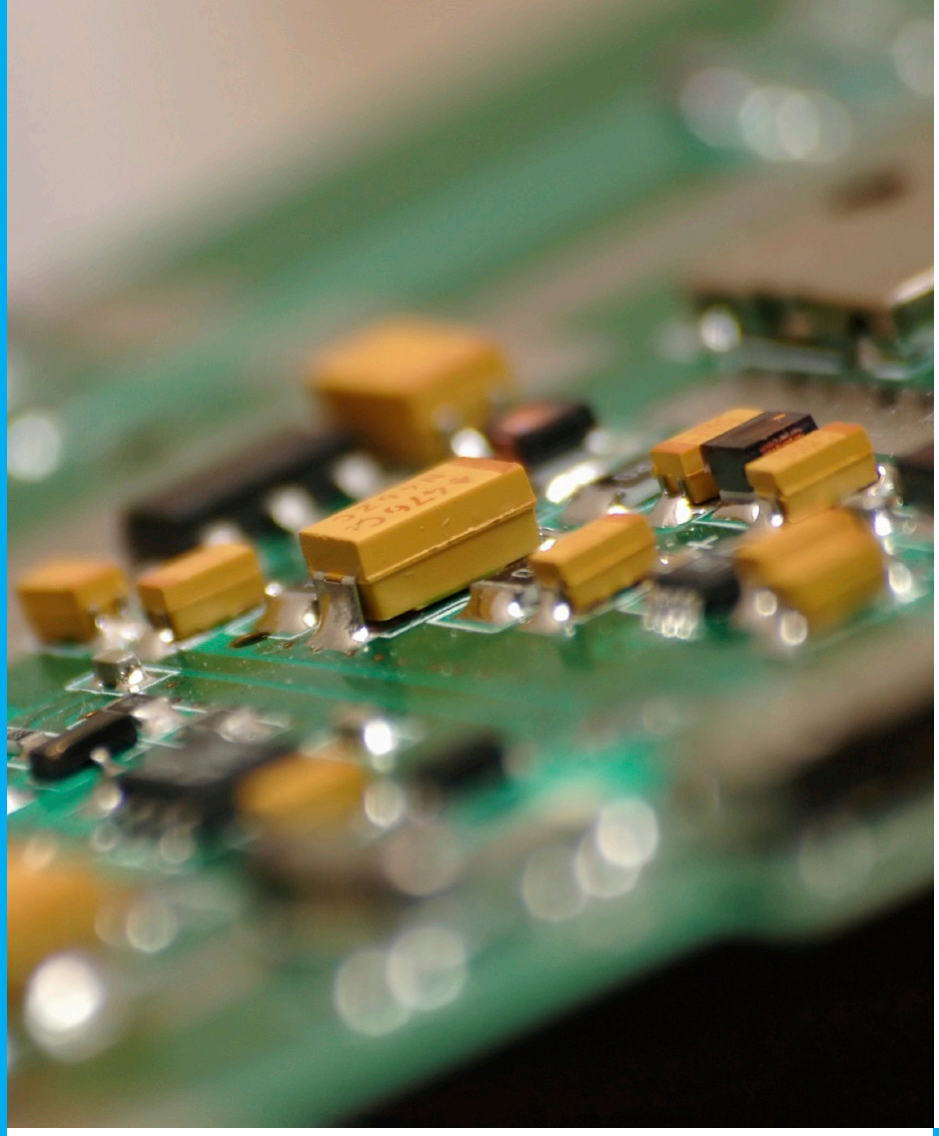

TECHNOLOGY • RESEARCH HIGHLIGHTS • VISIONS • SCIENCE •

10

The current state of FPGA technology in the nuclear domain

Jukka Ranta

VTT

VTT TECHNOLOGY 10

The current state of FPGA technology in the nuclear domain

Jukka Ranta
VTT

ISBN 978-951-38-7622-7 (URL: http://www.vtt.fi/publications/index.jsp)
ISSN 2242-122X (URL: http://www.vtt.fi/publications/index.jsp)

JULKAISIJA – UTGIVARE – PUBLISHER

VTT
PL 1000 (Vuorimiehentie 5, Espoo)
02044 VTT
Puh. 020 722 111, faksi 020 722 4374

VTT
PB 1000 (Bergsmansvägen 5, Esbo)
FI-2044 VTT
Tfn +358 20 722 111, telefax +358 20 722 4374

VTT Technical Research Centre of Finland
P.O. Box 1000 (Vuorimiehentie 5, Espoo)
FI-02044 VTT, Finland
Tel. +358 20 722 111, fax + 358 20 722 4374

The current state of FPGA technology in the nuclear domain

Jukka Ranta. Espoo 2012. VTT Technology 10. 62 p.

Abstract

Field programmable gate arrays are a form of programmable electronic device used in various applications including automation systems. In recent years, there has been a growing interest in the use of FPGA-based systems also for safety automation of nuclear power plants. The interest is driven by the need for reliable new alternatives to replace, on one hand, the aging technology currently in use and, on the other hand, microprocessor and software-based systems, which are seen as overly complex from the safety evaluation point of view.

This report presents an overview of FPGA technology, including hardware aspects, the application development process, risks and advantages of the technology, and introduces some of the current systems.

FPGAs contain an interesting combination of features from software-based and fully hardware-based systems. Application development has a great deal in common with software development, but the final product is a hardware component without the operating system and other platform functions on which software would execute.

Currently the number of FPGA-based applications used for safety functions of nuclear power plants is rather limited, but it is growing. So far there is little experience or common solid understanding between different parties on how FPGAs should be evaluated and handled in the licensing process.

Keywords nuclear power, instrumentation and control, I&C, FPGA

Preface

This report has been prepared as a part of the Coverage and Rationality of the Software I&C Safety Assurance (CORSICA) project. CORSICA is a part of the Finnish Research Programme on Nuclear Power Plant Safety 2011–2014 (SAFIR2014).

The goal of the project was to survey field programmable gate array (FPGA) technology and the situation of its use in nuclear power generation. FPGA technology has become a topic of interest as a potential technology suitable for the safety-critical I&C systems of nuclear power plants. The primary task was to collect an information package that can serve the interested parties, including utilities and regulators, when considering FPGA-based systems offered for use in Finnish nuclear power plants. In addition to the technology as such, the risks, advantages and special characteristics were also to be considered.

