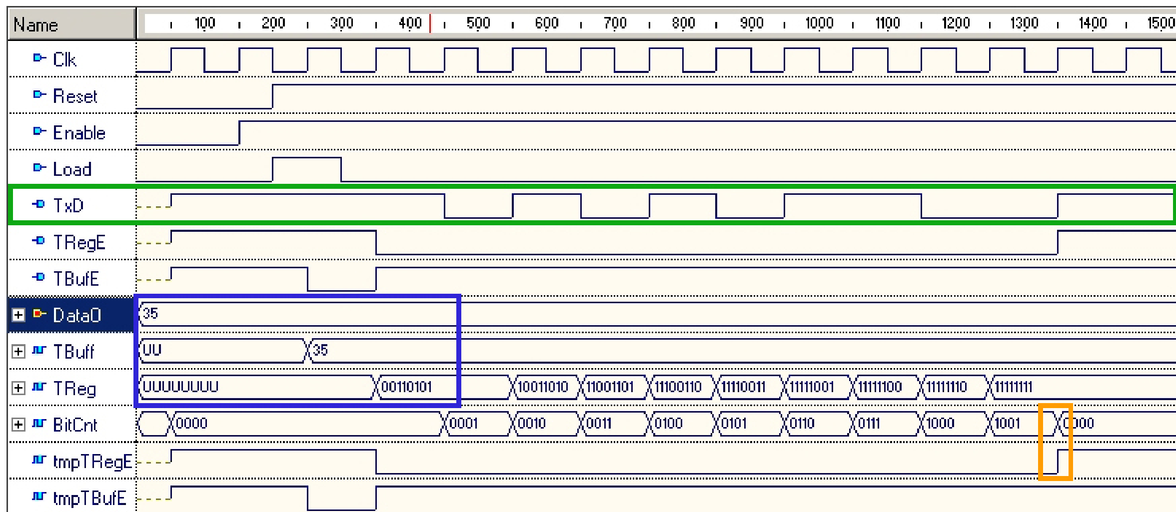

Figure 5.6. Simulation results for TxUnit module. Green marks the TxD with the individual bits from the sent codeword from DataO transmitted bit by bit. Blue shows the translation of input codeword to the transmit buffer and transmit register. Orange marks the bit counter signal, which indicates completing the sending process of data.

The radiation tolerance for state machines, both in the main module (sender.vhd) and in the frame sending module (framesender.vhd) was achieved using some special encoding schemes. In the straightforward approach states are defined as an enumerated type:

```
type S_Type is (ST0, ST1, ST2, ST3, ST4, ST5, ST6, ST7, ST8,
                    ST9, ST10, ST11, ST12, ST13, ST14, ST15,
                  ST16, ST17, ST18, ST19, ST20, ST21, ST22,
            ST13a, ST_SPI1, ST_SPI2, STCheckifSEU, ST4delay);
variable State: S_Type := ST0 ;
```

In such case there is no explicit definition of state values and the synthesis tools chooses the optimal way of assigning bit patterns to each value. In this case, where there are 28

legal states, the synthesizer may choose either a sequential encoding implementation, where 5 bits are sufficient, giving 32 possible states. Then 4 states are left undefined. Similarly, if such state machine is implemented as a one-hot design, there are 28 bits for state encoding, which gives a large number of illegal states (2^{28}-28). If the FSM goes into one of such states it can get stuck. That is why it is essential to use the 'others' statement to provide means for returning to one of the legal states. A simple, yet very effective solution for protecting the state machine is to define all states explicitly [Synplicity 99]. Changing the previously given code into:

```
signal State : std_logic_vector(4 downto 0) := "00000";
constant ST0: std_logic_vector(4 downto 0):="00000";
constant ST1: std_logic_vector(4 downto 0):="00001";
constant ST2: std_logic_vector(4 downto 0):="00011";
constant ST3: std_logic_vector(4 downto 0):="00010";
...
```

results in an exact implementation during sythesis. This ensures that the 'others' statement distinguishes the allowable states. Next aspect to be taken under consideration is the optimal encoding of state values. The most common approaches for assigning states are one-hot, one-cold, sequential or Gray code approach. In this project one-hot or one-cold solution would be impractical, since there are 28 states in the top module. Each codeword for states would thus have 28 bits. The sequential approach is the most straightforward. However, Gray encoding provides a more reliable design, since the consecutive states differ in only one bit position. If neighbouring state values have more than one differing positions, during state changes the FSM can go to illegal state if the transitions of different bits represented by flip flops are not synchronized [Synplicity 99]. Using Gray codes provides safer transitions between states, since in each state transition only a single bit is changed. The state values for the top module are given in Table 5.4 and for the framesender module in Table 5.5.

Table 5.4. State encoding for sender module.

State	Encoding	State	Encoding
ST0	"00000"	ST14	"01001"
ST1	"00001"	ST15	"01000"
ST2	"00011"	ST16	"11000"
ST3	"00010"	ST17	"11001"
ST4	"00110"	ST18	"11011"
ST5	"00111"	ST19	"11010"
ST6	"00101"	ST20	"11110"
ST7	"00100"	ST21	"11111"
ST8	"01100"	ST22	"11101"
ST9	"01101"	STCheckifSEU	"10011"
ST10	"01111"	ST4delay	"10000"
ST11	"01110"	ST13a	"11100"
ST12	"01010"	ST_SPI1	"10100"
ST13	"01011"	ST_SPI2	"10101"

Table 5.5. State encoding for framesender module.

State	Encoding
ST0	"00000"
ST1	"00001"
ST2	"00011"
ST3	"00010"
ST4	"00110"
ST5	"00111"
ST6	"00101"
ST7	"00100"

After defining states explicitly the next step was to apply Hamming codes to the state register. The encoder and decoder were designed for the top module, where 28 states were coded with 5 bit vectors. In order to achieve single error correction and double error detection capability, a (10,5) Hamming code was needed. A coding table for this code is

presented as the Table 5.6.

Table 5.6 Constructing a (10,5) Hamming code

Codeword:	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9	n_{13}
Parity and data:	p_1	p_2	d_1	p_3	d_2	d_3	d_4	p_4	d_5	c
Original data:			✓		✓	✓	✓		✓	Additional parity
p_1			✓		✓		✓		✓	
p_2			✓			✓	✓			
p_3					✓	✓	✓			
p_4									✓	

The fourth parity bit (p_4) is needed to include the fifth data bit (d_5). Otherwise it would be checked only by the first parity bit (p_1). Besides, following the described rules for constructing Hamming codes forces a (10,5) scheme. This enables correct syndrome calculations in the decoder and distinguishing the flipped bit in the codeword. The additional parity bit is used for calculating the total parity over all bits in the codeword, including all parity bits. Such scheme provides functionality for detecting and correcting single errors and detecting double errors. Although using 5 parity checks is enough to obtain the desired EDAC capabilities, it turned out during the synthesis phase of the project that the parity digit being in fact duplication of one of the data bits is not routed. In the structural Verilog file one bit always obtained the value UNCONNECTED, no matter if the parity check bit was altered in some ways. One way was to use some dummy logic functions producing just the value of the parity check or to route the signal through some additional signals. However, nothing caused the proper connecting of the whole coded state bus. The only way that proved to be successful was using an additional data bit, which made the codeword longer (11 not 10 bits), but the connections were synthesized correctly. In the *State* values the additional bit was realised by just adding a zero in the front of the binary vector. With such approach the Hamming code changed into a (11,6) type.

Hamming encoding was applied to the state register. The previous definition of states stays the same, however, the state machine now works using the *decodedState* signal. Original, not encoded version was:

```
if (Reset='1') then
      ...
      State <= ST0;
elsif (clk50_in'event and clk50_in='1') then
      case State is
            when ST0 =>
            ...
            when others => State <= ST0;
...
```

both in framesender and sender modules. In the version with encoded states everything stays as originally but the case statement changes to consider the *decodedState* signal, not the original *State* signal:

```
      if (Reset='1') then
      ...
      State <= ST0;
elsif (clk50_in'event and clk50_in='1') then
      case decodedState is
            when ST0 =>
            ...
            when others => State <= ST0;
...
```

This is possible after adding additional signals working with additional components for Hamming coding. These modules are summarised in the Table 5.7.

Table 5.7. Modules for Hamming coding used in the design

Name	Function	Inputs	Outputs
hmng_enc_11_6	encoder module	input (6 bits)	output (11 bits)
hmng_dec_11_6	decoder module	input (11 bits)	output (11 bits) singleError (1 bit) doubleError (1 bit)
codedStateRegister	state register	input (11 bits)	output (6 bits)

The codedStateRegister module is just a transmit buffer. This also proved to be necessary in the synthesis procedure. The Hamming decoder module outputs not only the decoded signal bus but also single error and double error flags. Mapping of all signals used in the Hamming modules is presented in the Table 5.8. This applies to both framesender and sender modules.

Table 5.8. Signal mapping in the Hamming modules

Module	Signal	Direction	Mapped signal name
hmng_enc_11_6	input	input	State
hmng_enc_11_6	output	output	codedStateIn
codedStateRegister	input	input	codedStateIn
codedStateRegister	input	input	codedStateOut
hmng_dec_11_6	input	input	codedStateOut
hmng_dec_11_6	ouput	output	decodedState
hmng_dec_11_6	single_error	output	sglerr
hmng_dec_11_6	double_error	output	dblerr

The 5 bit *State* signal, which was originally the only signal for storing state values in the new approach is encoded with Hamming code and as an 11 bit signal *codedStateIn* goes through the codedStateRegister. Then, the unchanged 11 bit vector *codedStateOut* is decoded in the decoding module and as a 5 bit *decodedState* signal is used in the state machines. In other words, the next state register is being coded before it is stored in the state register. Then it is decoded before being used in the state machine [Lima]. This scheme is used both in framesender and sender modules.

Additional *sglerr* and *dblerr* signals are outputs from the Hamming codes decoder. The single error signal is mapped to the output port *SingleError* and is just an indication that a single error has occured in the state register. As it was explained, the single errors are corrected with the applied scheme and have no influence on functioning of the state machines. However, the double error signal, being mapped to the *DoubleError* port, serves not only as an indication but also triggers refresh of the system. Whenever a double error occurs in the state register, it cannot be corrected, thus the state machine goes back to the initial state. The refresh process is shown below:

```
Refresh_process:
process (Clk50_In)
begin
      if (Clk50_In'event and Clk50_In = '1') then
            if dblerr = '0' then
                  ResetAfterDblErr <= '0';
            else
                  ResetAfterDblErr <= '1';
            end if;
      end if;
end process;
```

The active high main *Reset* signal is driven into high state whenever the *ResetAfterDblErr* signal goes high, as shown below:

```
Reset <= '1' when (... or ResetAfterDblErr = '1') else '0';
```

5.5 Design Path

This section describes the design path followed during the work on this project. The whole implementation from behavioural description in VHDL to technological layout ready for manufacturing is presented. Also the results of the post-layout simulations are given. The complete design flow is presented in the Figure 5.7.

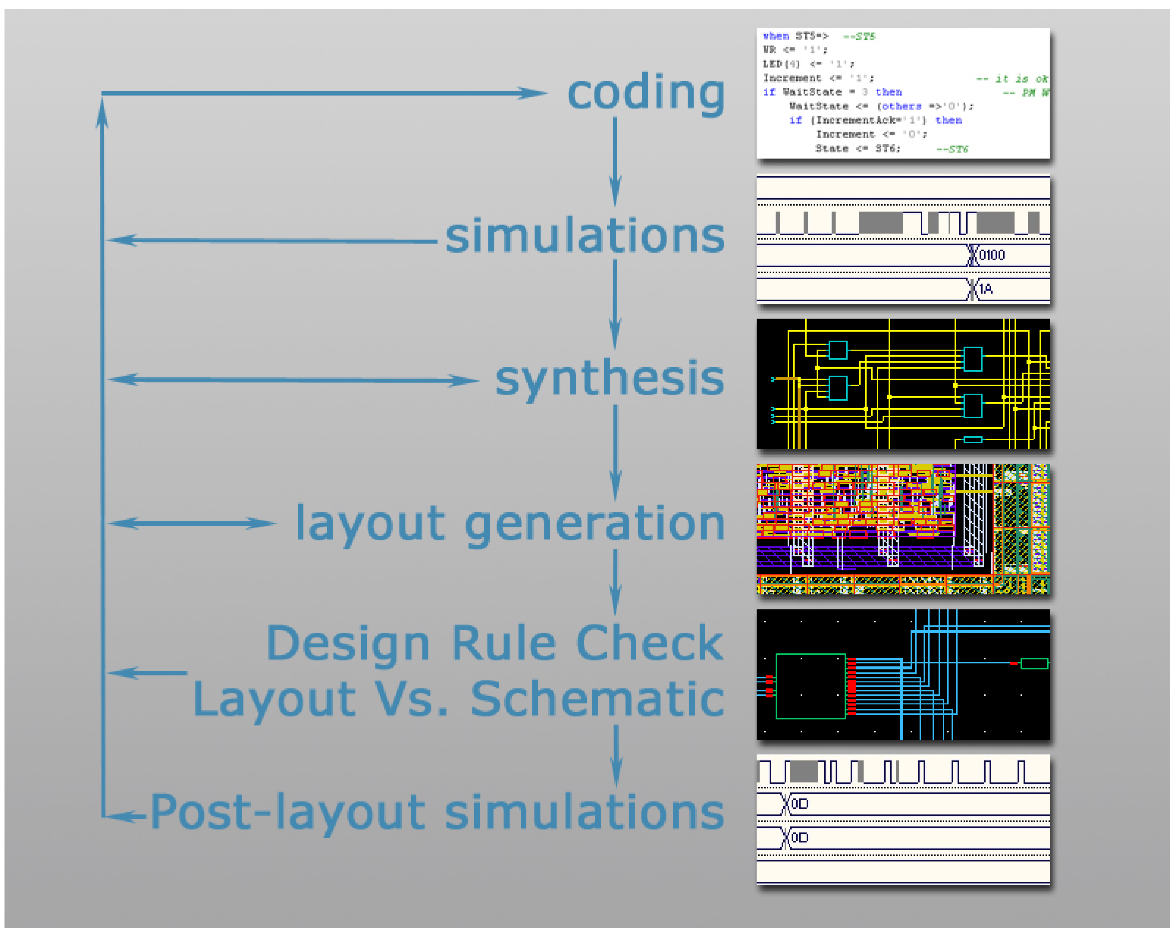

Figure 5.7. Complete design flow for the readout system

5.5.1 Behavioural Description and Simulations in Aldec Active HDL

The described project was encoded using Aldec Active HDL environment. After debugging, the behavioural description was used for simulations. Firstly, a simulation assuming that no errors occur in the state register was performed. Main system clock was set to 10 MHz, which corresponds to the period of 100 ns. *PowerOnReset* was set to binary '0' for the first 200 ns, and after that to binary '1'. Two clock periods ensure resetting all signals. To simulate a data vector read from memory, the inout *DataA* port was set to the input direction and forced with an exemplary bitstream ("0000000100000000"). Thus, the readout system should count 1 SEU per line of memory. The *MemoryDensity* constant in the AddressCounter module was changed to 256 for simulation's sake, since its purpose was only to verify the functioning of the circuit and this value was sufficient to see if the SEUs are counted. Plotted simulations results for the sender module with the described stimulators are shown in the Figure 5.8.

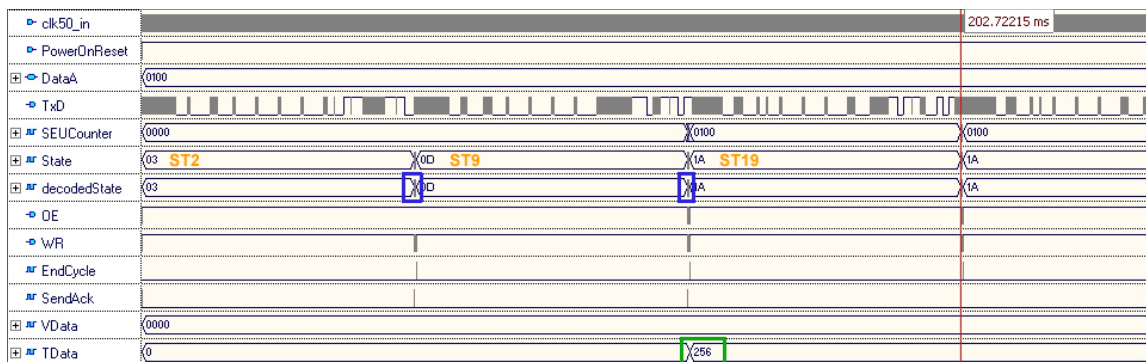

Figure 5.8. Plot of Aldec's Active HDL simulation of top module sender. The regions marked with blue indicate memory operations, in this case very short, since the memory size has been set to 256 bytes. Green shows the correct value of SEUCounter transmitted on the TData signal and then serially on the TxD pin.

SEUCounter is incremented to hexadecimal "0100" (decimal 256), which is equal to the number of lines in memory. Each line contained one forced SEU bit flip. Thus, *SEUCounter* indicates the number of errors correctly. *TData* signal then takes the number of SEUs and this is sent on the *TxD* pin. As an acknowledgement of completing the

sending phase the *SendAck* signal goes high and the state machine goes back to reading the memory contents. At this stage the *SEUCounter* is reset to zero.

The next step was to simulate errors in the state register. The top module vhdl file was modified to contain 11 bit *error* and *errorState* signals. The design was changed so that *errorState* was constructed through XOR operation of *error* and *codedStateOut* signals and driven as an input to the decoder module in place of *codedStateOut* signal. The *decodedState* signal was supposed to stay the same in the case of a single error or be reset to zero in the case of a double error. Simulation results plotted in Active HDL are presented in the Figure 5.9.