Espoo, December 2011

Contents

List of symbols

ASIC	Application Specific Integrated Circuit
BiS	Built-in Software
BIST	Built-in Self-test
CAB	Configurable Analog Block
CCF	Common Cause Failure
CEC	Complex Electronic Component
CLB	Configurable Logic Block
CMOS	Complementary Metal Oxide Semiconductor
CPLD	Complex Programmable Logic Device
CPU	Central Processing Unit
EDA	Electronic Design Automation
EDAC	Error Detection And Correction
EDIF	Electronic Design Interchange Format
EEEPROM	Electrically Erasable Programmable Read-Only Memory
EM	Electromigration
EMC	Electromagnetic Compatibility
EPLD	Erasable Programmable Logic Device
EPROM	Erasable Programmable Read-Only Memory
ESL	Electronic System Level
FPAA	Field Programmable Analog Array
FPGA	Field Programmable Gate Array
HCI	Hot Carrier Injection
HDL	Hardware Description (Definition) Language
HDVL	Hardware Description and Verification Language
HIT	Heavy Ion Tolerant
HMI	Human Machine Interface

I&C	Instrumentation and Control
IAEA	International Atomic Energy Agency
IC	Integrated Circuit
IDE	Integrated Development Environment
IP (Core)	Intellectual Property Core
LAB	Logic Array Block
LUT	Look-Up Table
MOSFET	Metal Oxide Semiconductor Field-Effect Transistor
NBTI	Negative Bias Temperature Instability
NPP	Nuclear Power Plant
P/E Cycle	Program/Erase Cycle
PCEC	Programmable Complex Electronic Component
PDED	Pre-Developed Electronic Design
PDS	Pre-Developed Software
PLD	Programmable Logic Device (see also EPLD, CPLD)
RTL	Register Transfer Level
SEE	Single Event Effect
SEFI	Single Event Functional Interrupt
SEL	Single Event Latch-up
SER	Soft Error Rate
SET	Single Event Transient
SEU	Single Event Upset
SoC	System on Chip
SOI	Silicon On Insulator
SRAM	Static Random Access Memory
STA	Static Timing Analysis
TDDB	Time Dependent Dielectric Breakdown
TID	Total Ionizing Dose
TMR	Triple Modular Redundancy
V&V	Verification and Validation
VHDL	Very high speed integrated circuit (VHSIC) HDL
VHSIC	Very High Speed Integrated Circuit, see VHDL
VLSI	Very Large Scale Integration

1. Introduction

Interest in the use of field programmable gate array (FPGA) technology in nuclear power plant (NPP) automation has increased in recent years. The technology is not new and is widely used in other areas. But it is new, in particular in safety-related systems, in NPPs. Old analog and micro-controller systems are becoming obsolete and need to be replaced. At the same time, decisions on the technology to be used in new plants need to be made. The experience of software-based systems is that demonstrating their reliability and safety in the licensing process is difficult and laborious. FPGAs are seen as an option that provides flexibility and capability similar to software but with the lower complexity, simpler system structure, and improved performance characteristic of hardware. There are other advantages to using FPGAs, but there are also risks involved. The suitability, reliability, and possible problems of licencing a novel technology in the nuclear power generation domain all have uncertainties.

FPGAs are just one of the new trends in instrumentation and control technology (I&C). See, for example, the survey reports prepared for the United States Nuclear Regulatory Commission (U.S. NRC) [NRC, 2003], [NRC, 2006] and [NRC, 2009a]. The FPGA-specific reports by the Electric Power Research Institute [EPRI, 2009, 2011] indicate a strong interest on behalf of the industry, while the review guidelines document [NRC, 2010b] indicates the interest of regulators to prepare for increased use of FPGAs and similar technologies.

The FPGA device itself is a semiconductor silicon chip with a regular array of transistors. As such it has no functions. The device can be configured (programmed) for a task after manufacture by configuring the connections between the components on the chip. This allows standardised mass production of the devices independent of the applications for which they are intended. And this, in turn, provides advantages for the development process and manufacture.

The development of an FPGA application has strong similarities with software development. Both use rather high abstraction level programming/description languages and the design process is heavily dependent on software tools. The design of complex systems is easy, but verification of the correct functioning of those systems is not. A significant difference is that an FPGA application does not need an operating system, hardware drivers, or other similar platform. However, such a computing environment can be configured onto a device with sufficient

capacity. In addition, FPGAs use parallel processing with dedicated hardware for each function instead of executing a program one instruction at a time.

Reasons to use FPGAs include the hope for an easier licensing process compared to microprocessor and software-based alternatives. On the other hand, the simpler analog systems are becoming obsolete. FPGAs can provide fast response times with dedicated hardware for each task and reduced concerns about tasks interfering with each other. Cyber security issues are also considered to be less with FPGAs than with software, because an FPGA can contain all of the memory and functions and it is difficult, or impossible for certain technologies, to alter it, that is, make the device run viruses.

The risks associated with FPGAs and software-based systems have much in common. The correctness of the functioning may be difficult to prove due to complexity of the system and the layers of design tools between the requirements specification and the final product. Because the technology is new in the nuclear power generation industry, there is little experience of what it is most suited for or how it should be implemented.

Currently there are only a few licensed and operational FPGA-based systems performing safety-related functions at nuclear power plants. A rather large number of them are in Ukrainian and Bulgarian NPPs and were installed as parts of modernisation projects. Another major project is going on in France and also deals with modernisation. Mostly, the other developed applications are experimental and are intended for testing in a simulator and developing experience and expertise.

A standardisation process is underway to develop the IEC 62566 standard for use of hardware description language programmable devices in nuclear power plants. It is expected to be published in spring 2012. Due to the nature of FPGAs having a mixture of features from hardware and software, both types of standards can be relevant for licensing.

This report aims to give a basic overall working knowledge about FPGA technology. From this report, the reader should gain sufficient background information and readiness to access the more topically specialised literature. The main focus is on safety and reliability issues. In Chapter 2, physical technological aspects are considered. Chapter 3 presents the application design process and related issues. The reasons to use FPGA technology along with selected risks and security issues are considered in Chapter 4. Some FPGA-based systems in development or already in use are presented in Chapter 5. A summary is given in Chapter 6.

2. FPGA technology – hardware aspects

Field programmable gate arrays appeared on the market in the late 1980s. They are a step in the continuum of evolution of integrated circuits (IC). They are one of the programmable logic devices (PLD) and, due to their flexibility and capacity for various applications, are considered complex electronic components (CEC). Their strength is the idea of a functionally blank device with the potential to be configured for various implementations, thus allowing mass production of devices independent of the final application. An FPGA device can be programmed after manufacture to execute the desired logic or computations. The chip contains a large number of similar basic elements in a fixed structure. The programming is done by configuring these basic elements and the connections between them. The words programming and configuring are used interchangeably in the literature.

Physically, an FPGA chip is CMOS (complementary metal oxide semiconductor) technology, which is the most common technology in modern ICs. The fundamental components of CMOS are paired P- and N-type semiconductor transistors, which implement the logical gates. Functions are designed by configuring the wiring between the gates.

Technologically, the basics of FPGA have stayed the same and are the same as in most digital computing technology. The capacity, though, has increased tremendously. The amount of logical operations that a chip can contain, measured in the number of logical gates on the chip, has increased from a few thousand to millions. In addition, separate RAM memory and dedicated blocks for, for example, numerical computations or even microprocessors are available in the more advanced chips. This leads to the possibility of producing system on a chip (SOC) applications. SOC is something of a sliding concept but means that all functions (an entire computer) are on the same chip.

An FPGA application consists of the device (chip and packaging with contact pins), the circuit board, and the auxiliary hardware on the board. In addition, there are the cabinets to hold the boards and wiring for measurement signals, communication with other devices or instrumentation, and power. The board can also contain separate hardware for signal processing (e.g., conversion between analog and digital signals and filtering), input/output connectors, memory, and other processing units.

Architecturally, FPGAs consists of I/O blocks, logical gates which form configurable logic blocks (CLB), and the wiring between them. The logic blocks can execute simple logic themselves, such as a 4-input lookup table, and typically have flip-flops for memory. The switches of the wiring allow the connection of I/O blocks and logic blocks to generate more complex logic.

Three types of FPGA technology are available with regards to the programming technology. These are antifuse, SRAM, and flash. Each one stores the configuration in a different way, which affects the way the chip can be used and also the physical properties, such as aging and radiation tolerance. Aside from the programming technology, all three use CMOS technology to implement the actual logic operations.

A distinct difference between an FPGA application and a microprocessor application running software is in parallel computing. An FPGA executes all of the logic on each clock cycle, whereas a microprocessor executes one program instruction per cycle.

2.1 CMOS

Complementary metal oxide semiconductor (CMOS) technology was developed in the 1960s. The basic components are MOSFETs (MOS field effect transistor). The term “complementary” in CMOS refers to the use of complementary pairs of N-type and P-type MOSFETs to construct the logical gates (AND, OR, NOR, etc.), see Figure 1. An N-type transistor lets current pass between the source and drain when the voltage of the gate electrode is positive. A P-type transistor lets current pass when the gate voltage is negative. When one of the paired transistors is on, the other is off, and therefore there is an electric current and power consumption only during the short time the gates are changing state. This is the main advantage of CMOS over the other technologies for building integrated circuits from semiconductors. In other technologies, in one of the states there is a constant current through the transistor.

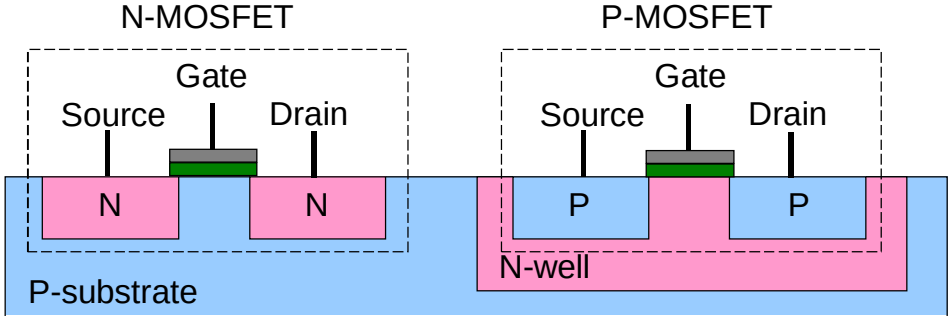

Figure 1. P- and N-type transistors on a P-substrate.

2.2 Chip architecture

The basic parts of an FPGA chip are input/output connections located on the outer edges of the chip, logic blocks, and the connections (wiring routed by switchboxes) between them, as shown in Figure 2. In addition to these, more advanced chips can also have, for example, separate memory, dedicated computation and signal processing blocks, or even a microprocessor. For example, the chip used in [Lu et al., 2010] has a wide range of auxiliary functions.

The main parts of configurable logic blocks (CLB), also known as logic array blocks (LAB), are typically lookup tables and flip-flops [NI, 2008]. These are built from the elementary gates and provide a higher abstraction level for programming the device. Different architectures can have a different number of lookup tables and flip-flops per CLB. Flip-flops, or shift registers, provide one bit of memory each and the lookup tables can describe combinatorial logic. Hence, together they enable the construction of sequential logic with memory. Currently, four or six input lookup tables are typically used.

The wiring runs between the logic blocks and is controlled by switchboxes. The switchboxes connect the logic blocks to the wires and the wires to each other. There is a limited quantity of wiring resources and the design, placement, and distribution of the logic onto the logic blocks must take this into account.

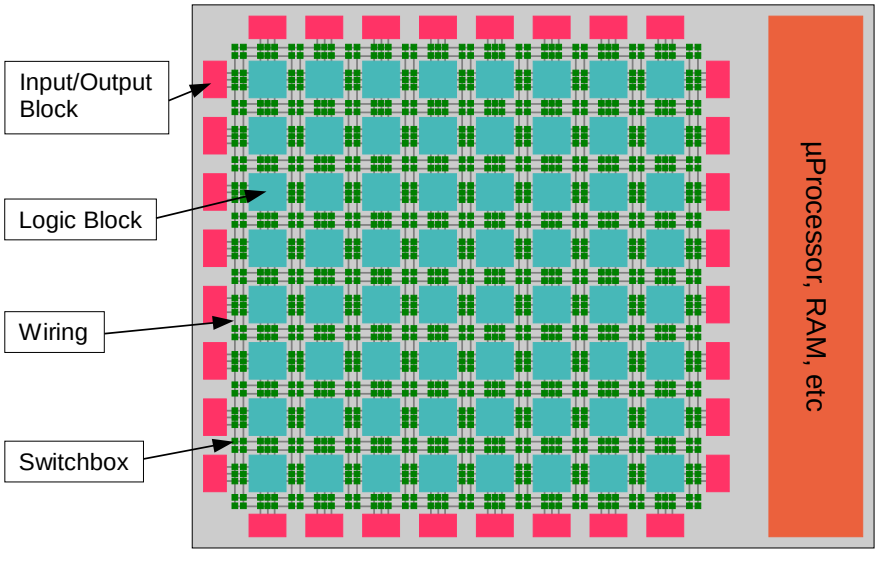

Figure 2. A regular grid FPGA layout.

2.3 Antifuse, SRAM, and flash technologies

The implementation of FPGA technology comes in three “flavours”: antifuse, SRAM (static random access memory), and flash. From a user perspective, the major difference is in how the chip is programmed, that is, how the logic is stored on the chip and how the chip retains its programming. The name refers to the technology used for implementing the switches that control the configuration and connections of wires, and thereby the programming. In all cases, the logic gates themselves are implemented using the same CMOS technology.

A SRAM chip holds the logic in memory only while the chip is powered, that is, it is *volatile* memory, and the chip must be programmed as a part of the start-up sequence. The technology is the same as for the working memory of computers. A chip based on flash technology is reprogrammable like SRAM but it retains the programming without power. Flash technology is essentially the same as in USB memory sticks. The third technology is antifuse, which is not reprogrammable and maintains its programming without power.