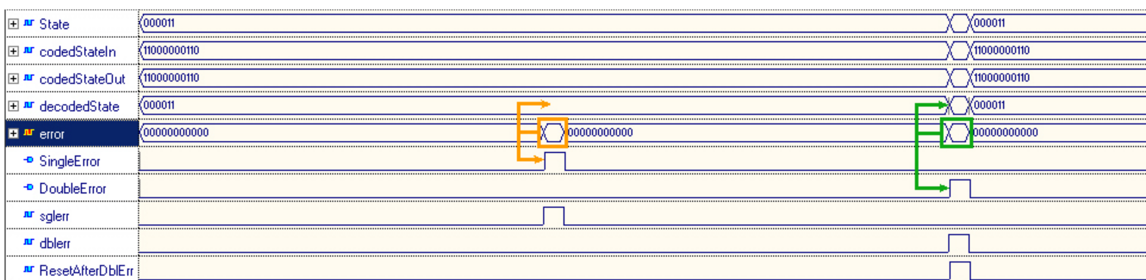

Figure 5.9. Plot of Aldec Active HDL simulation of top module sender with additional error vector. Orange markers indicate a single error, which is transparent for the state register. Green markers indicate a double error, which in turn results in refreshing the state machine. Both types of errors trigger the appropriate signals for observing the errors in the state register.

Table 5.9. shows values forced for the error vector.

Table 5.9. Error values

Time	Value
0 µs	"00000000000"
2000 µs	"00000001000"
2100 µs	"00000000000"
4000 µs	"00010010000"
4100 µs	"00000000000"

The plot shows that *decodedState* signal is not affected by a single error. Furthermore, a double error causes the *ResetAfterDblErr* signal to go to a high state, which has an effect of an immediate jump to the initial state ST0. The *error* and *errorState* signals were used just to verify that the Hamming coding works as expected and are not included in further stages of the design.

5.5.2 Synthesis in Cadence BuildGates Physically Knowledgeable Synthesis

After testing the behavioural description of the readout system, the project was moved to Cadence BuildGates PKS (Physically Knowledgeable Synthesis) tool. Here the synthesis was performed, an effect of which was the structural Verilog file, being a netlist of the design. First the correct technological libraries were loaded to conform to the rules of AMS v3.70 CMOS technology. Then the VHDL source files were read and the RTL schematic was generated. At this stage it was verified that the sender module was recognised as the top level of the design hierarchy. The next step was optimization according to the technology rules. Medium effort was chosen for placement of instances together with setting priority on area. Optimization produced a schematic view of the design, which was carefully analyzed. After optimizing the project, the structural Verilog file was generated. Furthermore, an area report was generated, which showed areas occupied by the whole chip and also by the individual modules. The whole readout system, not including peripheral cells, should have the area equal to 297915,80 μm^2, which is a square with side length equal to 545,81 μm. This calculation was performed using information included in the attached AMS libraries. It was supposed to be compared with the core size of the generated layout in further design stages.

5.5.3 Synthesis in Xilinx ISE version 8.1i

To compare dimensions of radiation-tolerant and unprotected version of the design, it was synthesized in Xilinx ISE. Synthesis for both designs was performed for a Xilinx Spartan XCS200FT256 FPGA, since this type of device is available for testing at the University. Speed Grade of -4 was used. Table 5.10 shows a design summary for the unprotected design, whereas Table 5.11 presents the same data for the radiation-tolerant version.

Table 5.10. Design Utilization Summary for unprotected design

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	642	1920	33%
Number of Slice Flip Flops	469	3840	12%
Number of 4 input LUTs	1157	3840	30%
Number of bonded IOBs	34	173	19%
Number of GCLKs	2	8	25%

Table 5.11. Device Utilization Summary for radiation-hardened design

Device Utilization Summary (estimated values)			
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slices	666	1920	34%
Number of Slice Flip Flops	448	3840	11%
Number of 4 input LUTs	1198	3840	31%
Number of bonded IOBs	36	173	20%
Number of GCLKs	2	8	25%

The protected design occupies negligibly more resources than the unprotected one. Only less than 1 % more slices are used, the same applies to input LUTs. However, unprotected design uses almost 1 % more of slice flip flops. These results prove that making the design more radiation-tolerant by means of adding coding techniques to state machines does not require sacrificing a lot of resources. Hamming codes prove to be an efficient way for protecting the state register. It should be noted that the Xilinx synthesis tool overrides the state values definition. Thus, Gray codes were not always used because during synthesis

the software chooses optimal approach for FPGA applications. For example with changes applied to the behavioural VHDL description, a state machine could be coded either with sequential, Gray or even one-hot coding. Chosen scheme can be viewed in the detailed synthesis report.

The maximum operating frequency for both designs was calculated to be 36.144 MHz, thus the minimum period equals 27.667 ns.

5.5.4 Cadence FirstEncounter Place & Route Environment

Encounter is a part of Cadence package responsible for Place & Route process. When the design is imported user loads the structural Verilog file together with LEF (Library Exchange Format) technological libraries, TLF (Timing Library Format) or LIB timing libraries, IO (Input Output) file, timing SDC file and a set of constraints. All these files can be loaded using one configuration file (.conf). The readout system was loaded using files and constraints listed in the Table 5.12.

Table 5.12. Parametres used for loading project into Encounter environment

Type	Value	Comment
Netlist	sender.v	Structural Verilog
LEF files	c35b4.lef CORELIB.lef IOLIB_4M.lef	Technology rules for AMS v3.70 CMOS technology with 4 metallization layers
Timing libraries	c35_CORELIB.lib c35_IOLIB_4M.lib	Timing constraints for the used technology
IO file	sender.io	IO file with pads definition
Core aspect ratio	1.0	Priority of width/height aspect ratio of 1, thus a square-shaped core
Core utilization	0.85	Core utilization factor
Core to left/right/top/bottom	300/300/300/300 μm	Spacing between core and periphery cells
Power nets	vdd! vdd3r1! vdd3r2! vdd3o! VDD	Names from techfiles

Ground nets	gnd! gnd3r! gnd3o! VSS	Names from techfiles

Core was designed to be square-shaped, thus an aspect ratio equal to 1.0 was chosen. This was possible also due to the pads definition choosing equal number of pads on each side of the core. This is included in the IO file. The system, as described in the previous sections, has 54 pins. Together with additional power and ground pins, 56 pads are needed, which was divided as presented in the Table 5.13. Also placement of pads is given, where N,E,W and S represent North (Top), East (Right), West (Left) and South (Bottom) regions of the core, respectively.

Table 5.13. Pads name and orientation

Pad name	Orientation	Pad cell	Pad name	Orientation	Pad cell
CORNER4	SW	CORNERP	CORNER3	NW	CORNERP
CORNER1	NE	CORNERP	CORNER2	SE	CORNERP
DataA14	N	BBC8P	Address18	E	BU16P
DataA13	N	BBC8P	Address17	E	BU16P
SingleError	N	BU16P	Address16	E	BU16P
DoubleError	N	BU16P	Address15	E	BU16P
TxD	N	BU16P	Address14	E	BU16P
VDD	N	VDD3ALLP	Address13	E	BU16P
clk50_in	N	ICCK8P	Address12	E	BU16P
LED4	N	BU16P	Address11	E	BU16P
LED3	N	BU16P	Address10	E	BU16P
LED2	N	BU16P	Address9	E	BU16P
LED1	N	BU16P	Address8	E	BU16P
LED0	N	BU16P	Address7	E	BU16P
OE	N	BU16P	Address6	E	BU16P
WR	N	BU16P	Address5	E	BU16P
DataA13	W	BBC8P	PowerOnReset	S	ICP
DataA12	W	BBC8P	SPI_Clk	S	BU16P
DataA11	W	BBC8P	SPI_In	S	ICP
DataA10	W	BBC8P	SPI_CS	S	BU16P
DataA9	W	BBC8P	RadFETSwitch	S	BBC8P
DataA8	W	BBC8P	TempSPI_Clk	S	BU16P
DataA7	W	BBC8P	TempSPI_In	S	ICP

DataA6	W	BBC8P	TempSPI_CS	S	BU16P
DataA5	W	BBC8P	VSS	S	GND3ALLP
DataA4	W	BBC8P	Address0	S	BU16P
DataA3	W	BBC8P	Address1	S	BU16P
DataA2	W	BBC8P	Address2	S	BU16P
DataA1	W	BBC8P	Address3	S	BU16P
DataA0	W	BBC8P	Address4	S	BU16P
Totally: 56 pads, 14 pads per each side of the chip					

The clock pad was placed in the middle of the top set of pads for optimal clock tree generation. This way minimised the distances to instances driven by clock signal. An explanation to the pads orientation and a simplified chip layout is given in the Figure 5.10, where all the most important regions and structures of the device are presented.

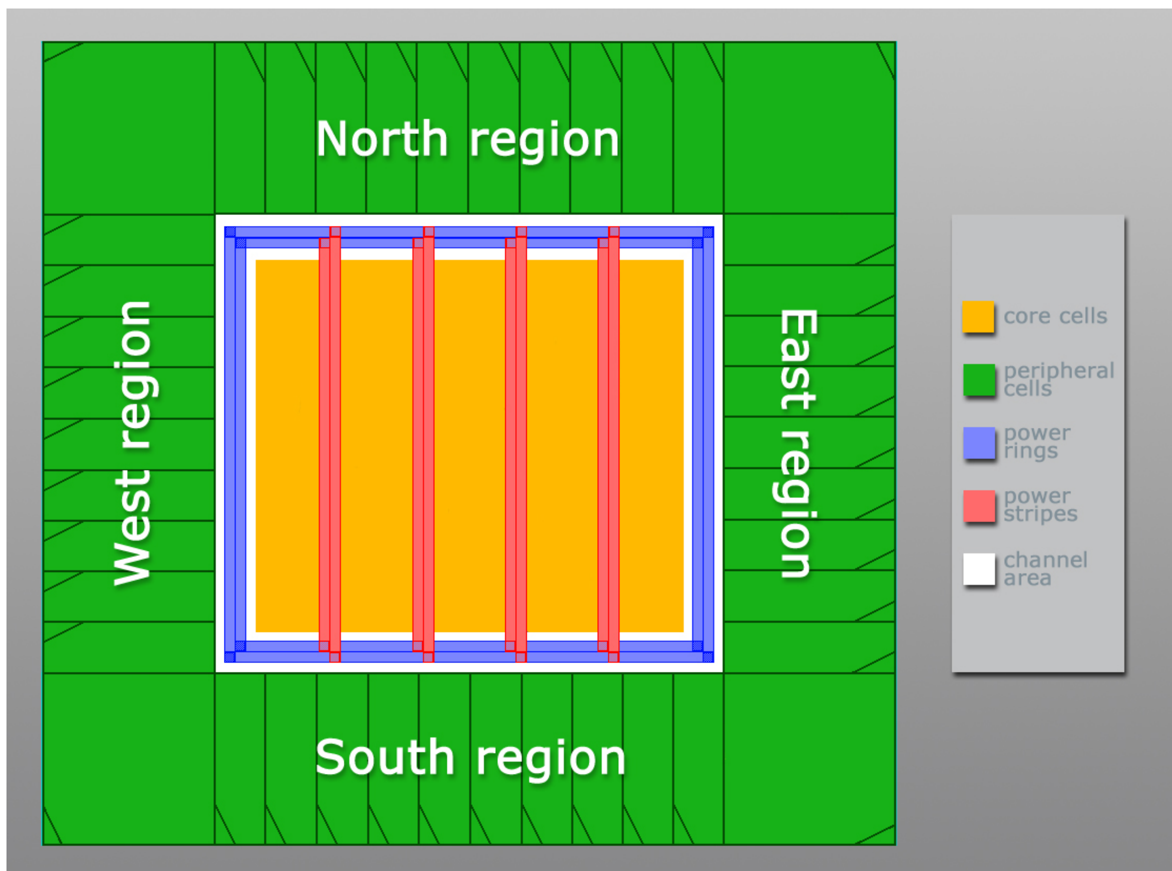

Figure 5.10. Explanation of the most important regions and structures on the designed chip

After importing design first the floorplan was specified. Since it turned out that the core size inside the rectangle defined by the pads is much larger than needed for placing the design, the core dimensions had to be decreased. BuildGates synthesis provided chip dimensions of approximately 550 x 550 µm. Thus, the core size had to be a compromise between the required area for correct placement and the area defined by the periphery cells. The core cannot be too big, since such approach introduces many filler cells and also more delays because the cells are further apart. The core size was thus chosen to be about 800 x 800 µm, which guarantees safe placement of all core cells and also leaves a 300 µm wide channel for power rings and routing to the periphery cells. This will be also discussed in the next sections of this chapter. After defining the floorplan correctly the power planning was performed. This included designing power rings and stripes, assigning power net connections and routing rails for power and ground connections for all instances. Global net connections are summarised in the Table 5.14.

Table 5.14. Global net connections for power and ground pins

Pin name	Connected to Global Net
vdd! vdd3r1! vdd3r2! vdd3o! VDD	VDD
gnd! gnd3r! gnd3o! VSS	VSS
TIEHI	VDD
TIELO	VSS

VDD and *VSS* are power and ground pads, respectively. *vdd! vdd3r1! vdd3r2! vdd3o!* and *VDD* as well as *gnd! gnd3r! Gnd3o!* and *VSS* are pin names specified in technology files. *TIEHI* and *TIELO* pins connect to *VDD* and *VSS* pads, respectively.

After defining power and ground connections the power rings were created. Layers 1 and 2 of metallization were chosen for right/left and top/bottom rings, respectively. Width was set to 20 µm and spacing to 1 µm, according to technology rules. Rings were centered in the channel between core and pads, the width of which was earlier set to 300 µm on all sides.

Vertical power stripes were made with first layer of metallization connected to power rings. Stripes facilitate power and ground connections to instances in the core. Here also width was set to 20 µm and spacing to 1 µm. 4 stripes sets were created at equal distances from one another, starting 150 µm from the core boundaries.

Horizontal power and ground rails were created using SRoute. However, to perform SRoute correctly, endcap cells had to be defined first. These special cells are placed on the right and left sides of the core and act as edge points for rails. 56 such cells were created on both sides. After specifying endcaps SRoute was used to create rails.

After preparing all power features the design was placed. Medium effort was used, as was done during synthesis optimization. Placed instances did not fill the whole core area, leaving some empty spaces. This was partly because the core utilization factor was set to 0.85 and partly because the core area was greatly influenced by the dimensions of the periphery cells. In the used technology each pad has the dimensions of 340 µm x 100 µm. With the used set of pads and 300 µm channel for power rings core size was assigned automatically. It turned out that the design does not occuppy the whole space available. Thus, some filler cells had to be added. AMS technology defines 5 different core filler cells (FILL25, FILL10, FILL5, FILL2, FILL1). Encounter places them automatically, from the largest to the smallest ones, filling all holes in the core. In the designed system 2840 core filler cells were added (360 x FILL25, 386 x FILL10, 559 x FILL5, 986 x FILL2 and 549 x FILL1). Another type of fillers are peripheral filler cells that are used for spaces between pads. Since in this design pads defined size of the chip and were packed closely together, there were no empty spaces and thus no peripheral filler cells were needed. Figure 5.11 illustrates the concept of filler cells. Part a. is a design before filler cells insertion, whereas part b. shows a complete placement with filler cells inserted between the core cells. The spaces come from the fact that the core had to be enlarged to ensure correct placing with added power and ground stripes. Otherwise Encounter placed some cells on top of other cells. Figure 5.12 presents placed design with pads, power and ground connections and added core filler cells.

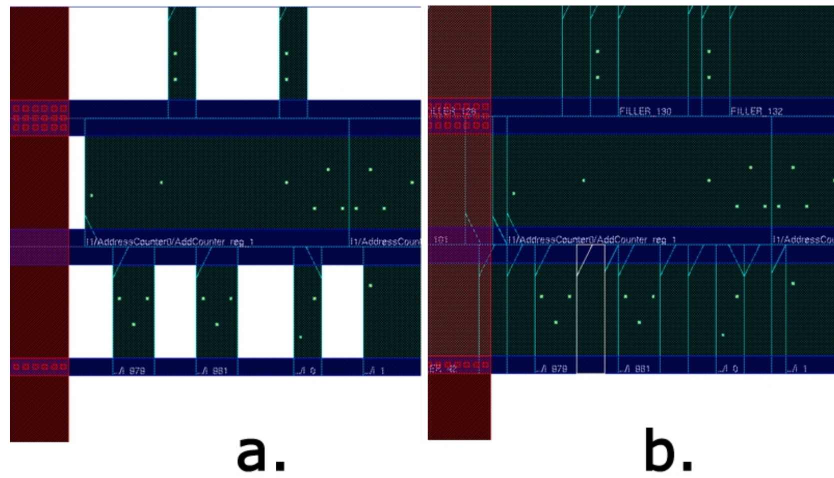

Figure 5.11. Placed design before (a.) and after (b.) inserting filler cells between existing core cells. Also seen are power and ground rails (blue horizontal bars) and power and ground stripes (red vertical bars) together with vias

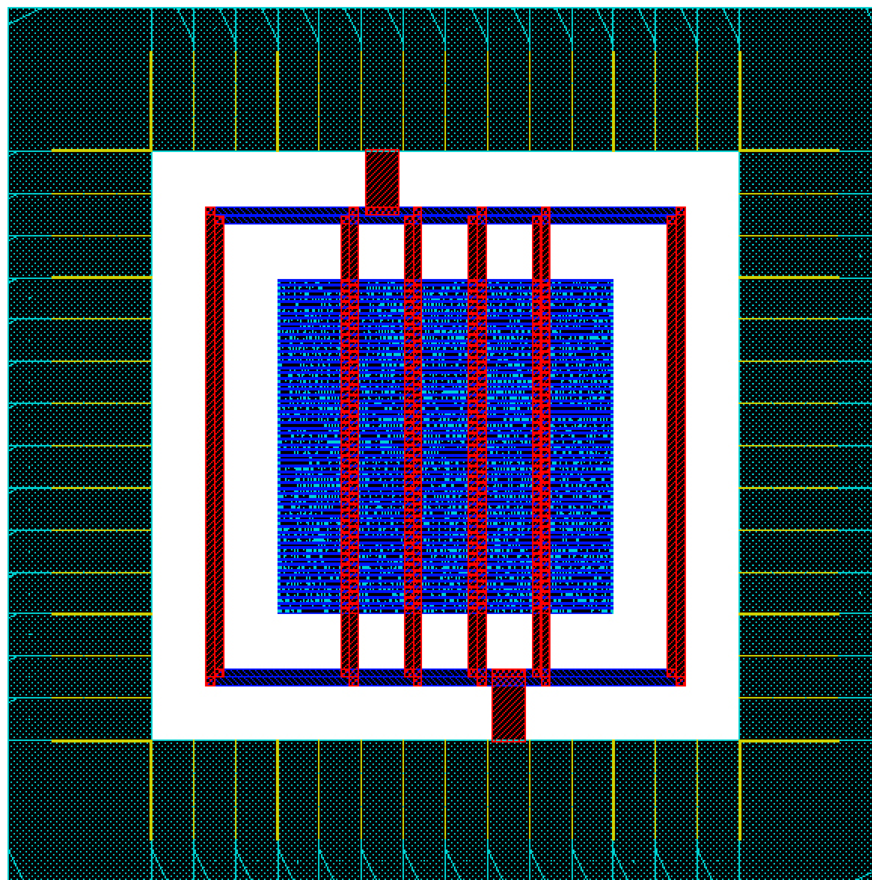

Figure 5.12. Placed design in Encounter

When the design was placed it was ready for routing. However, first the clock tree was generated and analyzed. Figure 5.13 presents clock tree after synthesis. Path of the *clk50_in* signal is indicated with yellow. Coloured instances show delays. Blue ones have minimum whereas the red ones maximum delays of the clock signal. *clk50_in* pad is indicated with blue colour. After viewing the timing report it turned out that the delay differences throughout the core were negligible and should not influence correct functioning of the sequential elements. Table 5.15. summarises delay values for extreme cases.