SRAM technology holds the programming as the state of groups of cross-coupled transistors. See Figure 3A. Each group forms a bit of memory and determines if a switch is open or closed. While the supply power is provided, the transistors reinforce each other to hold the system in a state of “1” or “0”. If the operating power is lost, the state of the transistor system is lost. SRAM consists only of CMOS technology and tends to be at the forefront of development and used in devices which have the highest performance capacity.

The term antifuse comes from fuse, which is a device designed to cut power when the voltage or current through it increases too much. Antifuse functions in the opposite way. It starts out with high resistance and does not let current pass. After it is triggered by, for example, the application of high voltage and current through it, it changes so that its resistance drops and it lets current pass. See Figure 3C. The change is mechanical and irreversible, hence the non-reprogrammability. An antifuse can be implemented by placing a thin layer of non-conducting amorphous silicon between two metal conductors. With the application of sufficient voltage to drive a current through the insulation, the silicon turns into a conducting poly-crystalline silicon-metal alloy.

Flash technology is a variant of EEPROM (electrically erasable programmable read-only memory). The term flash comes from the erasure process of large blocks of data. The method of storing data is based on transistors with a “floating gate”, which controls the behaviour of the transistor. See Figure 3B. The floating gate is fully insulated and therefore holds a charge without a power supply. Writing and erasing are performed using a high voltage, which causes the charge in the floating gate to change through hot-electron injection and quantum tunnelling. Flash memory withstands only a limited number of program/erase (P/E) cycles, ranging from a typical value of 100 000 up to a million.

For the design of logical functions, the choice between these technologies has little effect. The differences come into play when physical characteristics, such as

tolerance to operating environment, or operational practices related to, for example, maintenance and data security, are considered. The resources needed for the switches are shown in Figure 4. SRAM requires several transistors, flash requires only one transistor, and antifuse is the smallest and simplest.

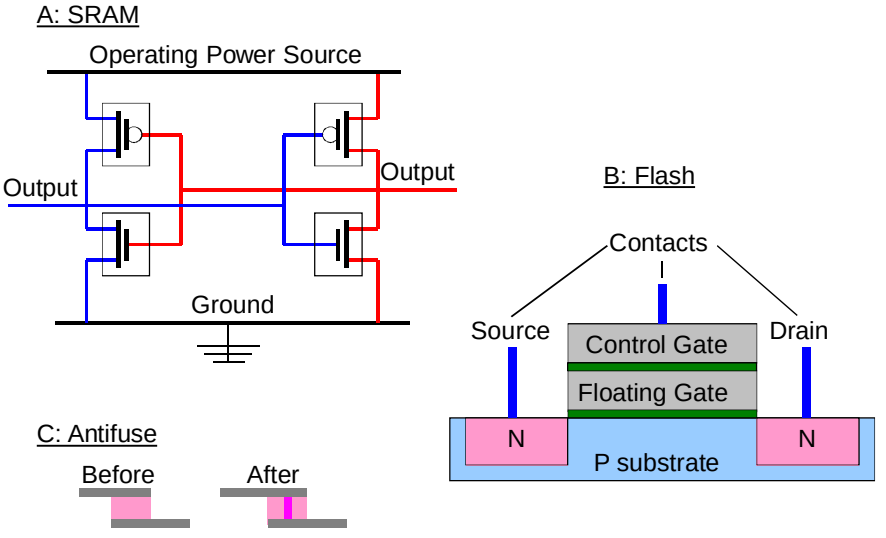

Figure 3. Different technologies for storing FPGA configuration data.

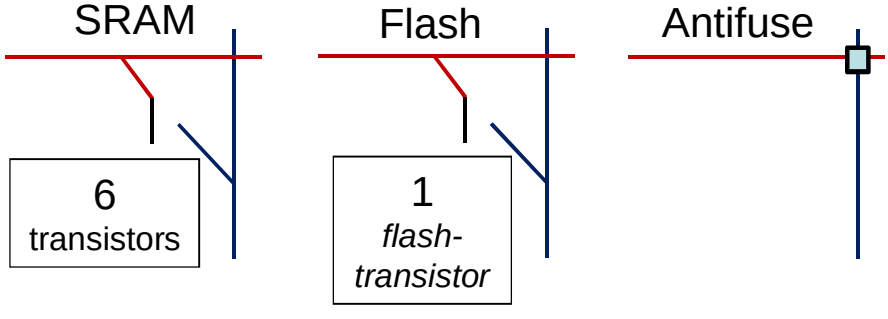

Figure 4. Hardware and structure of switches connecting wiring using different technologies.

2.4 Circuit board and connections

The FPGA device is attached to a circuit board, which provides contacts for data signals and operating power. Connections include those to the process instrumen-

tation, measurement data, set point values, and the internal state of the chip. Connections are also needed for programming the device and the other functions on the chip, such as signal processing. There is often separate hardware on the board for, for example, signal processing to perform analog to digital (and vice versa) conversion.

A basic chip based on antifuse technology has no reprogrammable memory, and therefore a separate memory device is required for changing data such as set point values. On the other hand, SRAM devices require separate memory from which to load the configuration at start-up.

2.5 Electrical and mechanical properties and reliability

2.5.1 Faults

The faults of an integrated circuit can be grouped into permanent faults and transient faults, which can cause incorrect operation, degradation in performance, or complete malfunction. Temporary transient errors are referred to as *glitches*. Gradual aging and degradation can cause the device to malfunction as a sudden event or can lead to a gradual change in performance, as in response time. Manufacturing faults can lead to devices with inferior performance or faster deterioration, or to an entirely non-functional device. Design faults are considered in Chapter 3. Aging, manufacturing, and design faults are all permanent faults. Single event effects (SEE) can be caused by power fluctuation or radiation. A SEE can cause a permanent physical change in the chip, a change in the programming or memory that can be repaired by a restart or “scrubbing”, or an incorrect operating state which resolves in normal operation. In [Ziener, 2010], faults are listed and categorised with their error effects. In addition to the integrated circuit, the system includes the circuit board and other electronic components, which are susceptible to additional faults, such as whisker growth on soldered contacts.

2.5.2 Environmental tolerance

As in other electrical and electronic equipment, there are limits to the amount of physical abuse that FPGA-based systems can sustain and still function properly. The damage can be caused by several different kinds of phenomena in the operating environment. Sufficient stress will always cause the equipment to malfunction. Thus, the normal and extreme conditions of the intended operating environment must be taken into account. The section “3.3.1 Environmental Qualification of System” in the licence amendment of the Wolf Creek Generating Station [NRC, 2009b] serves as an example. It covers the topics and how they were tested for that particular application.

Humidity and **temperature** are tested together and extreme conditions as well as rapid changes in conditions are considered. Humidity refers to non-condensed

air humidity and affects exposed components. Temperature can change the electric properties of materials, alter dimensions of components, and cause damage through thermal shock from rapid temperature change. **Seismic** effects cover acceleration and vibrations from outside forces. Effects of **radiation** are considered in more detail later.