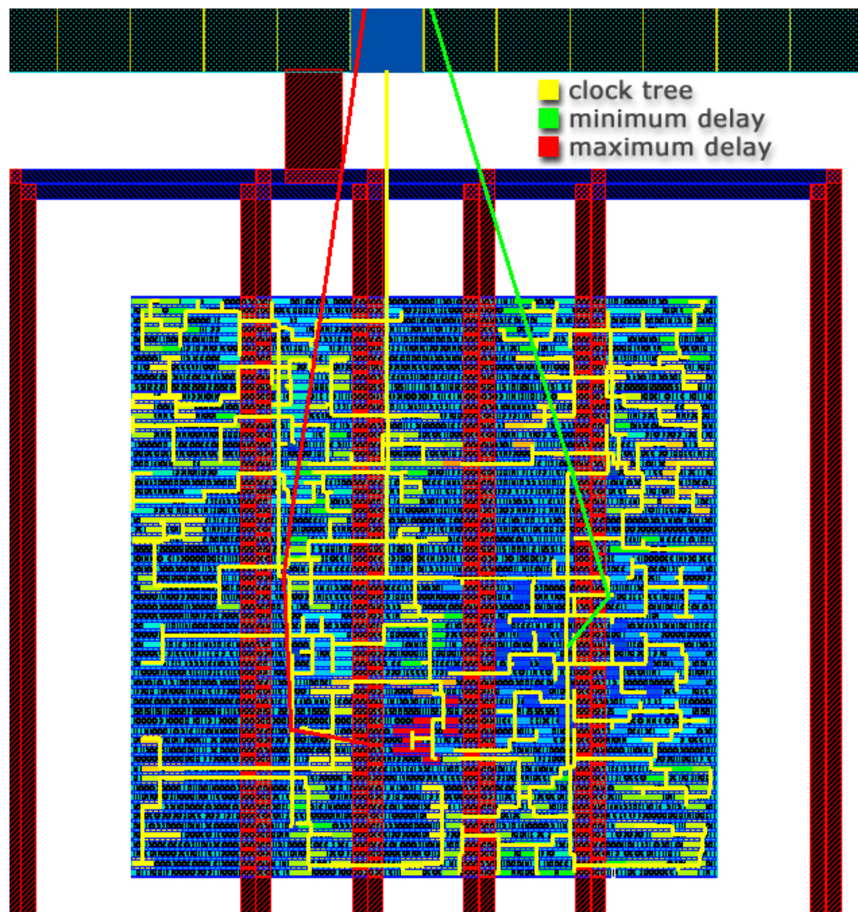

Figure 5.13. Clock tree with colours indicating variation in clock signal delay. Clk50_in pad is indicated with blue colour.

Table 5.15. Delay values after synthesizing clock tree

	Minimum delay	Maximum delay
Instance name	Cnt16_reg_18	TData_reg_7
Delay [ps]	1096.4	1130.1
Delay difference [ps]	30.5	

After analyzing the synthesized clock tree the design was routed using WRoute. This function performs global and detailed routing of the placed instances. Advanced options of 3 Search and Repair passes were chosen to guarantee correct routing of all the nets. The screenshot of routed design is shown in the Figure 5.14.

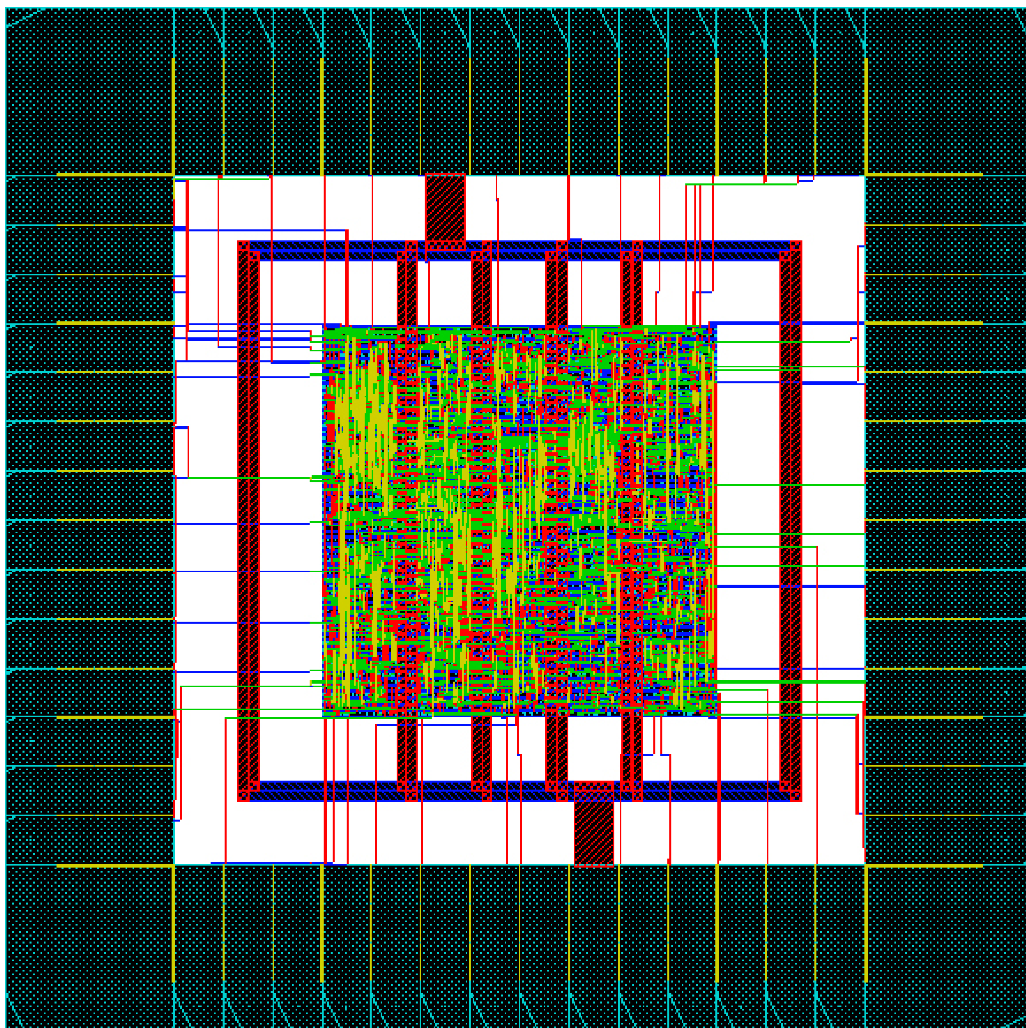

Figure 5.14. Design after routing using WRoute seen in Encounter

Having routed the design, the MetalFill function was applied. It is used to fill the open areas on all metallization layers with an inactive metal. This provides smoother surface of the chip, since variations in metal thickness are minimised. Even distribution of the dielectric improves also the chip performance .

Using Ruler function in Encounter the design was measured to compare the obtained layout with area report generated in BuildGates synthesis tool. The results are summarised in the Table 5.16.

Table 5.16. Chip dimensions comparison between BuildGates and Encounter environments

	Width [μm]	Height [μm]	Area [μm^2]
Core			
BuildGates	544,26	544,26	296223
Encounter	830	830	688900
Chip with pads			
Encounter	2100	2100	4410000

Dimensions for the whole chip with pads are given only for Encounter, since peripheral cells were added after synthesis. It turns out that the area after synthesis equals to only 43 % of area in layout for just the core and including pads this ratio drops to almost 7 %. It can be explained considering that in Encounter pads set the constraints for the floorplan. Pads dimensions are values specified in technology files, thus, if the design is quite small, pads define directly the chip size. Another aspect was the core utilization ratio value, which was set to 0.85. This is an empirical value and has been found to produce good results in other projects.

After generating layout it became apparent that the project using described pads gives possibilities of extending the design with no effect on the chip size. Thus, some additional modules could have been added and the size of the chip would not change provided that

the number of pads stayed the same. A comparison of chip sizes for the described radiation-tolerant design with the same design but with no Hamming codes and state values defined as an enumerated type variable is given in the Table 5.17.

Table 5.17. Comparison of core dimensions for original design and its radiation-tolerant version

	Width [μm]	Height [μm]	Area [μm^2]
Core			
Design with no protection	740	730	540200
Radiation-tolerant design	840	730	613200
Ratio	0.88	1.00	0.88

It can be seen that the dimensions do not differ significantly. It is only the result of two additional pads used in the protected design version, *SingleError* and *DoubleError*. The design is pads limited. Thus, adding Hamming codes to the project does not affect the silicon area used if the number of ports is maintained in the design.

Encounter was used also to create another SDF file, which was used later for the post-layout simulations.

5.5.5 Post-Layout SDF Simulation

Having structural Verilog file generated in BuildGates and SDF files from both BuildGates and Encounter, the post-layout simulations were done. Since the delays observed on the plots were the same, either of SDF files is assumed to have produced the correct results. The post-layout simulations were performed using Aldec Active HDL environment again. Figure 5.15 shows a plot for the top module, which at this stage of the design was the core with pads. The stimuli were as previously, that is *clk50_in* was set to 10 MHz which corresponds to the period of 100 ns. *PowerOnReset* signal was set again to 200 ns and *DataA* port was forced to input the bit vector “0000000100000000” to simulate counting SEUs from memory.

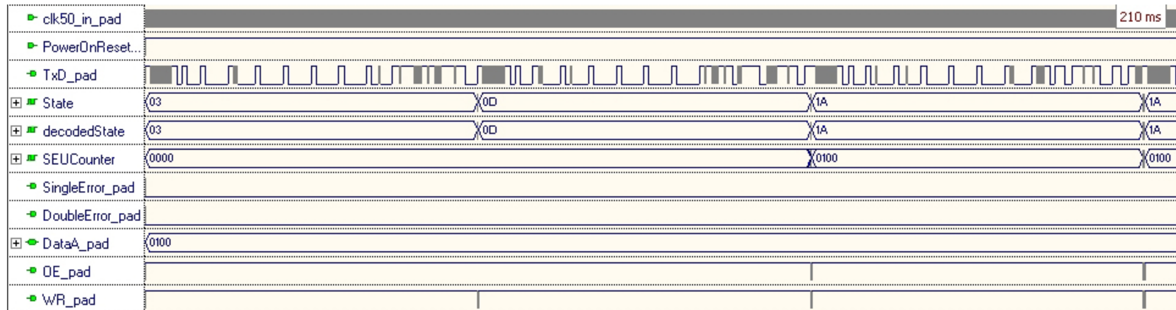

Figure 5.15. Post-layout simulation of radiation-tolerant readout system

The delays between writing *State* signal and reading the *decodedState* signal are equal to 3 ns. Thus, considering the clock period of 100 ns they should not affect correct functioning of the state machine. Each instance included in the Hamming coding process introduces a delay of 1 ns. This is satisfactory for the purposes of this project. It can be observed that the device works exactly the same as originally. Thus, the layout is assumed to have been performed correctly.

5.5.6 Cadence Virtuoso Layout Editor

GDSII (Graphic Data System) stream file created using Encounter software served as a basis for layout generation in Cadence Virtuoso Layout Editor. This can be used for manufacturing the design. Layout view is presented in the Figure 5.16, where all layers can be seen.

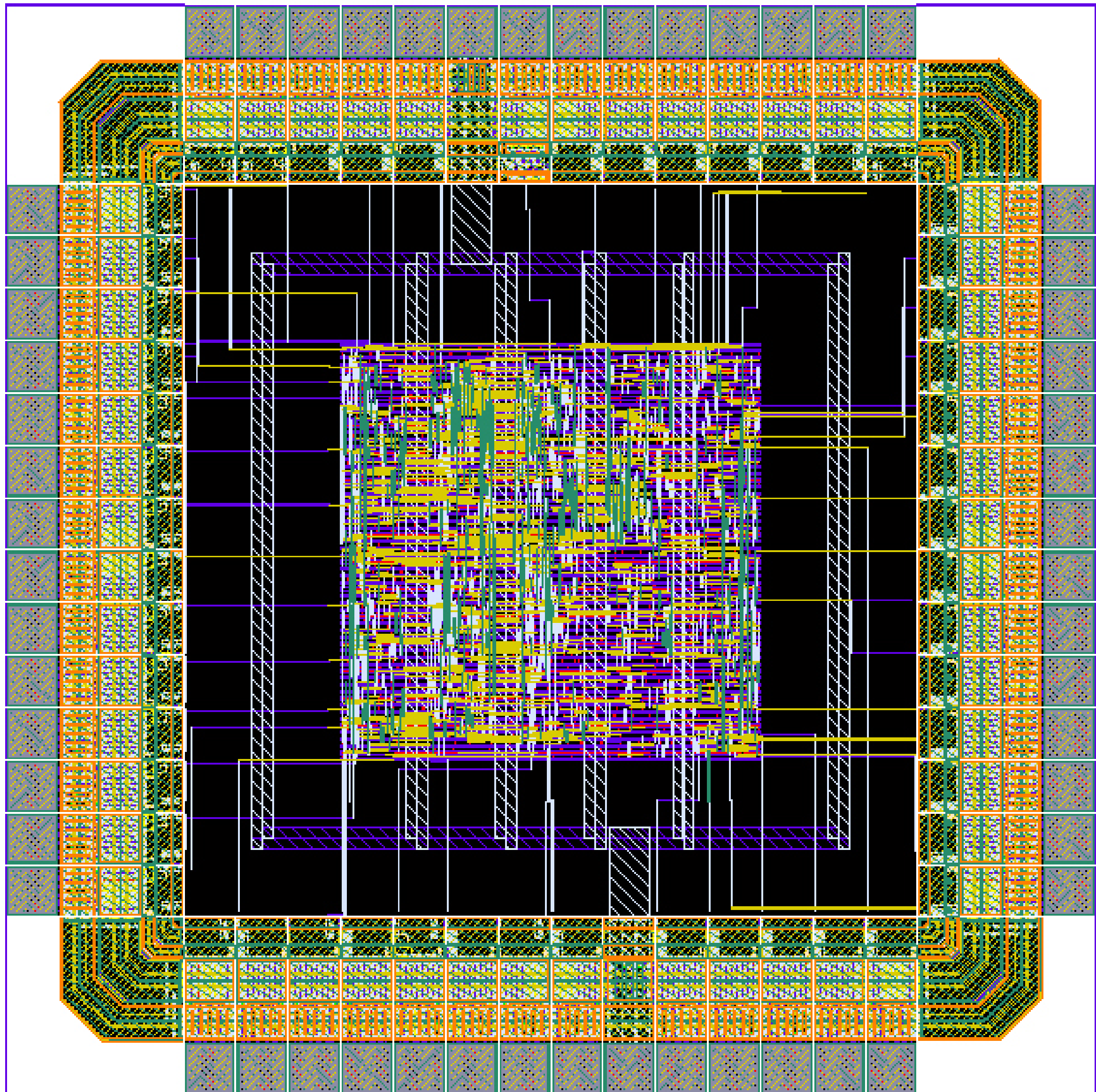

Figure 5.16. Layout view in Cadence Virtuoso Layout Editor after GDS file import

Also *VDD* and *VSS* pad connections can be seen, connecting power and ground rings. After layout generation the Design Rule Check (DRC) was performed. Then, to verify that the design complies to the structural Verilog file, the netlist was imported. This created a schematic view of the device. Verification was done using Layout Versus Schematic (LVS) tool. LVS analysis has shown that the generated layout conforms to the imported Verilog schematic.

6 Conclusions and Suggested Improvements

This thesis has shown how to create a simple radiation tolerant design using an exemplary readout system for a neutron detector. The design was based on a Finite State Machine. Radiation tolerance has been achieved by applying Gray coding to definitions of state values and by using Hamming codes to code these values. Hamming encoding served as a means of detecting single and double errors in the state register. Also it enabled correction of single errors, which became transparent for the device. Since such type of errors is the most common effect of Single Event Upsets on electronic systems, the state register can be considered SEU tolerant. Furthermore, CRC32 encoding was used for the message transmitted from the readout system to provide Error Detection capability when performing the transmission process.

This thesis also shows a full design path from behavioural description in VHDL to a technological layout. All steps have been described, including encoding, synthesis, Place & Route, layout generation and post-layout simulations. The system has been checked to have the same functionality when simulated using behavioural description as well as using netlists and constraints files created during layout generation.

The generated layout can be further used for a complete neutron detector system integrated in one chip. There are two ways for achieving this goal. First is manufacturing the readout system as presented and the detector separately. Then the two parts would have to be connected. An alternative is placing the SRAM based detector inside the readout chip as a separate block. This would be the least area consuming solution. All connections between two devices would then be wires inside the device core. Such solution would however include some changes in the behavioural description and performing the whole design path once again, including the detector in the chip.