Electromagnetic compatibility (EMC) is the equipment's ability to operate in the intended electromagnetic environment without causing or suffering ill effects to or from other electrical equipment. EMC **susceptibility** considers the effects of external sources of electromagnetic fields, that is, other equipment, on the device in question. EMC **emission** considers the effect the device in question has on other equipment. **Surge** and **electrostatic discharge** are abnormally high voltages affecting the equipment. They can enter through the power or signal lines or as discharges of static electricity on contact with the equipment. In NPPs, there is particular interest in the influence between equipment of different safety categories. There is special concern about the influence of lower safety category equipment on higher safety category equipment.

2.5.3 Aging

There are four main aging processes that wear down IC devices. See, for example, [Bernstein et al., 2006, Stott et al., 2010a and 2010b] and the references therein. In [Bernstein et al., 2006], some of the involved physics is explained. The effects can be a gradual deterioration of performance leading to, for example, timing faults, or a sudden event, such as a short circuit.

Electromigration (EM) is a diffusive process resulting from the flow of electrons in conductors during operation. The momentum of electrons is transferred to the atoms of the conductor and gradually moves them. Positively charged metal ions are pulled towards the cathode by the electrostatic force, while electrons scattering from the ions push them towards the anode. With high enough current density, the force caused by scattering becomes dominant. The larger the current densities are, the faster the atoms move. In time, this leads to a depletion of atoms on the cathode side and a build-up on the anode side. The conductivity will change and eventually the result is a short or open circuit.

Hot carrier injection (HCI) causes charges to form in the insulating oxide layer of the gate. When a current flows in the channel between source and drain, carriers (negative electrons or positive holes) can be injected and trapped in the oxide layer. The term "hot" refers to the energy of the carriers. The gradually accumulating electric field of the charge in the oxide combines with that of the gate and affects the threshold voltage of the transistor. This leads to slower switching speeds, which causes timing faults.

Negative bias temperature instability (NBTI) is caused by trapped charges in the interface of silicon and oxide. This is similar to HCI, but the trapped charges are only positive holes in P-type transistors. The phenomenon is prevalent in high temperatures. The result is a change in the threshold voltage and timing performance.

Time-dependent dielectric breakdown (TDDB) results from the electric field applied across the insulating oxide layer (the dielectric). TDDB is also known as oxide breakdown. The process starts with charges being trapped in the oxide. The material gradually deteriorates, the potential barrier it creates decreases, and conductive paths begin to form. Higher temperatures and stronger electric fields accelerate the process. The result is first an increase in leakage current and eventually a short circuit.

2.5.4 Whisker growth

Whisker growth [NASA/Tin whiskers, Fang et al., 2006] is a phenomenon in which filaments of metal build up and protrude from the surface. It is a slow gradual process of crystal formation due to a number of possible causes. The whiskers are thin but can grow to several millimetres. This is not a chip-level issue but affects the board and soldered connections mostly. A major concern is the potential for short circuits that the whiskers can cause. The problem has become a more relevant issue after certain materials (in particular lead) have been banned in electronics for environmental reasons. Whisker growth is more common in pure tin solder than tin-lead alloy solder.

2.5.5 Radiation dose and single event effects

Radiation effects on ICs have been mostly studied in relation to aviation and space flight. The radiation at high altitudes and in space is much stronger than at sea level under the protection of the atmosphere. See, for example, [Quinn, Graham, 2005]. Radiation can cause both gradually developing effects and sudden changes. The gradual changes are due to the total ionizing dose accumulating over time and are rather similar to the aging processes. The sudden effects, or single event effects, are caused by a hit from a single energetic particle.

The effect of the total ionizing dose (TID) is due to high energy radiation causing the formation of electron-hole pairs in the transistor's dielectric. This causes a build-up of charge that interferes with the control voltage of the gate. The threshold voltage changes, there will be an increase in leakage current, and the timing properties change. Eventually there will be a functional failure. Either the performance deteriorates so much that correct functioning is no longer achieved, or a there is a sudden failure as a result of, for example, dielectric breakdown and a short circuit.

Whereas TID takes time to develop, single event effects (SEE) caused by a single energetic particle are instant randomly occurring effects that include single event upset (SEU), single event transient (SET), single event latchup (SEL), single event gate rupture (SEGR), single event burn-out (SEB), and single event snap back (SES) [Wang, 2003, Sexton, 2003].

A single event upset is a soft error in the memory of the device. A soft error is one that does not cause permanent physical damage to the device. The energy

transferred by the radiation, an ionizing particle, can cause the state to change in a memory element. If sufficient charge is transferred to disturb the voltages of the transistors in a SRAM element, the state of memory may not recover to the correct value but may settle to the incorrect value. Alternatively, a voltage pulse caused by the particle can propagate through the circuit and alter the value of a memory element if the pulse arrives at a memory element at a suitable moment. The term single event functional interrupt (SEFI) is used if the upset affects the program memory and alters the functional configuration.

A single event transient is a voltage pulse propagating through the circuit. It can cause timing errors by shifting or introducing additional clock edges when the pulse is on a clock line. Incorrect value and operation can result if the pulse is on a data line. If the outcome is not transient, that is, if the pulse causes a change in memory, the effect is considered to be a SEU.

A single event latchup is a latchup with this specific initiating cause. Below is a more general discussion of a latchup. A single event snap back is similar to a latchup but the parasitic pnpn structure is not required. Both cause a high current that can destroy the transistor.

A single event gate rupture occurs when an energetic particle (a heavy ion) creates plasma of electron-hole pairs while there is a voltage applied across the dielectric. Somewhat different models of the chain of events leading to failure of the gate oxide are given in [Schwank et al., 2008] and [Sexton, 2003].

Single event burnout [Sexton, 2003] is an event observed in power bipolar (both holes and electrons act as carriers) transistors and MOSFETs, which control high voltages. A heavy ion hit triggers a condition known as second breakdown (a combination of excess voltage and current), in which the increased current heats up and destroys the transistor.

The different technologies, antifuse, flash, and SRAM, are compared in [Wang, 2003] and [iROC, 2004]. Antifuse is the most resistant to effects of radiation, while SRAM is the most susceptible. All three have the same CMOS technology for the gates but SRAM also has CMOS for the switches controlling the wiring and configuration.

The devices can be designed to be more resistant to radiation, or *radiation hardened*, through manufacturing techniques. The use of different materials and manufacturing techniques is considered in [Schwank et al., 2008]. The hardening is a part device fabrication, and thus using a COTS (commercial off the shelf) device limits the available selection. A specially designed and manufactured hardened version of a device would be significantly more expensive.