Work on layout generation has shown a strong dependence of chip size on used pads. It has been observed that dimensions of the peripheral cells defined in technological datasheets influence the total dimensions of the device and provided that large number of ports is used, changes in the core are not visible in the chip size. Thus, apart from making the core as small as possible it is very important to carefully choose the used Input/Output features. A solution to minimise the size of the system could be a system using a FIFO (First In First Out) register as the radiation detector. In this way no *Address* bus is necessary and a single serial pin is used in the place of the *Data* bus. Thus, 19 address pins and 16 data pins are exchanged for just one pin. Rough calculations give the core size of approximately 500 x 500 μm, taking into account the pad dimensions in the described technology. This greatly increases the integration of the system but using a different design of the sensor. The appropriate solution should be chosen based on precise calculations of silicon area, cost and performance of the whole design.

This thesis presents radiation tolerance achieved only by applying some coding schemes to the design. However, other described solutions may be used to improve design's susceptibility to SEUs. Next version of the readout system could use Triple Modular Redundancy. This includes designing some modules again and including TMR schemes.

References

[Allenspach 95] ''Single-Event Gate-Rupture in Power MOSFETs: Prediction of Breakdown Biases and Evaluation of Oxide Thickness Dependence'', M. Allenspach et al., IEEE, 1995

[AMS] AMS Hit-Kit v3.70 datasheet

[Andraka] ''A Low Complexity Method for Detecting Configuration Upset in SRAM Based FPGA'', R.J. Andraka, J.L. Brady

[Anelli 00] ''Conception Et Caracterisation De Circuits Integers Resistants Aux Radiations Pour Les Detecteurs De Particules Du LHC En Technologies CMOS Submicroniques Profondes'', Giovanni Maria Anelli, 2000

[Anghinolfi 00] ''Radiation Hard Electronics'', F. Anghinolfi, CERN/EP, 2000

[Baloch 06] ''Design of a Single Event Upset (SEU) Mitigation Technique for Programmable Devices'', S. Baloch et al., IEEE, 2006

[Bezerra] ''Improving Reconfigurable Systems Reliability by Combining Periodical Test and Redundancy Techniques: A Case Study'', E. A. Bezerra et al.,IEEE

[Blomgren] ''Nuclear Data for single-event effects'', J. Blomgren, Uppsala University

[Carson 97] ''Radiation Hardening of Electronics'', M. Carson et al., 1997, http://www.mse.vt.edu/faculty/hendricks/mse4206/ /projects97/group02/radhard.htm

[CRC 96] ''A painless guide to CRC error detection algorithms'', 1996, http://www.repairfaq.org/filipg/LINK/F_crc_v3.html

[Encounter 04] Encounter User Guide, Product Version 3.3.3, Cadence, June 2004

[Foutz 92] ''Power Transistor Single Event Burnout'', Jerrold Foutz, 1992, http://www.smpstech.com/power-mosfet-single-event-burnout.htm

[Hentschke 02] ''Analyzing Area and Performance Penalty of Protecting Different Digital Modules with Hamming Code and Triple Modular Redundancy'', R. Hentschke et al., IEEE, 2002

[Holmes-Siedle 02] ''Handbook of Radiation Effects'', Second Edition, Andrew Holmes-Siedle and Len Adams, 2002

[Katz 97] ''Radiation effects on current Field Programmable Technologies'', R. Katz et al., IEEE, 1997

[Koga] ''Single Event Functional Interrupt (SEFI) Sensitivity in EEPROMs'', R. Koga

[Lenahan 99] ''Predicting radiation response from process parameters: Verificationof a physically based predictive model'', P.M.

Lenahan, 1999
[Li 00] ''References for Neutron Irradiation Tolerance Testing for CSC electronics'', Yong Li, 2000, http://positron.ps.uci.edu/~yong/neutron_rad.html
[Lima 00] ''Designing and Testing a Radiation Hardened 8051-like Micro-controller'', F. G. Lima et al., 2000
[Lima 03] ''Designing Fault Tolerant Systems into SRAM-based FPGAs'', F. G. Lima et al., DAC'03, June 2-6, 2003
[Lyons 62] ''The Use of Triple-Modular Redundancy to Improve Computer Reliability'', R. E. Lyons and W. Vanderkulk, IBM Journal, April 1962
[Makowski 04] ''Application of a genetic algorithm to design of radiation tolerant programmable devices'', D. Makowski, M. Grecki, G. Jablonski, 11th International MIXDES Conference, 2004
[Makowski 06] ''A Distributed System for Radiation Monitoring at Linear Accelerators'', D. Makowski, M. Grecki, A. Napieralski, S. Simrock, B. Mukherjee, IEEE Transactions on Nuclear Science, Vol 53, Issue 4, Part 1, Aug. 2006
[Massengill 02] ''Single Event Circuit Effects'', Lloyd W. Massengill, Vanderbilt University, 2002
[Mavis 02] ''Single Single Event Transient Phenomena - Challenges and Solutions'', David G. Mavis, 2002
[Mielczarek 05] Design of Radiation Tolerant Transmission Channel Circuit, Jakub Mielczarek, MSc thesis at the Technical University of Lodz, 05
[Mohanram 00] ''Cost-Effective Approach for Reducing Soft Error Failure Rate in Logic Circuits'', K. Mohanram and N.A. Touba, 2003
[Neamen 02] ''Semiconductor Physics And Devices'', Donald Neamen, 2002
[Normand 97] ''Neutron-Induced Single Event Burnout in High Voltage Electronics'', Eugene Normand et al., 1997
[Olcayto 97] ''Information Theory lecture notes'', E. Olcayto, 1997
[Phelan 04] ''Solutions for Soft Errors in System on Chip Designs'', Richard Phelan, 2004
[Quicklogic 03] ''Single Event Upsets in FPGAs'', White Paper, Quicklogic Corporation, 2003
[Ritter 86] ''The Great CRC Mystery'', Dr. Dobb's Journal of Software Tools. February. 11(2): 26-34, 76-83, Terry Ritter, 1986
[Shaneyfelt 98] ''Challenges in Hardening Technologies Using Shallow-Trench Isolation'', M.R. Shaneyfelt et al., Sandia National Laboratories, IEEE Transactions on Nuclear Science, Vol 45, No 6, December 1998
[Snoeys 02] ''A New NMOS Layout Structure for Radiation Tolerance'', Snoeys, Gutierrez and Anelli, IEEE, 2002
[Srinivasan 04] ''Improving Soft-error Tolerance of FPGA Configuration Bits'', Suresh Srinivasan et al., IEEE, 2004

[Synplicity 99] ''Designing Safe VHDL State Machines'', Application Note, Synplicity, 1999

[Sze 01] ''Semiconductor Devices: Physics and Technology'', Simon M. Sze, 2001

[Tanenbaum 03] ''Computer Networks'', Fourth Edition, Andrew S. Tanenbaum, 2003

[Wikipedia] Wikipedia, the free encyclopedia, wikipedia.org

[Wirthlin] ''Hardness By Design Techniques for Field Programmable Gate Arrays'', Michael Wirthlin et al., IEEE

[Yoshioka 94] ''A Radiation-Hardened 32-bit Microprocessor Based on the Commercial CMOS Process'', Yoshioka et al., IEEE Transactions on Nuclear Science, Vol 41, No 6, December 1994

[Ziegler 96] ''IBM experiments in soft fails in computer electronics (1978-1994)'', J.F. Ziegler e alli, 1996

Appendix A: **Tutorial Presenting Layout Generation from VHDL**

This appendix provides a tutorial that shows how to generate a complete layout having a behavioural description of a design written either in VHDL or Verilog. It is targeted for Cadence Hit Kit v3.70 for 0,35 µm CMOS silicon technology.

Cadence BuildGates Physically Knowledgeable Synthesis

Having VHDL or Verilog files ready for synthesis open Cadence Physically Knowledgeable Synthesis Build Gates tool by running the following script from the console:

```
cds_paths.sh
```

It may be necessary to set the PATH environmental variable by entering:

```
setenv PATH ${PATH}:/cad/cadence/SOC33USR3/BuildGates/version/bin
```

Having BuildGates running click Open a File and check

Tcl source script

and choose

```
/cad/deskits/AMS/ams_v3.70/buildgates/c35_3.3V/read_libs.tcl
```

Figure A.1 presents the Open a File window for loading Tcl source scripts.

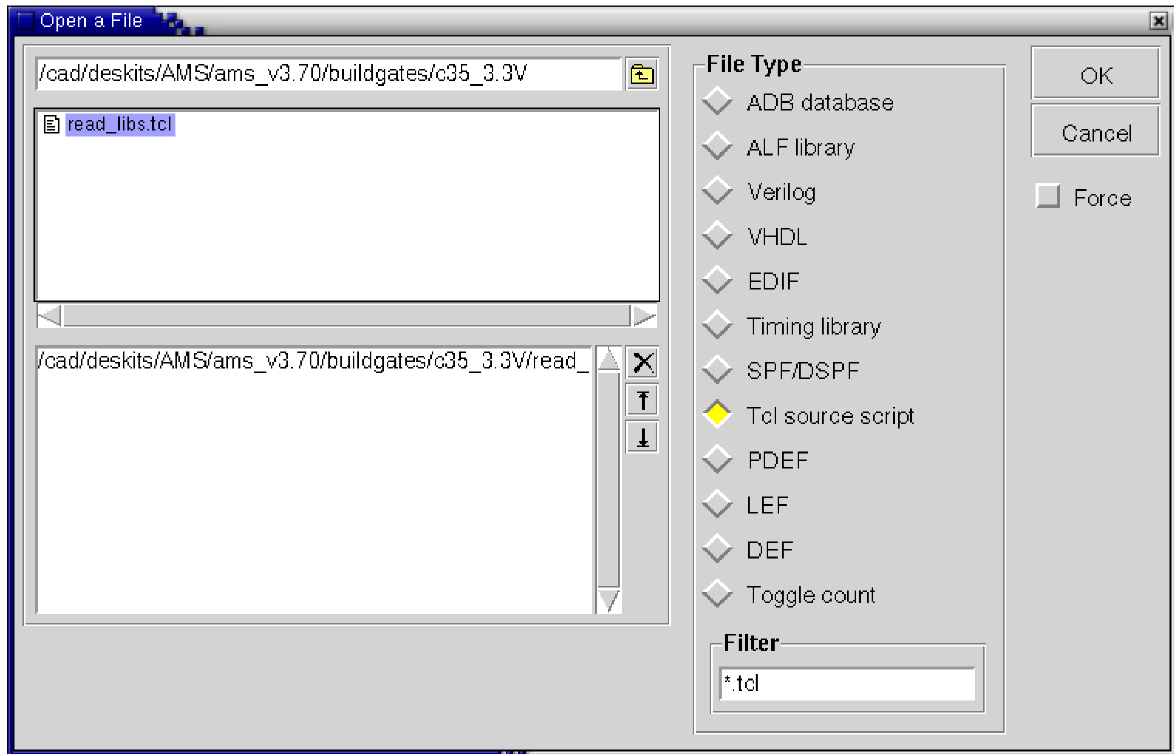

Figure A.1. Loading Tcl source script into Cadence BuildGates.

Wait until technological libraries are loaded and then choose Open a File again and check either VHDL or Verilog, depending on the type of HDL files prepared. Then go to the catalogue where those files are stored and add them to the project just by clicking on their names. It is important that all the packages and library files needed in the design modules are loaded first. Also lower modules in the hierarchy should be chosen before clicking on the top module, which in this case is sender. This ensures that all files are loaded correctly. Loading source files into BuildGates is shown in the Figure A.2.

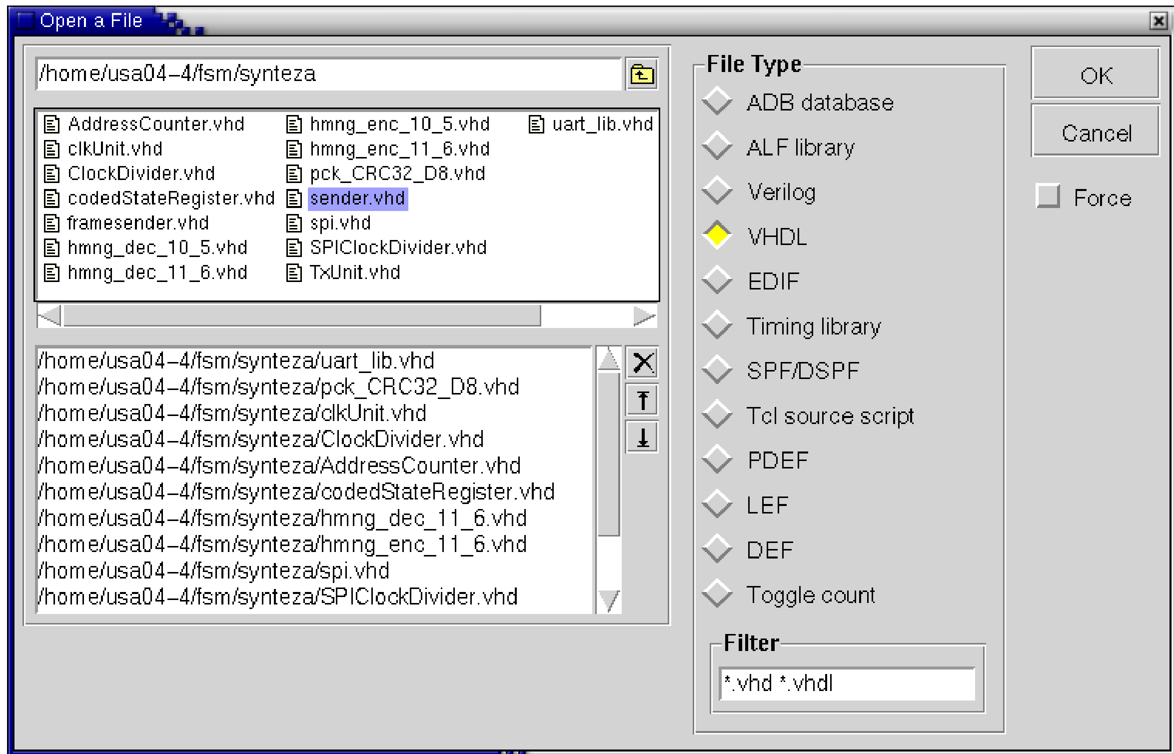

Figure A.2. Loading VHDL source files into Cadence BuildGates.

After all source files have been loaded correctly choose Commands -> BuildGeneric, check All, as shown in the Figure A.3. and click OK.

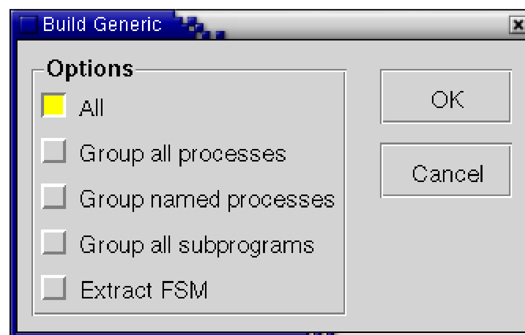

Figure A.3. Commands -> BuildGeneric menu

After BuildGeneric command the structure of the design is generated and the RTL schematic of the project can be viewed. It should be noted that the module sender has been detected as top module. So far the technological rules are not applied. This is done after

clicking Commands -> Optimize. In the window that appears choose Effort Level: Medium and Priority: Area, as presented in the Figure A.4.

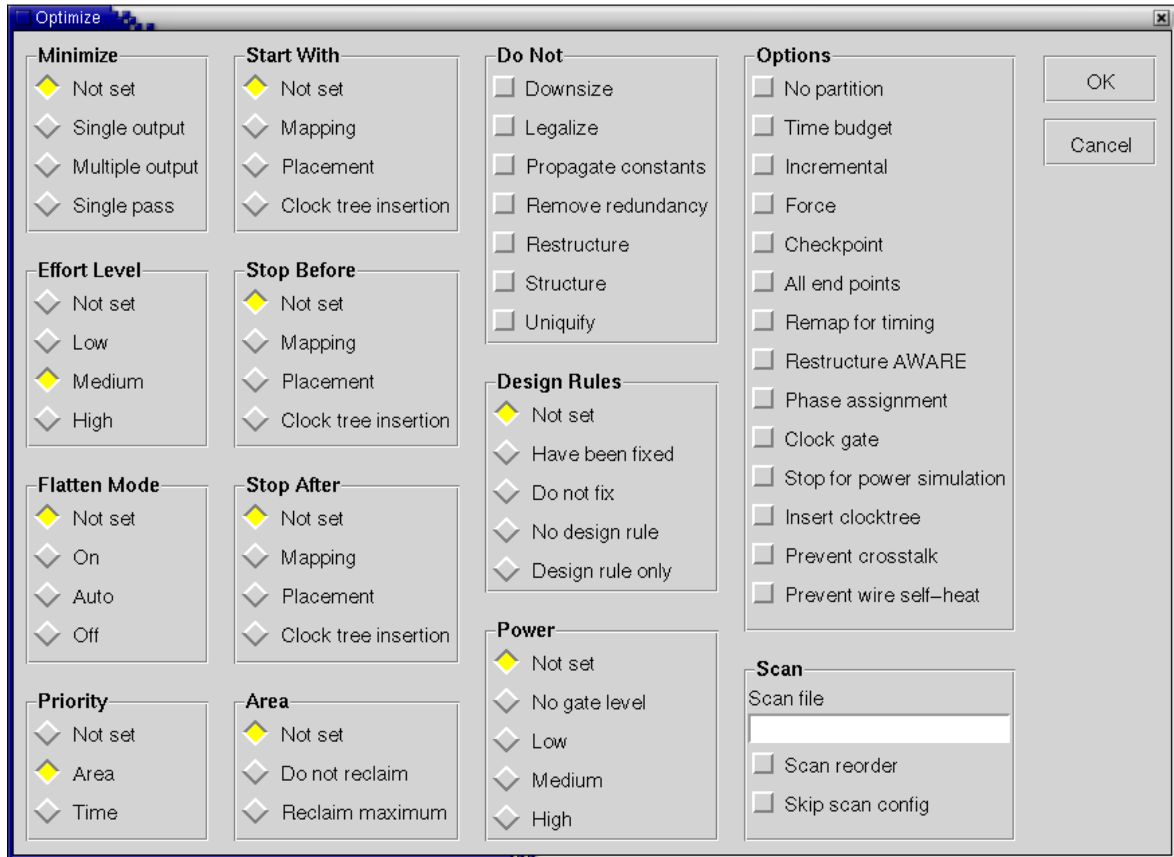

Figure A.4. Commands -> Optimize menu

Optimization reduces the project structure to design modules. Also the technological rules for version 3.70 of AMS c35b4 technology are included at this stage. After optimization the area report can be generated, which shows how much space the project uses on the chip. Areas for individual modules as well as for the whole design are shown. The report is generated by clicking Reports -> Area... and then on Generate Report icon. This is illustrated in the Figure A.5. The report can be saved to a text file by clicking on Write Report to File icon.