2.5.6 Current leakage

In theory, CMOS components use energy only when the states change. However, insulation is not perfect and while there is a voltage difference, there will be a small current. This unintentional current is referred to as leakage current. A small current passes through the transistors and also through the oxide of the gate. The

smaller the components and the thinner the insulation, the greater the current leakage will be. This additional energy consumption leads to increased heating, which in turn leads to other effects. For example, aging effects and timing are affected by temperature. Since the leakage happens in every transistor, even if the transistor is not actually used for implementing the logic, FPGAs with large numbers of transistors need to take it into account.

2.5.7 Latchup

CMOS technology uses paired NMOS (npn structure of N- and P-type semiconductors) and PMOS (pnp structure) transistors. Their proximity creates an unintentional parasitic pnpn structure which can enter a low impedance and high current state known as latchup [Morris, 2003]. When in latchup, the pnpn structure is a short circuit and current continues to flow through both transistors as long as the operating voltage is available. The heat generated by the current can destroy the device. Smaller scale and smaller components increase latchup risk while decreasing operating voltage reduces it.

Latchup requires an initiating event, which can be operating power fluctuation, radiation induced voltage pulse, or circuit switching noise. A trigger current in the transistors needs to be exceeded to initiate a latchup. To maintain it, the operating voltage must exceed holding voltage. If the holding voltage is not exceeded, a soft latchup can still occur. In a soft latchup, the initiating event triggers a latchup, which then recovers but the data or state of memory may be corrupted.

In silicon on insulator (SOI) technology the transistors are built on an insulating layer. Because there is no conductive path between the transistors, a parasitic pnpn structure is not formed and latchup cannot occur. This technology is becoming more common in space systems [Schwank et al., 2008].

2.5.8 Manufacturing technology size scale

Earlier it was mentioned that the manufacturing techniques and choice of materials can affect the radiation tolerance, or *hardness*, of the device. One issue to consider when choosing a device is the size scale of the technology. As the manufacturing technology of ICs has progressed, a major direction has been the decrease in size of the components and width of wiring on the chip. Currently technologies smaller than 100 nm are used. The “XX nm” technology is a naming convention, not an exact size of the components on the chip. Originally it referred to the size of the gate, that is, the distance between the source and drain. As the size decreases, the component’s robustness decreases. Radiation susceptibility increases, aging processes may hamper the operation sooner, and heating issues may become problems. Using an older more robust technology is therefore advisable if the capacity of the device is sufficient.

2.6 Timing, clock skew, and race condition

The flip-flops of a circuit are clocked to change value at specific times according to a clock signal, that is, at the rising or falling edge of a square wave clock signal. For the component to function properly, the input signal should remain steady for *set-up time* before the clock edge and *hold time* after the clock edge. The components do not react instantly but there is a small delay from the clock signal arriving to the output changing. If the input signal changes value, or is still in the process of changing too close to the clock edge, a timing error, such as an incorrect or metastable state, may result. The shorter the clock period, that is, the higher the operating frequency, the faster the components need to be to operate correctly.

Clock skew is the phenomenon of the clock signal arriving to components at different times. The cause is due to finite signal propagation speed and routing of the signals. The clock skew between components under ideal conditions can be determined from the layout and properties of the components and the wiring between them. However, the amount of clock skew is affected by factors such as temperature and the effects of aging on the circuit.

Timing margin is the increase in clock period from the minimum value necessary for correct operation. Its purpose is to avoid timing faults due to the variation in component properties and clock skew.

The term *race condition* refers to a situation in which the output of computation depends on the timing between signals, that is, the result is determined by the order in which the “racing” signals arrive. Poor timing can cause a gate to change state more than once during a cycle, which in turn can lead to, for example, metastability or the incorrect state propagating further.

2.7 Metastability

Metastability is a state in which the logical state is not properly defined. The circuit does not settle to either “1” or “0” but remains in between or oscillates between the values. It is caused by a timing error and persists for an indefinite time [Altera, 2009, Erickson, 2000].

Short-lived metastable states are more common than those of a long duration. The mean time between failures increases at an exponential rate to the recovery time. The recovery time should be taken into account when determining the clock frequency of the circuit. Even though the situation resolves itself, the error may have propagated and corrupted the state of the system. If the clock frequency is slower, there is more time for the metastability to resolve between clock cycles so that the value is correct by the time the next component reads it.

To avoid metastability, the timing of the circuit should be carefully analysed for worst case performance. External input signals to the circuit and signals from components that change value slowly should be synchronised. Flip-flops in sequence form a synchroniser and each reduces the probability of a metastable state propagating further.

The system may end up in an inconsistent state if two or more components use the metastable value and interpret it differently. This can lead to difficult to identify byzantine faults [Driscoll et al., 2003].

2.8 Parallel computing

One distinct difference between microprocessor-based systems running software and those using pure hardware such as ASIC or FPGA lies in parallel processing of the “program”. A microprocessor executes a single instruction of a single program per clock cycle. Of course, more than one processor can execute multiple programs which interact, but this complicates matters even more. An FPGA application executes all “instructions” on each clock cycle. Furthermore, there is separate hardware for each “instruction” of every program.

In parallel processing, different functions (for software this includes the operating system, driver software, and other functions of the platform) do not compete for the same processor time or memory resources. Timing, from the perspective of all instructions to be finished in time, is more reliable, but the order the instruction are executed is subject to the timing properties of the circuit. Different applications can be placed on different FPGA devices on different circuit boards to prevent harmful interaction and platform-related random-event common-cause failures.

2.9 System on a chip – SOC

The newer FPGA devices hold much more than just the configurable logic blocks. On the same chip, there can be a number of dedicated computational cores including complete microprocessors. In addition to the memory provided by the registers (flip-flops) in the logic blocks, separate blocks of memory are available. Signal processing units for analog-to-digital and digital-to-analog conversion are available along with more advanced hardware for communications. Actel’s flash-technology-based SmartFusion chip is an example of an advanced chip proposed for an NPP application [Lu et al., 2010].

As less and less hardware is needed on the circuit board in addition to the FPGA device itself, the designs are moving towards SOC solutions which are effectively complete computers on a single silicon chip.

2.10 FPAA – a glimpse of analog technology

Field Programmable *Analog* Array (FPAA) is in a way the counterpart of FPGA in analog technology. The philosophy is quite similar but instead of logic blocks, a FPAA chip contains configurable analog blocks (CAB). FPAA technology is younger than FPGA and is still in its infancy. See, for example, [Hall, 2004]

3. Application design and development

Even though the final product is a hardware component, the design and implementation of an FPGA-based system has strong similarities with software-based systems. A number of hardware description languages (HDL) or higher-level languages rather similar to software programming languages are typically used. This, together with the automated design tools, makes it very easy to define very complex functions. The steps of design use software tools to transform the design from one representation to another. The final output of the design process before configuring an actual device is a binary configuration code. FPGA device vendors have their own IDE (integrated development environment) tools that typically support at least the different design steps and simulation testing.

A characteristic of systems implementing complex functions or using a complex framework or platform, such as software and other programmable I&C, is the presence of errors of a deterministic nature (in addition to randomly occurring faults). The system does not fail randomly, but for a specific input the output is always incorrect.

Despite the similarities with software design, aspects of hardware design must be understood by the designer. The levels of abstraction that a software-based system offers are not present in FPGA design. There is no system level or an operating system behind which the hardware issues are hidden. On the other hand, the complexity caused by the shared computing environment and interaction with other programs is absent.

With both hardware and software aspects present, FPGA design and implementation can be characterised as “hardware implementation designed like software.” It is quite commonly agreed in the literature that, for applications with strict reliability and safety requirements, a design life cycle similar to software should be used.

In the following, we first describe a generic simplified design flow for an FPGA application. The each step and related issues are discussed in more detail. Finally, a number of design issues of particular interest are considered in more depth. The perspective of evaluating and reviewing an FPGA-based system, along with extensive lists of issues to consider, is given in [NRC, 2010b].

The following presentation focuses mainly on the design of the FPGA’s functions and the logic it implements. The connection of the chip onto the circuit board

along with the board's properties and connections are mostly ignored. Whereas functions are designed for a particular application, the FPGA device, circuit board, and related equipment are likely to be generic hardware combinations or platforms with few ties to a particular application field or industry.

A detailed beginner-level introduction to generic FPGA application development is given in [Smith, 2010]. More specific focus for nuclear power is provided in [Fink et al., 2010, Bobrek et al., 2009, Alvarado, Herrell, 2009].

3.1 Design life cycle, stages, and work flow

The first step is requirements specification. The requirements cover both logical functions and the physical hardware-related properties. Typically the requirements are drawn from the purpose of the system and the environment in which it is to operate. The result of requirements specification is usually a textual document with a list of requirements.

Based on the requirements, the chip type is selected, architectural design and detailed design of the application are performed. Architectural design considers the functions more as modules of abstract functions, while detailed design considers the specifics of implementation.

Code generation, or another form of inputting the design, implements the functions in a way similar to software code. The code must then be synthesised into a lower abstraction-level description of the functions and mapped onto the device hardware.

Verification and validation (V&V) is performed alongside the design. This includes various reviews, simulations, and tests. Reviews are used mainly in the earlier stages of design flow, where a higher level of abstraction is used for the descriptions, and simulations are used in the later stages. Static analyses can also be performed to, for example, analyse the code structure and style, timing performance, and correctness of the implemented logic.

The design process is iterative as it is likely that, at some point, a previous step will need to be revised or redone. For example, verification of the design step may fail or it may be discovered that some additional functions are needed or requirements need to be satisfied for correct and safe operation.

The naming of the design stages or phases is not universal and varies slightly in the literature. Some stages may be missing entirely or the definition may be different. See, for example, [Gaisler Research, 2002, NRC, 2010b]. The design stages and life cycle are depicted in Figures 5 and 6.

Stage	Artefacts	V&V
Requirements specification	Textual documents	Review
Architectural design Detailed design	Textual documents, high abstraction level descriptions, e.g., flow diagrams	Review (Behavioural simulation)
Behavioural description Design input	HDL code, schematic diagram	Behavioural simulation
Synthesis	Netlist	Behavioural (functional) Simulation
Place and Route	Configuration file	Functional simulation Timing analysis
Physical implementation	Programmed device	Testing Hardware in the loop simulation
System integration	Operational system	Testing

Figure 5. FPGA application design stages and related output (artefacts) and verification actions.

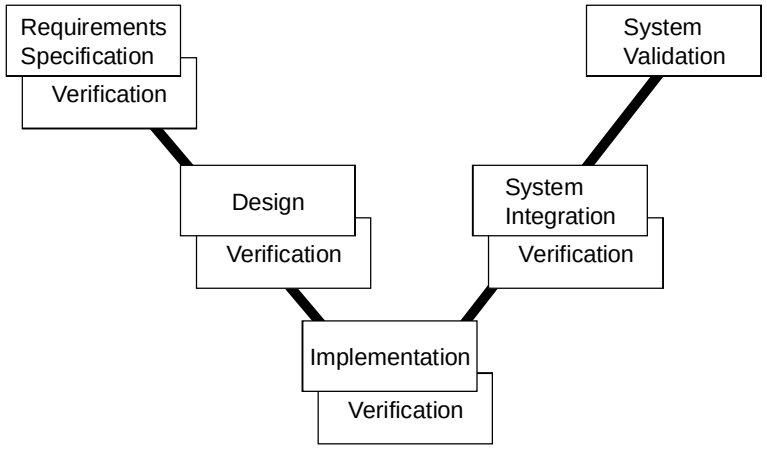

Figure 6. V-model of design life cycle.

3.2 Requirements specification

The purpose of requirements is to provide an objective and guidelines for the design process by specifying the properties of the final product. The requirements are based on the upper levels of system design and the role of the component or system to be designed. The core of the requirements is the specification of the functions of the device. The description includes the context and the environment in which the device operates. This means both the physical environment, for example, the temperature and radiation conditions, and the technological and "logical" environment, being the other systems it interacts with. In addition to the normal operating conditions, various extraordinary conditions need to be taken into account, such as the response to incorrect inputs, internal faults and fault tolerance, and changes in physical conditions.

The software aspects of the requirements mostly focus on the functional behaviour of the device, such as the purpose of operation and the reaction to certain inputs, but also things such as self monitoring, testability, and fault recovery. Requirements on hardware aspects may specify such things as connections and board size, response time and clock cycle, and mechanical acceleration (vibration) tolerance.

Requirements specification is one of the stages that often needs to be amended during later stages of design. Prior to knowing details of the design in question, and also the devices it interacts with, it is difficult to discern all the situations and conditions that may arise during operation and need to be prepared for.

3.2.1 ESL – electronic system level

An emerging trend is that an entire electronic system is described using a unified approach on a functional level, including all hardware, software, communications, and the functions they implement. This high-level description is usually created using a high abstraction level language, such as System C (essentially C++), System Verilog, or Matlab [Gassino, 2009]. Each part of the whole is represented by a component model that interacts with the other components. The behavioural models along with the communications form the overall system. This allows functional simulation of the system before any of the components are designed. The designers can test different allocations of functions and resources to components to find the best configuration. This electronic system level (ESL) description can then be used to derive requirements for the components. It may even be possible to generate the HDL code directly from the ESL description.

3.3 V&V: requirements

The output of the requirements specification consists typically of text documents with only a loose structure that is not suitable for automatic structured analysis and

verification. On the other hand, specifications drawn from an ESL description may even enable some level of behavioural simulation. For the most part, the verification is a series of inspections and reviews of the documents. The reviews should be performed by independent experts with no ties to the design team but sufficient skill and knowledge of the application area and design process. In particular, verifying that all necessary requirements are present may be difficult and requires in-depth understanding and expertise. Some structured review techniques are considered in [Lahtinen, 2012].