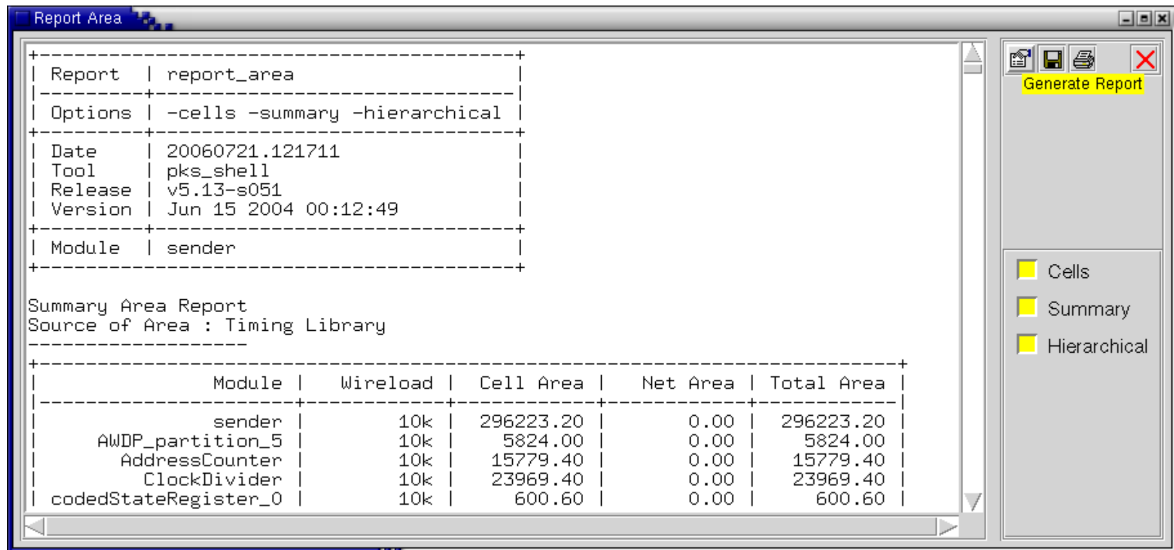

Figure A.5. Generate Report -> Area menu

The purpose of running BuildGates synthesis is generation of a structural Verilog netlist file, which will be later used for layout generation and post-layout analysis. First module sender should be set as top module to ensure the correct hierarchy by clicking right mouse button on the entity in the generated structure and choosing Set Current Module and then Set Top Timing Module. Writing structural Verilog file is done by writing the following command in the BuildGates pks_shell:

```
write_verilog -hier sender.v
```

This generates a sender.v structural Verilog netlist file and -hier option assures that the whole hierarchy of modules is included. Also an SDF file can be saved. However, before writing it, the timing constraints have to be specified. Go to Constraints tab in the main window of BuildGates and click New Ideal Clock icon. Enter clk50_in as Ideal clock name, as this is the name of the main clock signal in the design. As Ideal clock period enter 100 and click OK. This is illustrated in the Figure A.6.

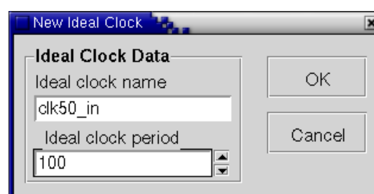

Figure A.6. New Ideal Clock menu

Then by clicking right mouse button in the upper right window choose New Port Clock... as shown in the Figure A.7.

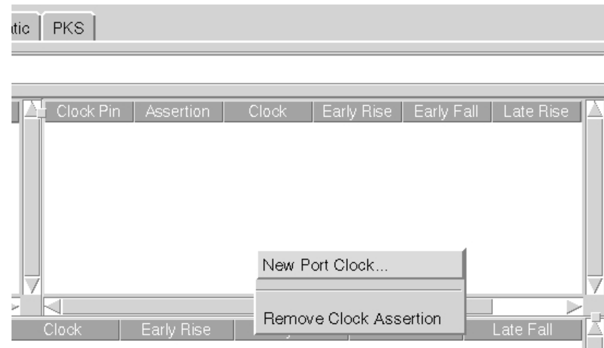

Figure A.7. Choosing New Port Clock menu

In the New Port Clock window choose clk50_in as Ideal clock and also as Port clock. Enter the values for Early rise and Early fall as shown in the Figure A.8.

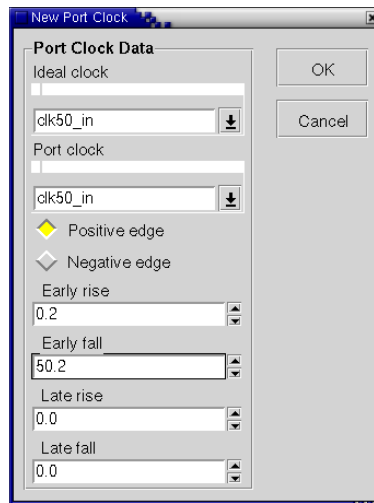

Figure A.8. New Port Clock menu

Now the SDF file can be generated by entering the following command in the pks_shell:

```
write_sdf -delimiter . -precision 4 sender_build_gates.sdf
```

Having structural Verilog file and SDF file first post-synthesis simulations can be

performed, using for example Aldec Active HDL software. The Verilog file will be now used for layout generation in Cadence Encounter environment. However, before running this software, the generated netlist has to be changed to include pads connections. To do this, open sender.v for editing and add an additional pads module, which from now on will be the top module of the design. This module just maps all ports of the sender module to external pads of the chip, connecting them via instances being the actual pads names taken from the technology datasheet. The used names for input, inout and output ports are gathered in the Table A.1 together with instances for clock, ground, power and corner pad connections.

Table A.1. Pads cell names used in the sender design

Type	Name
Input	ICP
Inout	BBC8P
Output	BU16P
Clock	ICCK8P
Ground	GND3ALLP
Power	VDD3ALLP
Corner	CORNERP

The complete Verilog netlist is attached as sender.v file. Apart from netlist, the io file has to be created. This serves as a guideline for pads insertion in Encounter. Not only instance names are included here, but also orientation of all the pads. Additionally, corner cells are defined in this file. Sender.io is included as an attachment. Assignment of all pads is presented in the Table A.2 together with their orientations.

Table A.2. Pads' names and orientation together with technological name

Pad name	Orientation	Pad cell	Pad name	Orientation	Pad cell
CORNER4	SW	CORNERP	CORNER3	NW	CORNERP
CORNER1	NE	CORNERP	CORNER2	SE	CORNERP
DataA14	N	BBC8P	Address18	E	BU16P
DataA13	N	BBC8P	Address17	E	BU16P
SingleError	N	BU16P	Address16	E	BU16P
DoubleError	N	BU16P	Address15	E	BU16P
TxD	N	BU16P	Address14	E	BU16P
VDD	N	VDD3ALLP	Address13	E	BU16P
clk50_in	N	ICCK8P	Address12	E	BU16P
LED4	N	BU16P	Address11	E	BU16P
LED3	N	BU16P	Address10	E	BU16P
LED2	N	BU16P	Address9	E	BU16P
LED1	N	BU16P	Address8	E	BU16P
LED0	N	BU16P	Address7	E	BU16P
OE	N	BU16P	Address6	E	BU16P
WR	N	BU16P	Address5	E	BU16P
DataA13	W	BBC8P	PowerOnReset	S	ICP
DataA12	W	BBC8P	SPI_Clk	S	BU16P
DataA11	W	BBC8P	SPI_In	S	ICP
DataA10	W	BBC8P	SPI_CS	S	BU16P
DataA9	W	BBC8P	RadFETSwitch	S	BBC8P
DataA8	W	BBC8P	TempSPI_Clk	S	BU16P
DataA7	W	BBC8P	TempSPI_In	S	ICP
DataA6	W	BBC8P	TempSPI_CS	S	BU16P
DataA5	W	BBC8P	VSS	S	GND3ALLP
DataA4	W	BBC8P	Address0	S	BU16P
DataA3	W	BBC8P	Address1	S	BU16P
DataA2	W	BBC8P	Address2	S	BU16P
DataA1	W	BBC8P	Address3	S	BU16P
DataA0	W	BBC8P	Address4	S	BU16P
Totally: 56 pads, 14 pads per each side of the chip					

Cadence FirstEncounter

Having these files prepared, the Encounter environment can be run by entering the following command in the terminal:

```
/cad/cadence/SOC33USR3/tools/bin/encounter
```

When Encounter starts, the GUI should look like in the Figure A.9.

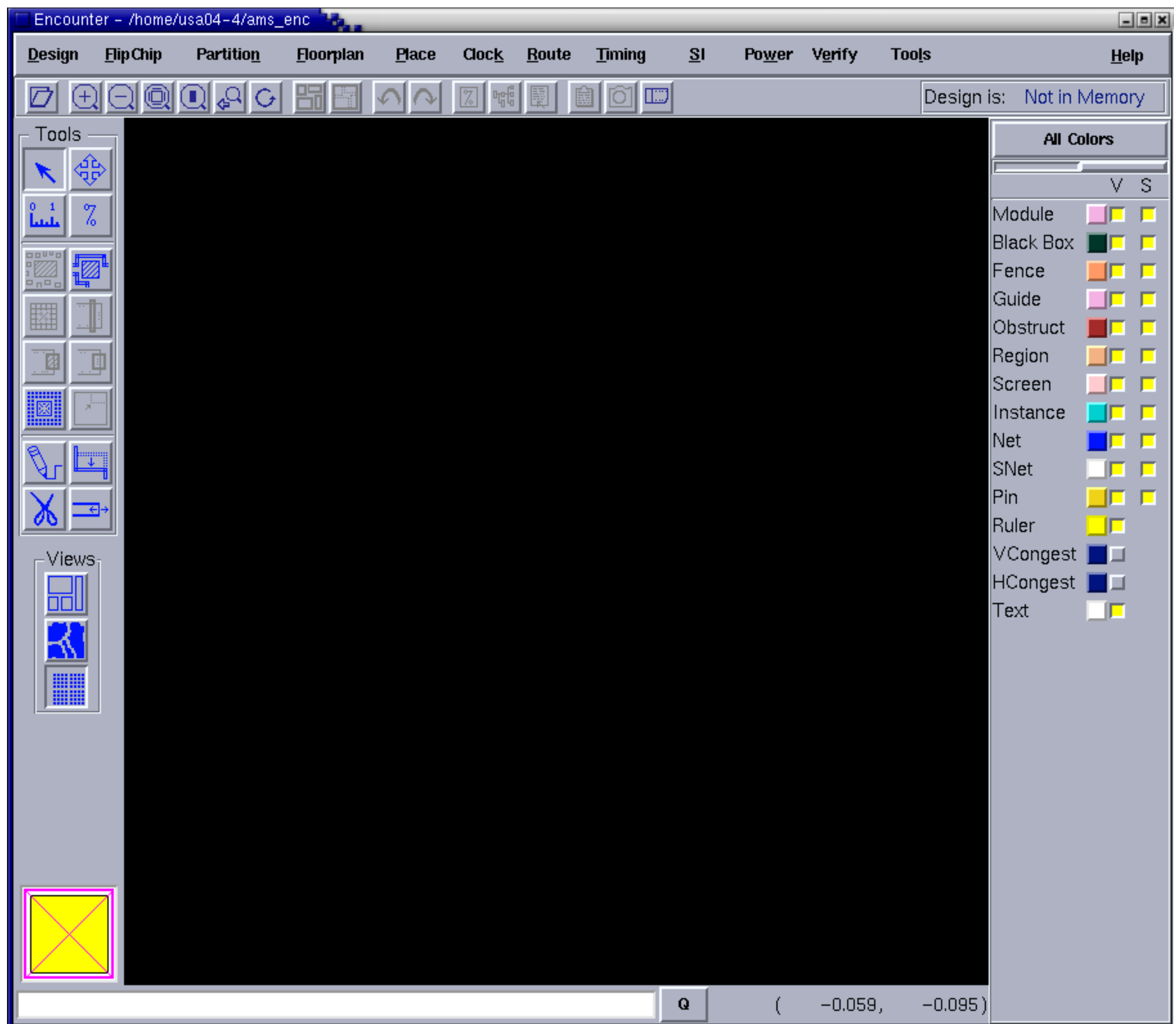

Figure A.9. Main window of Cadence Encounter environment

To load the design click on the Design Import icon or choose Design Import from the Design menu. In the Netlist section choose Verilog Files: sender.v, choose Top Cell: By User: and write pads. In the Technology Information/Physical Libraries section choose the following LEF Files:

```
/cad/deskits/AMS/ams_v3.70/artist/HK_C35/LEF/c35b4/c35b4.lef
/cad/deskits/AMS/ams_v3.70/artist/HK_C35/LEF/c35b4/CORELIB.lef
/cad/deskits/AMS/ams_v3.70/artist/HK_C35/LEF/c35b4/IOLIB_4M.lef
```

In Timing Libraries section enter the following values:

Common Timing Libraries:

```
/cad/deskits/AMS/ams_v3.70/liberty/c35_3.3V/c35_CORELIB.lib
/cad/deskits/AMS/ams_v3.70/liberty/c35_3.3V/c35_IOLIB_4M.lib
```

Buffer Name/Footprint:

`BUF2 BUF4 BUF6 BUF8 BUF12 BUF15`

Delay Name/Footprint:

`DLY12 DLY22 DLY32 DLY42`

Inverter Name/Footprint:

`INV0 INV1 INV2 INV3 INV4 INV8 INV10 INV12 INV15`

Check the option 'Generate Footprint Based on Functional Equivalence'.

In 'IO Information' section choose:

IO Assignment File: sender.io.

All this entries can be saved to a 'sender.conf' file. Such file is also attached to the project. Apart from the entered information there are also some other parametres of the design, such as core dimensions. In the case of this project the following values have been set:

`ui_aspect_ratio: 1.0` – aspect ratio for the chip dimensions, 1.0 means that the square shape of the core is prioritized;

`ui_core_util: 0.85` – core utilization parameter, set to a common value of 0.85;

`ui_core_to_left/right/top/bottom: 300.0` – distance from core to pads expressed in micrometers, assigns space later used for power rings generation;

This data can be seen in the 'Core Spec Defaults' tab of the Design Import menu.

The next step is setting the Timing parametres in the Timing tab of the Design Import menu. All values should be left as defaults.

After completing the Timing section, click on the Power tab in the Design Import menu. The values for Power/Ground Nets are following:

```
Power Nets: vdd! vdd3r1! vdd3r2! vdd3o! VDD
Ground Nets: gnd! gnd3r! gnd3o! VSS
```

Power Analysis Scaling should be left at default. After completing these tabs, leave the Misc. tab unchanged and click OK. Now the design is being loaded into memory. During loading the following errors will be displayed in the terminal:

```
**ERROR: Macro * obs coordinate x value * isn't on
manufacturing grid. It's likely result n placement/routing
that can't be manufactured.
```

This errors can be omitted since the coordinates belong to obstructions in LEF files that describe 45 degree shapes, which is of no importance for Place&Route. These cells are exchanged by the complete layouts. [AMS]

What user sees on the screen after the design has been imported is shown in the Figure A.10.

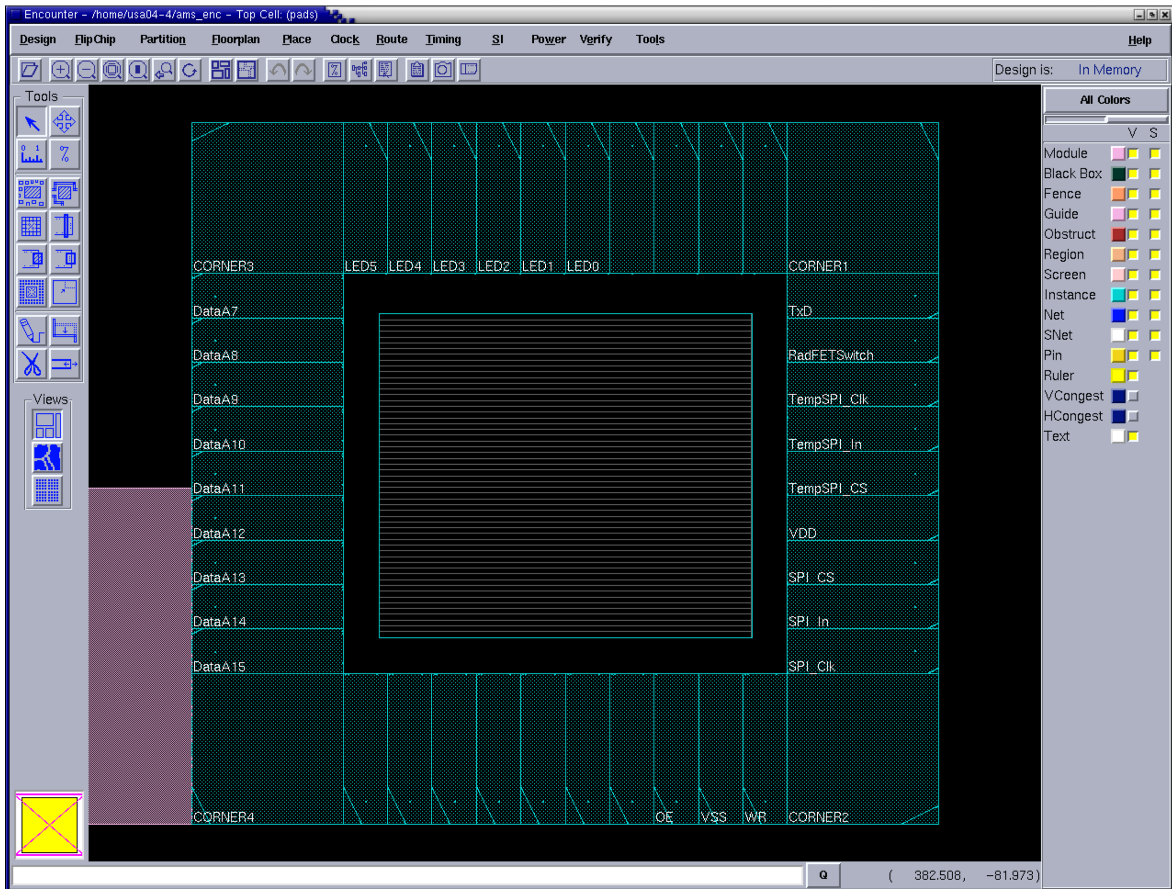

Figure A.10. Encounter GUI after importing the design

After successfully importing design the core block can be placed. Go to Floorplan -> Place Blocks/Modules -> Place and click OK. Now clicking on Floorplan -> Specify Floorplan the detailed settings for floorplan can be viewed and changed if necessary. If all parametres are satisfactory, go to Floorplan -> Global Net Connections and enter all necessary Power Ground Connections as follows in the Table A.3.

Figure A.3. Global Net Connections for Power Planning

Connect -> Pins:	In Instances:	Scope:	To Global Net:
vdd!	*	Apply All	VDD
vdd3r1!	*	Apply All	VDD
vdd3r2!	*	Apply All	VDD
vdd3o!	*	Apply All	VDD
VDD	*	Apply All	VDD

gnd!	*	Apply All	VSS
gnd3r!	*	Apply All	VSS
gnd3o!	*	Apply All	VSS
VSS	*	Apply All	VSS
TIEHI	*	Apply All	VDD
TIELO	*	Apply All	VSS

Global Net Connections settings are also shown in the Figure A.11. After completing all fields click Apply and then Check. After that click Close.

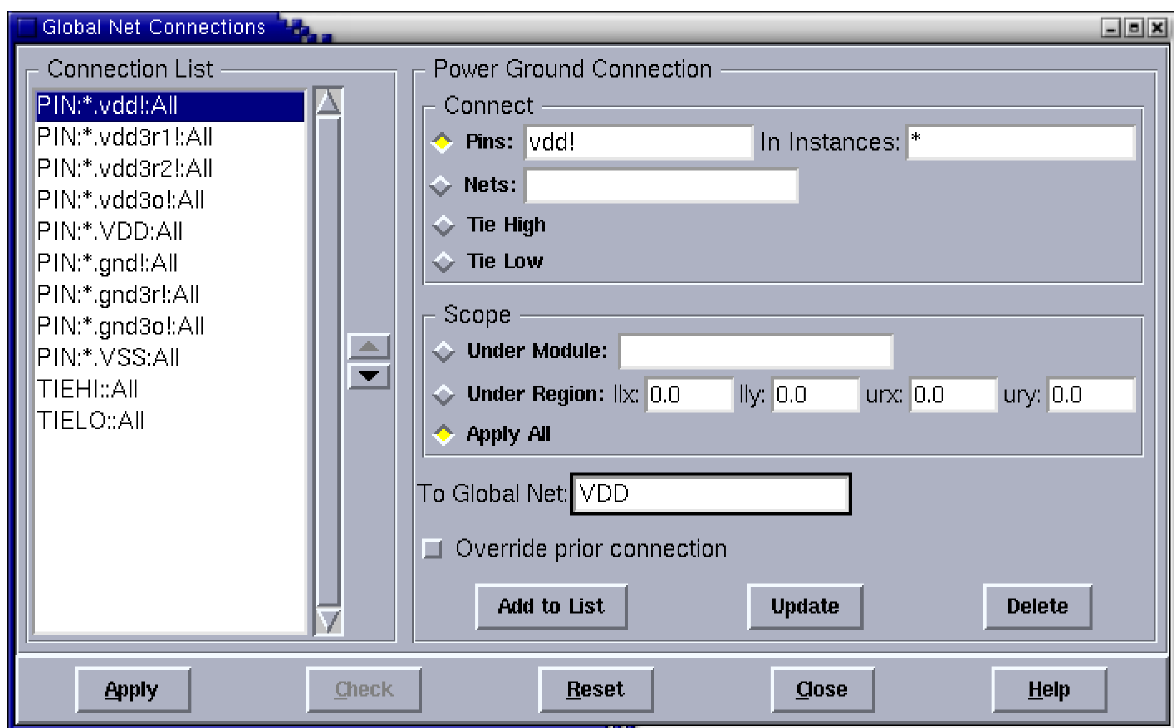

Figure A.11. Global Net Connections window

After specifying connections for ground and power supply the power rings can be added. Go to Floorplan -> Power Planning -> Add Rings... Enter Nets: VSS VDD and for the dimensions:

```
Width: 20 20 20 20
Spacing: 1 1 1 1
```

Check Offset: Center in channel. This generates two rings, the inner one is for VSS and the

outer one for VDD connections. Their dimensions are set according to the technology datasheet. Rings are placed in the middle between core and IO regions. Click OK. Then go to Floorplan -> Power Planning -> Add Stripes... in order to add additional connections accross the core. In the Add Stripes window enter Nets: VSS VDD and the dimensions as previously:

```
Width: 20
Spacing: 1
```

The direction stays at its default: Vertical. Then in the Set Pattern section set:

```
Number of sets: 1
```

and in Stripe Offset Boundary set:

```
Relative from core or area:
     X from left: 150
     X from right: 150
```

This command generates 4 vertical stripes connecting cells in the core to VSS and VDD that are approximately equally spaced accross the core. The next step is using Special Routing to generate horizontal rails used for power and ground connections for each individual cell. These are done interchangeably, so that each cell has access to both rails. However, before running SRoute, the End Cap Cells need to be added. These are cells that are placed at the edges of the core and define the limits for the power and ground rails. Go to Place -> Filler -> Add End Cap... and enter the following:

```
Pre Cap Cell: ENDCAPPL
Post Cap Cell: ENDCAPPR
```

Click OK. Now the SRoute may be run by clicking Route -> Sroute and enter:

```
Nets: VSS VDD
```

Go to Advanced tab, choose Extension Control and check the following options:

```
Primary connection for: None
Secondary connection/stop: Last cell in the row
```

After successful Special Routing the design is ready to be placed. First make sure that the placement blockage is declared for the second layer of metallization by clicking Place -> Specify -> Placement Blockage. Option M2 should be checked to ensure that the core cells

are not placed directly below the stripes. Now go to Place -> Place..., select:

```
Placement Effort Level: Medium Effort
```

and click OK. Now the placed design can be seen in the main window. This is presented in the Figure A.12.

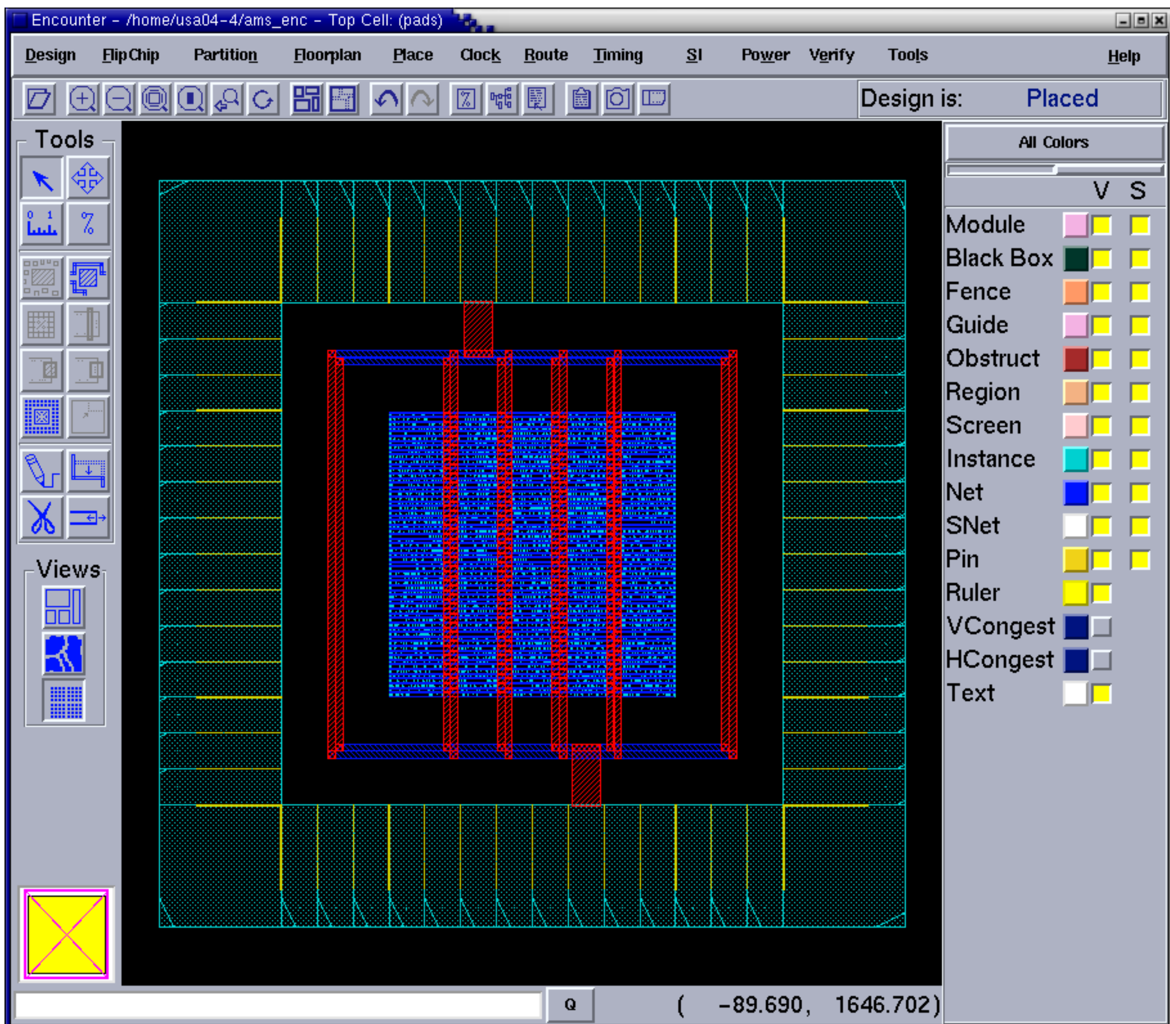

Figure A.12. Chip layout after placing the design

Now, in order to fill the empty spaces between the core cells, the filler cells have to be added. This is done by entering the following command in the Encounter shell:

```
addFiller -cell FILL25 FILL10 FILL5 FILL2 FILL1 -prefix FILLER
```

This adds filler cells from the largest to the smallest ones, depending on the spacing

between the adjacent core cells. The same should be done for the peripheral cells. In the Encounter shell enter the following commands:

```
addIoFiller -cell PERI_SPACER_100_P -prefix pfill
addIoFiller -cell PERI_SPACER_50_P -prefix pfill
addIoFiller -cell PERI_SPACER_20_P -prefix pfill
addIoFiller -cell PERI_SPACER_10_P -prefix pfill
addIoFiller -cell PERI_SPACER_1_P -prefix pfill
addIoFiller -cell PERI_SPACER_01_P -prefix pfill
```

Here it is also important that these commands are entered in the correct order. The fillers are added from the largest to the smallest ones. This ensures that the fillers are placed in the optimal way. The design after placing filler cells is presented in the Figure A.13.

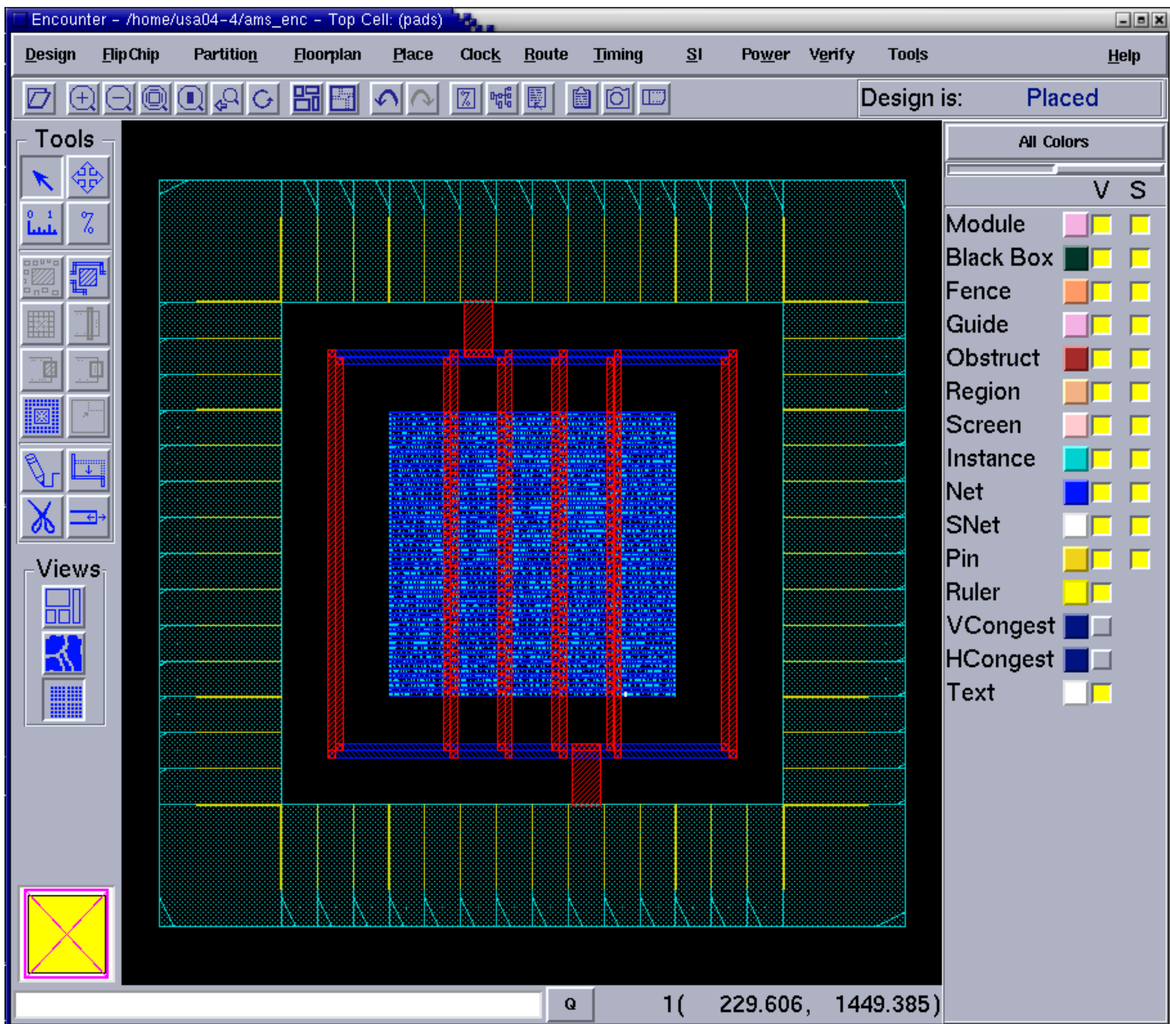

Figure A.13. Design view after placing filler cells

Before routing the design the clock tree can be specified and checked. To do this, go to Clock -> Create Clock Tree Specification... and enter:

```
Buffer Footprint: A+QA
Inverter Footprint: A+Q!(A)
```

These values are taken from the technology specifications. Enter the valid file name in Save Spec To: sender.ctstch. Click OK. Now go to Clock -> Specify Clock Tree... and enter Clock Tree File: sender.ctstch. Now click Clock -> Synthesize Clock Tree..., enter:

Result Directory: sender_cts

Base File Name: sender_cts

Now the clock tree can be displayed and analyzed. Choosing Clock -> Display -> Display Clock Tree... provides options for viewing the clock tree. Click Display Clock Tree and check All Level to display the clock path from the clk50_in pad to all modules driven by this clock signal. Choosing Display Clock Phase Delay views the differences in the clock signal delay by colouring the instances depending on the delay. The blue ones have the smallest delays, whereas the red ones the largest. By choosing Display Min/Max Paths one can see just the connections of the clock signal to the instances with the smallest and largest clock delays. To clear the clock tree display go to Clock -> Display -> Clear Clock Tree Display. The Display Clock Tree menu is shown in the Figure A.14.

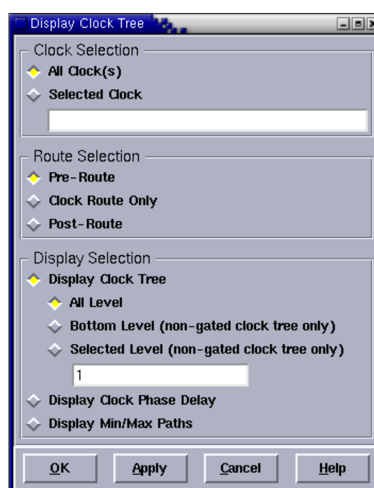

Figure A.14. Display Clock Tree menu

The Clock Tree Synthesis Report can be saved by going to Clock -> Report Clock Tree... and choosing the correct file name in Clock Tree Synthesis Report: sender.ctsrpt. This file can be later viewed for more detailed analysis.

Now the design is ready to be routed. Go to Route -> WRoute... Leave the Basic tab at its defaults. In the Advanced tab go to Search and Repair, check Run Automatically and Full Search and Repair. Choose Allow Modification of Prerouted Regular Nets at: 3 Search and Repair Pass. Then click OK. The routed design is presented in the Figure A.15.

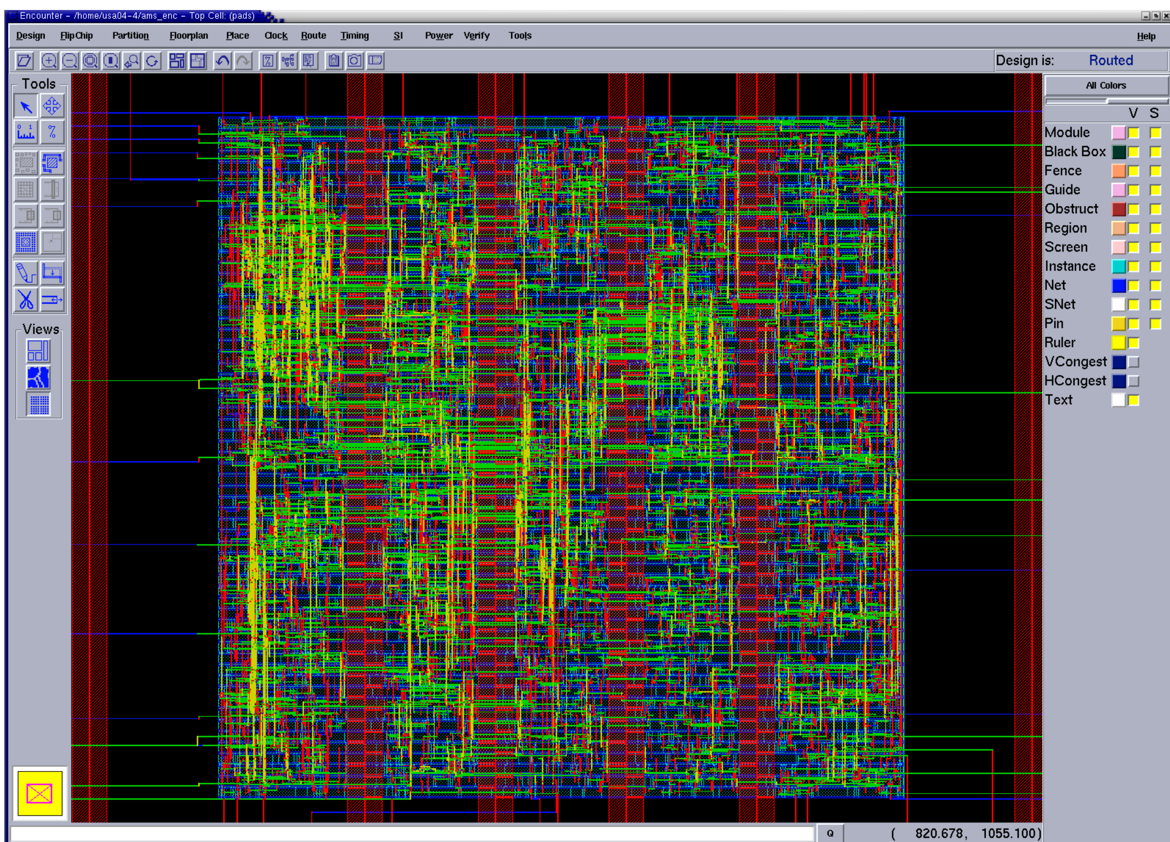

Figure A.15. Design core after routing with WRoute.

After Routing go to Route -> Metal Fill... This fills the empty spaces in the metallization layers in order to smoothen the chip surface and also enhance its performance. Choose

Connection: Tie High/Low to nets: VSS VDD. In the Layer Selection section all layers should be checked. Click OK. Design view after metal filling is shown in the Figure A.16.

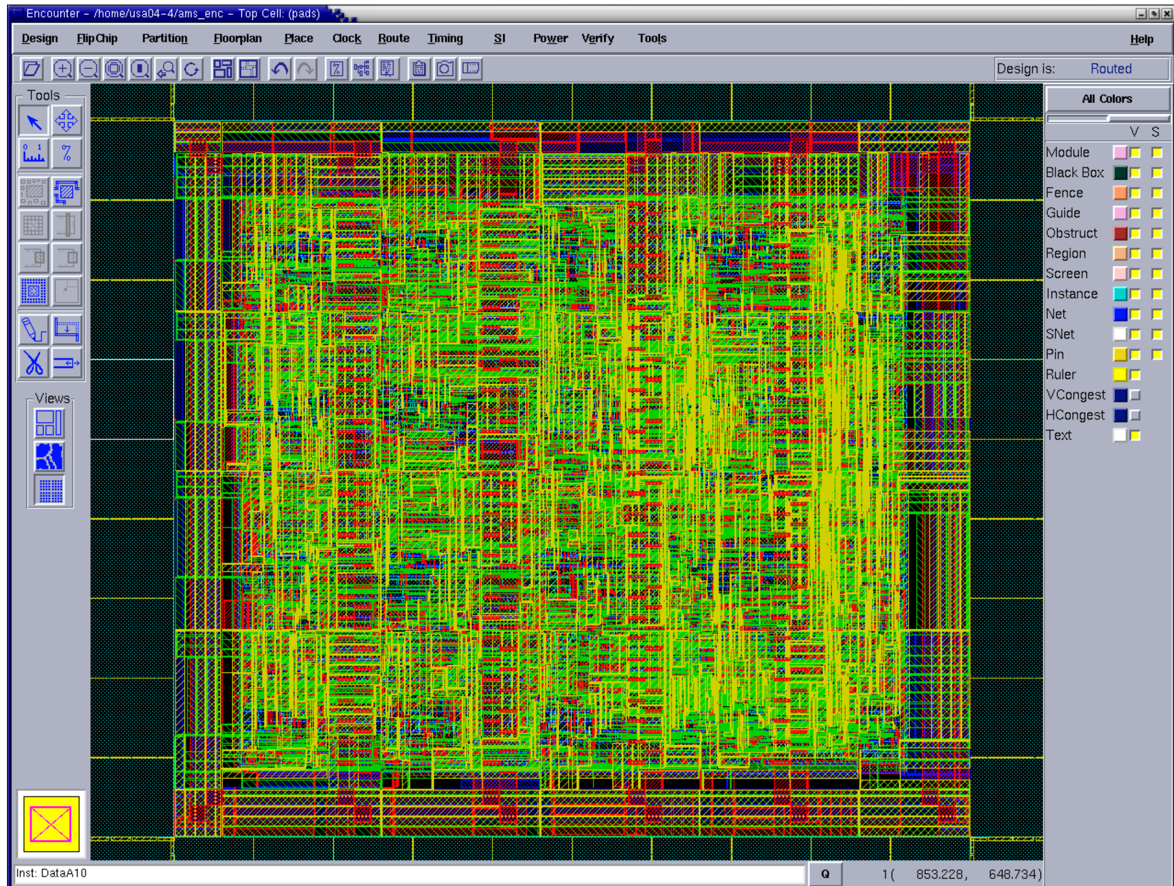

Figure A.16. Design after performing Metal Fill

After the design is routed the SDF file can be created again. Go to Timing -> Extract RC..., enter Save Cap to sender.cap and click OK. Then go to Timing -> Calculate Delay..., check Ideal Clock and enter SDF Output File: sender_cadence_encounter.sdf. After clicking OK the SDF file is generated. What is still to be done is generation of GDS file that will be needed for the layout generation in Cadence Virtuoso Layout Editor. Go to Design -> Save GDS... and enter: Output Stream File sender.gds. The other options can be kept at their defaults. Clicking OK generates GDS file.

Cadence Virtuoso Layout Editor

After importing all the necessary files Cadence Virtuoso Layout Editor can be opened. To run the Cadence environment write the following in the console:

/cad/cadence/bin/cdsams370 c35b4 &

Remember to execute this command in the project directory. The AMS Cadence version 3.70 of CMOS technology with 4 layers of metallization will be opened. First a library for the design should be created. In the Library Manager window go to File -> New -> Library... and in the New Library window that appears write

Name: sender

Click OK. In the next window choose the option that the library should be attached to an existing technological library:

You can: Attach to an existing techfile

Click OK. Then a window appears where a choice of technological libraries is given. Choose the c35b4 technology library by clicking:

TECH_C35B4

This is illustrated in the Figure A.17.

Figure A.17. Attach Design Library to Technology File

Clicking OK creates the desired library. It can be seen in the Library Manager window. Now the layout of the design should be imported. In the main icfb window go to File -> Import -> Stream... to import the previously created GDS file. The Stream In... window is presented in the Figure A.18.

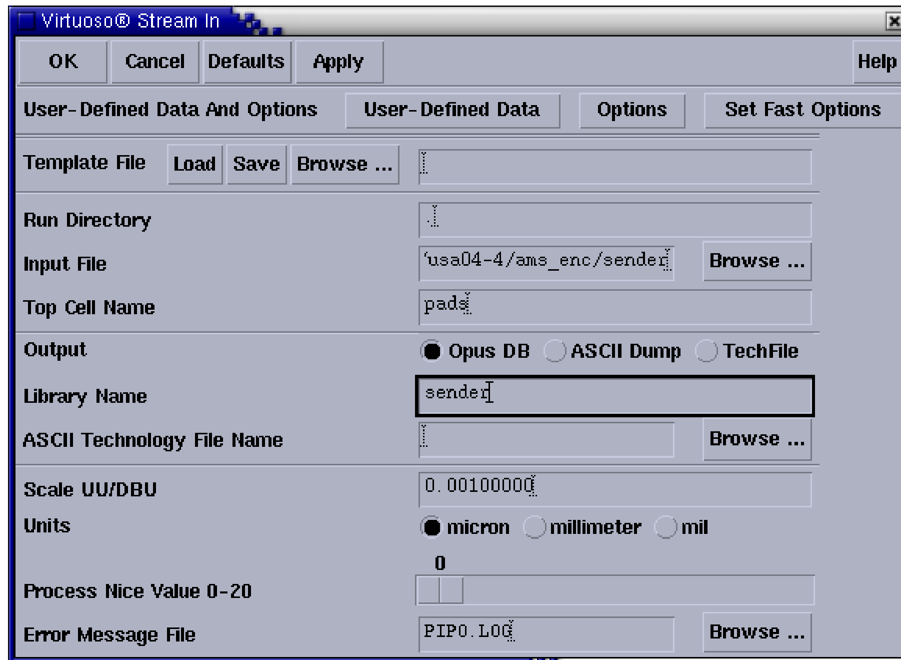

Figure A.18. Stream In window for importing GDS stream file

Enter the following data:

Input File: /project_directory/sender.gds

Top Cell Name: pads

Library Name: sender

Next, open the Options window, which is shown in the Figure A.19. Leave all options at their defaults except for 'Retain Reference Library (No Merge)', which should be checked. Apart from that it is better to increase the 'Hierarchy Depth Limit' from default 20 to 50 in order to make sure that all layers are placed and routed correctly. Click OK. At this stage the design is imported. However, only placed instances is visible so far and the routing layers still need to be imported. A screenshot of the project layout after importing GDS file is shown in the Figure A.20.

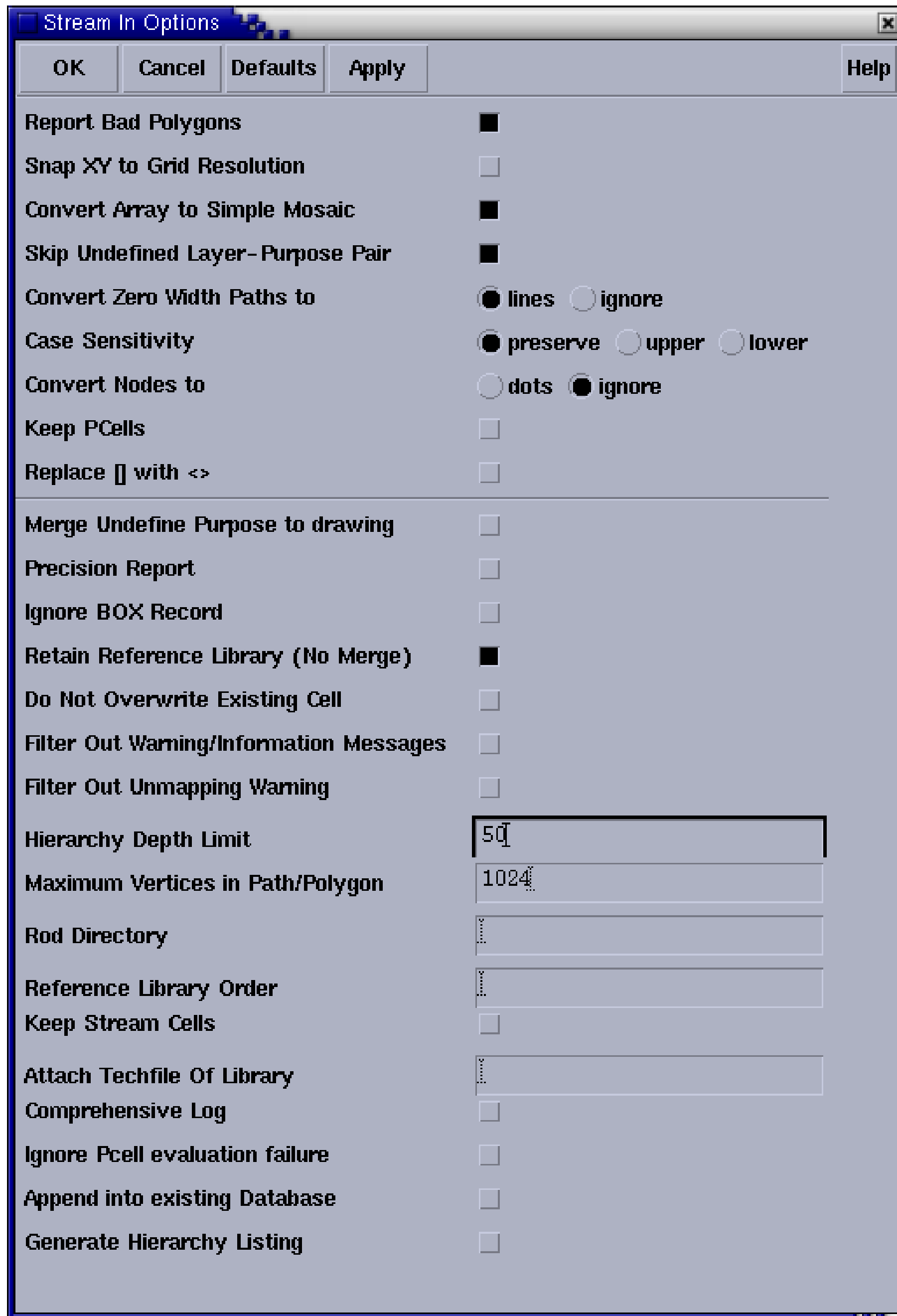

Figure A.19. Stream In Options window

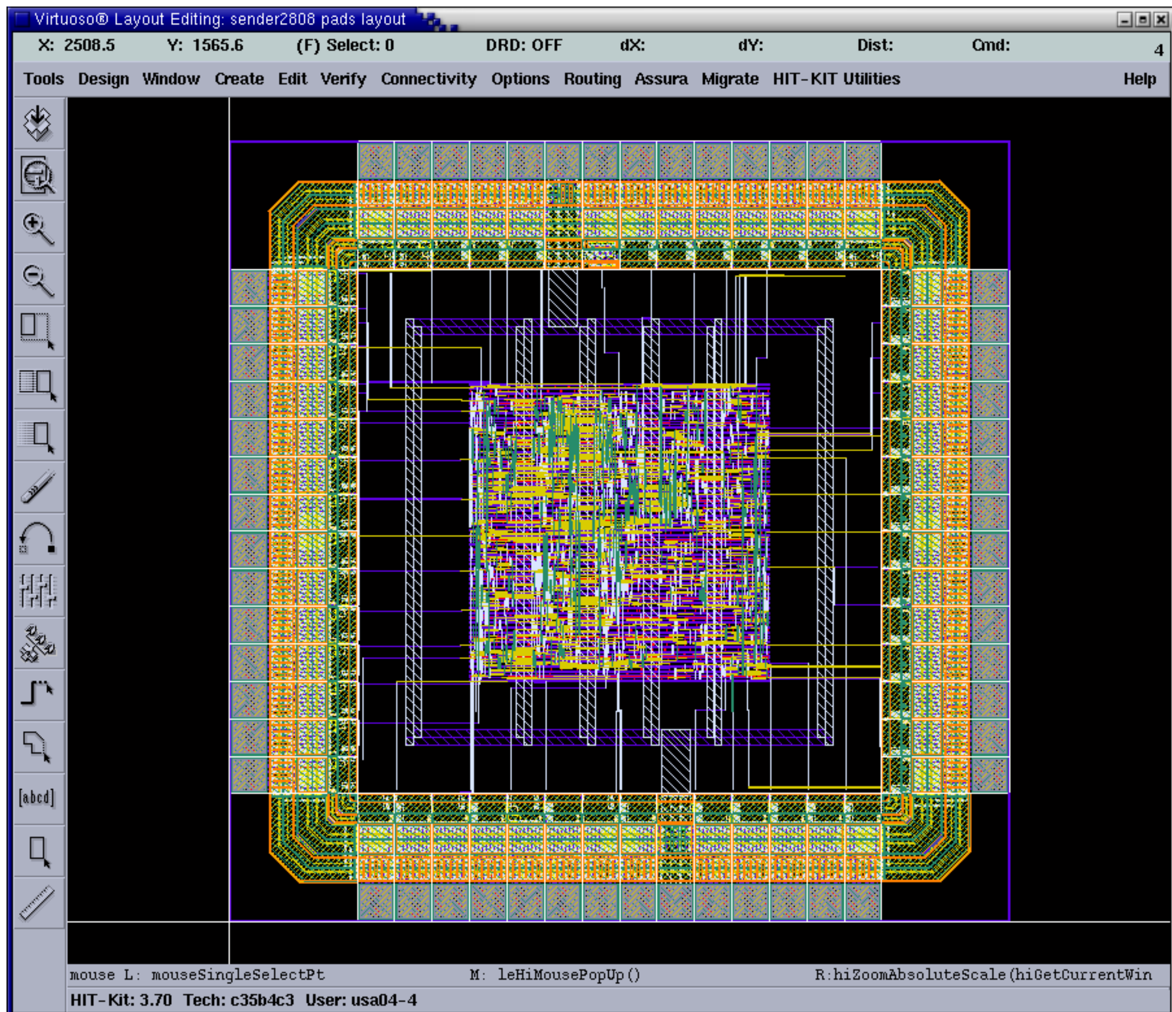

Figure A.20. Complete layout of the design with routing and connections to pads and power rings

At this stage of the project the layout is complete. Placed instances can be seen as well as routing and pads connections. Also power rings and stripes are connected properly. The layout is ready to be checked and saved. Now, the Layout Versus Schematic verification can be performed. First the schematic needs to be generated. To do this, in the icfb window go to File -> Import... -> Verilog... The Verilog In window is shown in the Figure A.21.

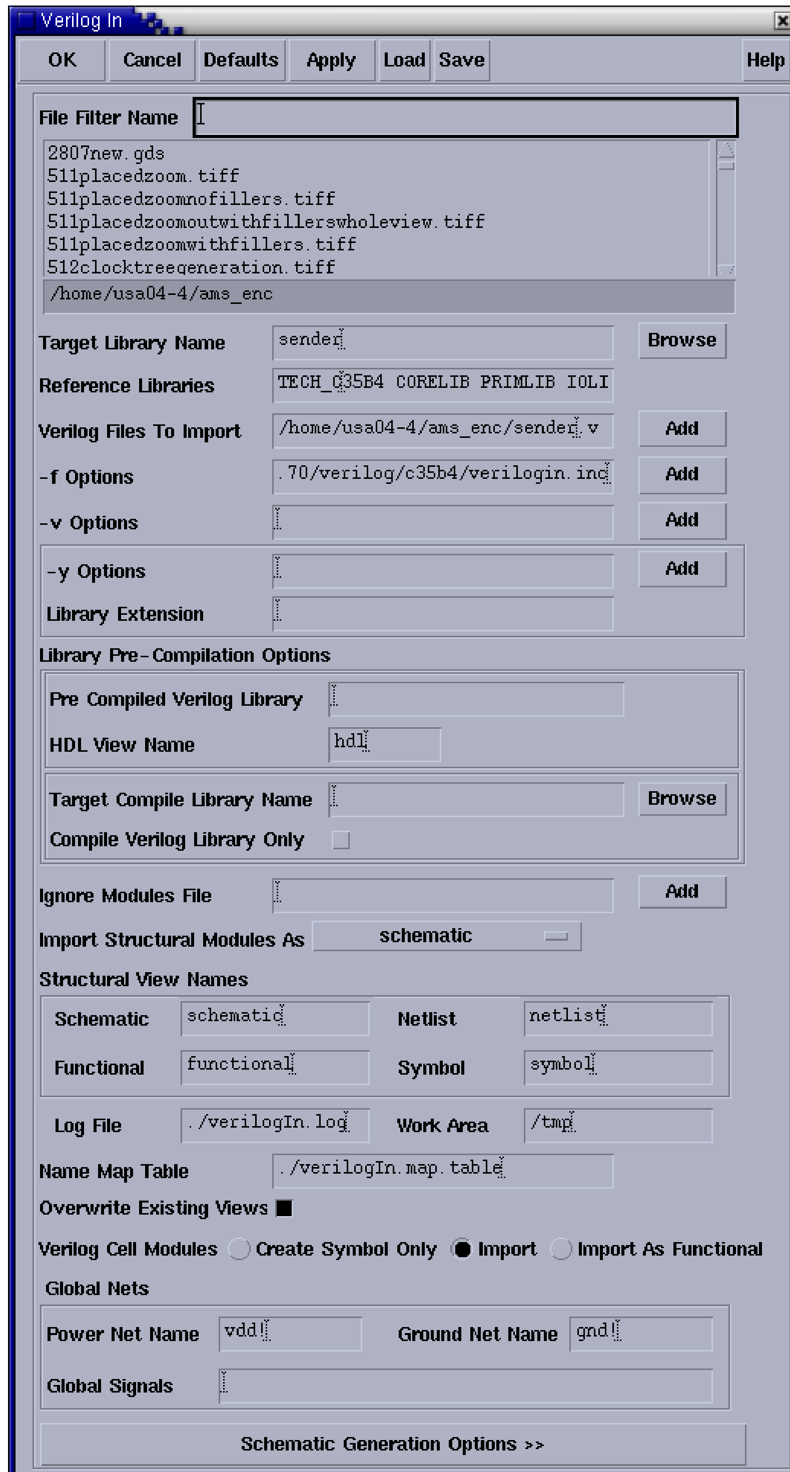

Figure A.21. Verilog In import window

To see the generated schematic choose the schematic view in the sender cell in Library Manager. The schematic of the top module is presented in the Figure A.22.

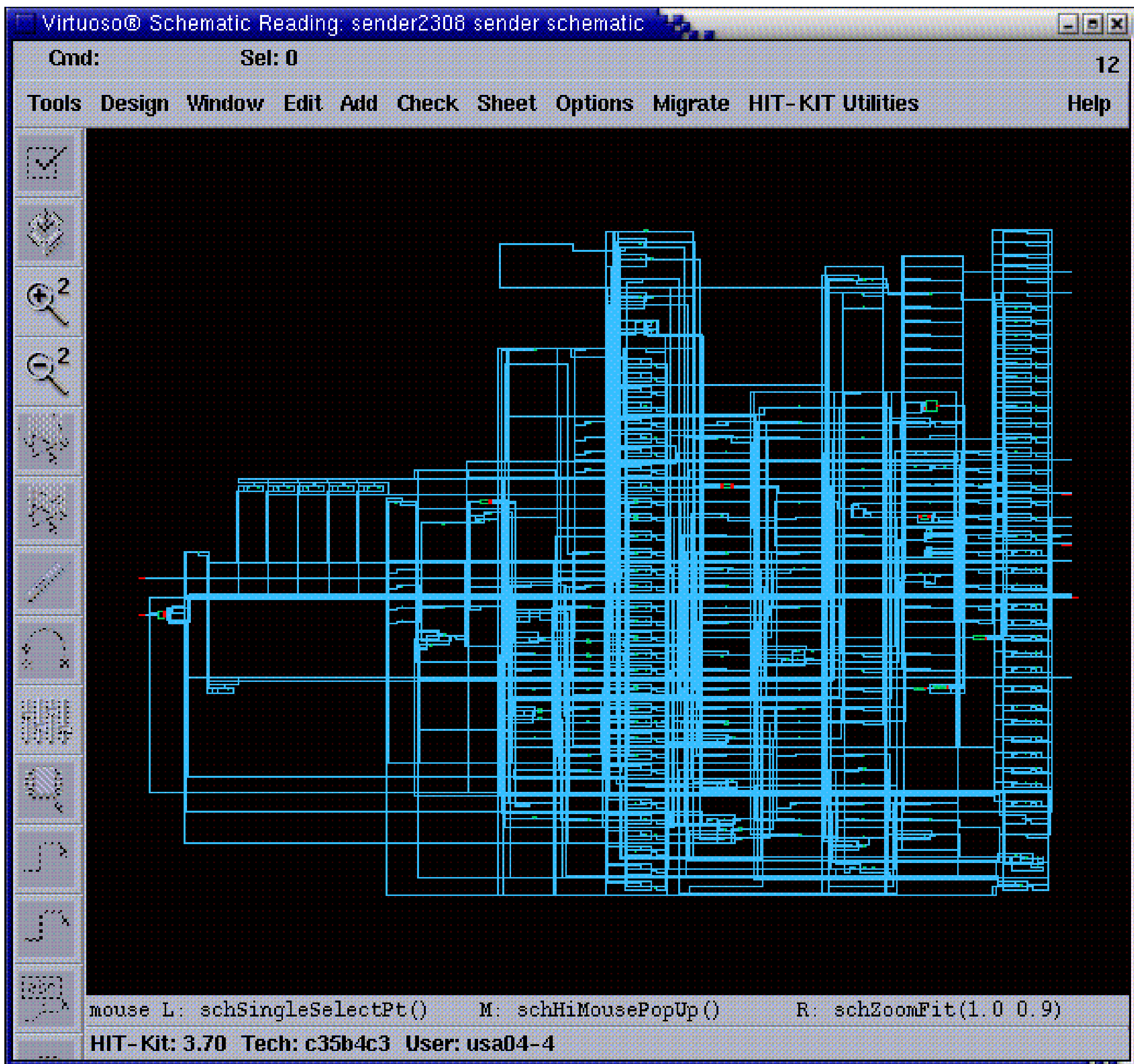

Figure A.22. Schematic view in Virtuoso Schematic Editor

Now, the Layout Versus Schematic analysis can be run. Go to Assura... -> Run LVS... The Assura Run LVS window is presented in the Figure A.23.

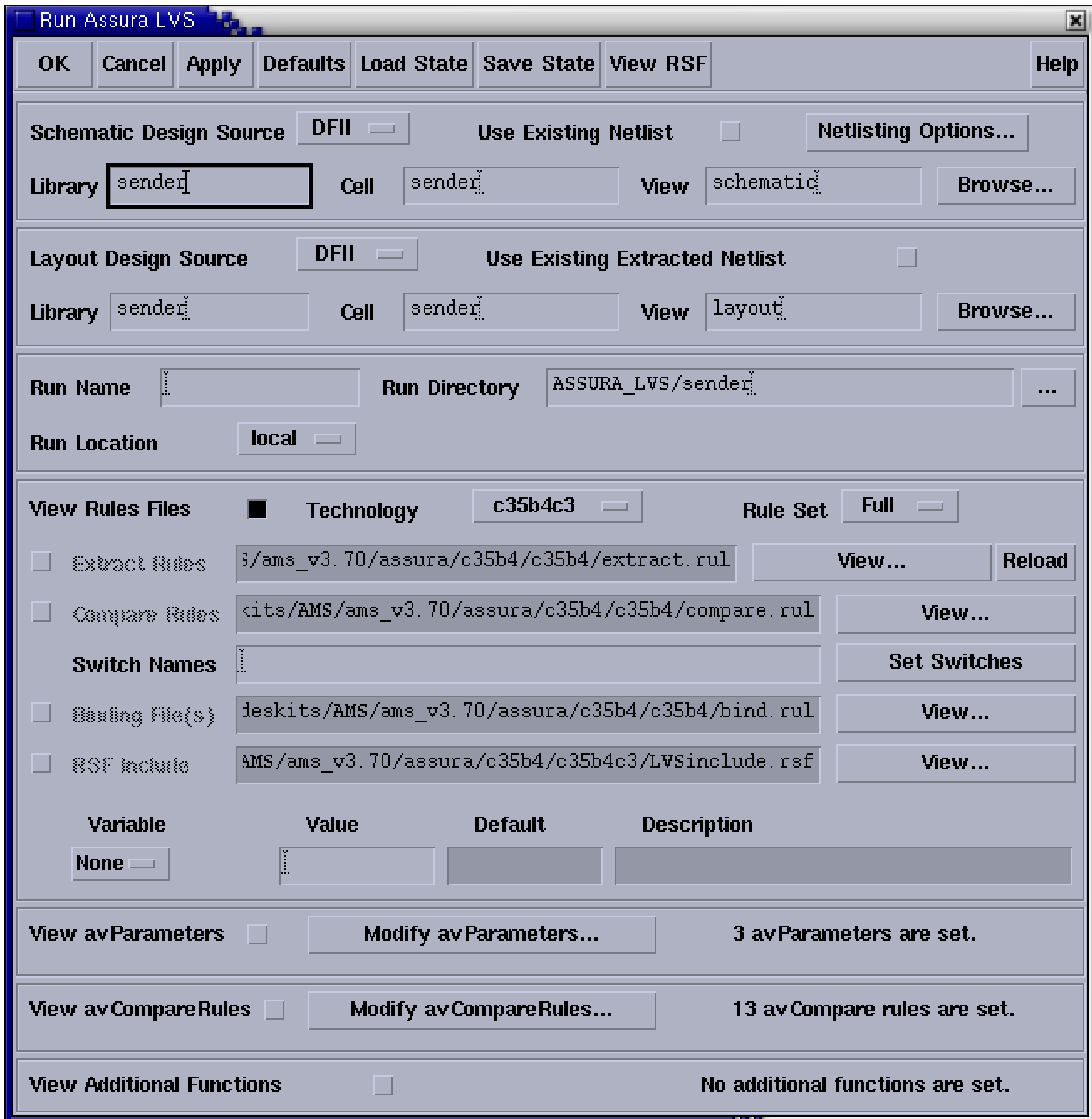

Figure A.23. Assura Run LVS window

In the window that appears, shown in the Figure A.24, choose Watch Log File... to observe the progress of simulations.

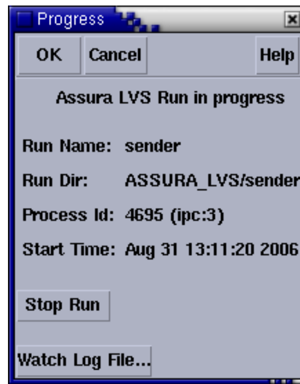

Figure A.24. Assura LVS Progress window

After LVS is performed one can view the results by opening the Debug window, which is prompted after the simulations. Extraction and comparison errors can be observed. This is shown in the Figure A.25 and A.26, respectively.

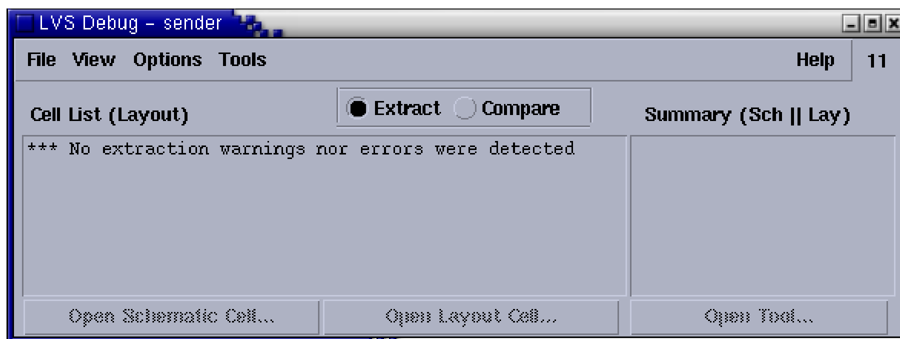

Figure A.25. LVS Debug window with Extract results

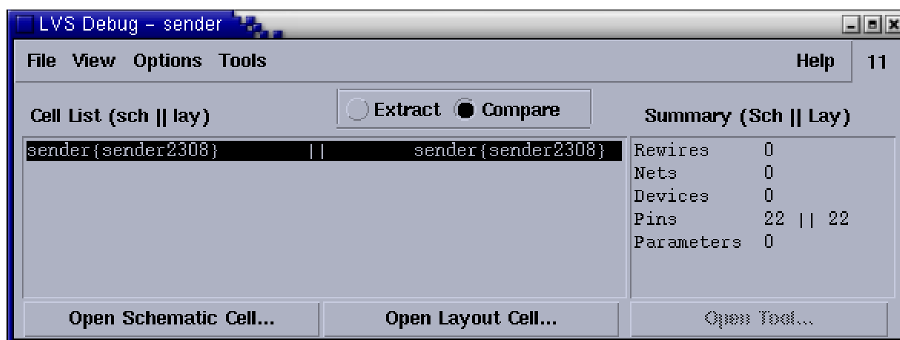

Figure A.26. LVS Debug window with Compare results

The errors for Pins come from the fact that the brackets used for bus definition are different in layout definition and in the imported structural Verilog formats, thus can be ignored. All other cheks are successful. After performing the LVS the layout can be considered checked and the post-layout simulations can be performed to verify the functionality of the design. These can be done using Aldec Active HDL environment. It should be noted that in this software the generated structural Verilog file is used together with the SDF file, either of the generated ones, as was described earlier. After generating layout the device should work as from the behavioural description. This completes the tutorial for layout generation from the behavioural code written in VHDL.

Appendix B: List of Abbreviations

The following appendix summarises the abbreviations used throughout this thesis.

Abbreviation	Comment
ADC	Analog to Digital Converter
AMS	Austria Microsystems
ASIC	Application Specific Integrated Circuit
CMOS	Complementary Metal Oxide Semiconductor
CONF	Configuration file format
COTS	Commercial Off-The-Shelf
CRC	Cyclic Redundancy Check
DRC	Design Rule Check
EDAC	Error Detection And Correction
EEPROM	Electrically Erasable Programmable Read-Only Memory
EIA-232	Electronic Industries Alliance standard for the serial communication
FIFO	First In First Out
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GDS	Graphic Data System
HDL	Hardware Description Language
IC	Integrated Circuit
IEL	Ionizing Energy Loss
ILC	International Linear Collider
IO	Input Output
KERMA	Kinetic Energy Released in MAtter
LEF	Library Exchange Format
LET	Linear Energy Transfer
LIB	Library format
LVS	Layout Versus Schematic
MBU	Multiple Bit Upset
MOS	Metal-Oxide-Semiconductor
MOSFET	Metal-Oxide-Semiconductor Field-Effect Transistor
NIEL	Non-Ionizing Energy Loss
PKA	Primary Knock-on Atoms
PKS	Physically Knowlegdeable Synthesis
RadFET	Radiation-Sensing Field-Effect-Transistor

Rad-hard	Radiation-hardened
Rad-tol	Radiation-tolerant
RTL	Real Time Logic
SCR	Silicon-Controlled Rectifier
SDC	Timing constraints file format
SDF	Standard Delay Format
SEBO	Single Event Burn Out
SEE	Single Event Effect
SEFI	Single Event Functional Interrupt
SEGR	Single Event Gate Rupture
SEL	Single Event Latch-up
SES	Single Event Snapback
SET	Single Event Transient
SEU	Single Event Upset
SHE	Single Hard Error
SOI	Silicon On Insulator
SRAM	Static Random Access Memory
TLF	Timing Library Format
VHDL	Very-high-speed integrated circuit Hardware Description Language
VHSIC	Very-High-Speed Integrated Circuit