Traceability of the requirements is an issue of particular concern that spans the entire life cycle. The fulfilment of the requirements should be traceable through design stages to the final product. Throughout the design process, it should be possible to trace choices and decisions back to the requirements. In addition to all the requirements to be fulfilled, there should not be any requirements added in later stages but not included in the actual requirements specification. Hence, work practices and tools should be chosen to support traceability.

3.4 Architectural design

In architectural design, also referred to as high-level design, the functions are divided into smaller modules (blocks) and their interactions are specified. Several reliability considerations are relevant during architectural design. The design affects the possibilities to use, for example, redundancy and error correction, and also affects the overall robustness of the design. The use of TMR (triple modular redundancy) on the higher levels needs to be designed at this stage. Due to the limited number of input/output pins on the device, the internal state of the chip is not fully detectable during testing or operation and the choice of modularity affects the available data. In addition, the possibility to control the execution of specific functions depends on the division.

While designing the modularity, the use of pre-developed components and repeating structures should be considered. Some of the functions can be implemented using modules developed by third parties, or IP (intellectual property) cores. One approach that aims for designs of high reliability is to split the design into small simple functions that can be verified and use them as the basis for more complex functions [Kojima et al., 2010, Lach et al., 2005 and 2006].

3.5 Detailed design

Detailed design specifies the particulars of implementing the functions of the modules. Decisions are made about, for example, the use of memory, state machines, and redundancies. This stage does not necessarily exist as a separate stage in the design process, but it can be included in the earlier design stage and the following behavioural description.

Whereas the architectural design is for the most part independent of the hardware aspects, detailed design is more involved with the resources available on the device, timing properties, and the clock cycle to be used.

3.6 V&V: architectural and detailed design

Depending on the tools and format, different verification methods may be available. While the functions are not specified in detail, inspection and review are still likely to be the main methods.

3.7 Behavioural description and design entry

The design is implemented as hardware description language (HDL) code, a schematic diagram, or another description of the functions. (The word *implementation* can refer to a number of different things and stages of the design process.) This stage corresponds to writing program code for a software-based application. The most common method today is the use of a HDL; in particular, VHDL or Verilog languages are used.

The use of high-level languages can lead to complex and intractable descriptions which in turn hinder V&V activities. Therefore, strict rules should be applied to the use of the languages. Only clear and easily understandable structures should be used and the allowed features of the language should be restricted to a suitable subset of the language. The approach is the same as using a subset of a programming language, such as MISRA C (Motor Industry Software Reliability Association), in software-based systems. The ESA (European Space Agency) has specified guidelines for the use of VHDL [ESA, 1994].

3.7.1 Hardware description languages – HDL

Hardware description languages (HDL) are rather similar to software programming languages. They are used to describe the functions, not the physical components or connections. The abstraction level is on the RTL (register transfer level) and has little to do with the circuitry that eventually implements the functions. Like programming languages, HDLs use variables, parameters, and subroutines. However, the execution of the codes is different because software is executed one instruction at a time, possibly on two or more parallel threads, whereas the implementation of HDL code “executes” in a fully parallel environment with separate hardware for each “instruction”. Consequently there are differences in, for example, the operators of the languages. Code examples of VHDL and Verilog, along with recommendations for good coding style and practices, can be found in [NRC, 2010b].

The most important HDLs are VHDL [IEEE 1076-2008] and Verilog [IEEE 1364-2005]. Both are standards but neither is a formal language that would allow strict

analysis and verification of the code and functions. VHDL (very high speed integrated circuit HDL) is built on the ADA programming language and has much in common in structure and syntax. Verilog, on the other hand, was designed to have a syntax similar to C.

In [Gassino, 2009], a relation is drawn between HDLs and software programming languages. Despite the differences in parallel or sequential execution, the two implementations are more or less interchangeable. As a part of the development process, the HDL descriptions are simulated on computers, thus showing that software can do the same task as hardware implementation. On the other hand, software runs on a microprocessor, which is hardware, and can be implemented using a programmable device.

3.7.2 Higher-level entry methods

In addition to HDLs, more complex entry formats can be used in design. These include C, Matlab, and LabView. One approach available in the Xilinx Embedded Development Kit is to provide C code that is then executed on a soft processor core generated by the development kit. Other tools convert C or Matlab code into HDL code [Huffmire et al., 2008, MathWorks].

When the design is generated in a higher abstraction level language, creating and managing complex structures is simpler from the perspective of the designer. This speeds up the design process and should result in fewer errors in the design on the part of the designer. One disadvantage is the temptation to create more and unnecessarily complex designs. Another disadvantage is the need for one more tool and one more conversion between the designer's original intent and the final product, as more of the details of the design are left for the tools to decide. Additionally, the generated HDL code can be very difficult to read.

SystemVerilog aims for a higher level description of the system and hardware with verification in mind: "Hardware Description and Verification Language" (HDVL) [System Verilog]. Verilog and SystemVerilog have been merged in [IEC/IEEE 62530-2011]. SystemC is based on C++ and provides a set of classes for description and simulation of hardware at the system level [OSCI, IEEE 1666-2005].

Esterel is a programming language that allows generation of VHDL and Verilog code for hardware, in addition to C code for software [Boussinot, de Simone, 1991, Brandt, Schneider, 2008]. It is designed to support formal specification and verification. An IEEE standards development working group is developing Esterel v7 Language Reference (IEEE P1778).

3.7.3 Other entry methods

Hardware description languages are already rather high abstraction level descriptions. Simple logic can be designed using gate-level descriptions. Graphic EDA (electronic design automation) tools are available for entering the design using schematic or block diagrams. Schematic diagrams are lower abstraction level

descriptions of components, such as gates and lookup tables, while block diagrams use a higher level abstraction of functions.

3.8 Intellectual property – IP cores

The manufacturers of FPGA devices, as well as independent providers, offer predeveloped modules to be used as part of a design. This is similar to pre-developed software or library routines in software-based systems. The IP cores can be small routines implementing simple logic or larger entities for both frequently used functions and specialised tasks.

Delivery format and design level of the IP cores can vary. They can come as high-level description HDL code, netlists that are not device specific, or device-specific binary files. Factors affecting the format include issues such as IP security, from the developer's point of view, and the format is not always the best choice from the user's perspective. IP cores distributed as RTL level descriptions, such as HDL code, or as netlists are referred to as soft cores. IP cores distributed as device-specific (partial) configuration files are referred to as hard cores.

For safety-related applications with high reliability demands, the use of IP cores should be restricted to well-tested products from reliable sources. In [NRC, 2010b], it is recommended that in safety-critical systems, IP cores are avoided or additional verification of the core is performed. In addition to safety and reliability, there are security considerations related to the use of IP cores. Aspects of IP core security issues are considered in [Ziener, 2010].

Applications are typically developed using system platforms that include, in addition to development tools and hardware infrastructure, a set of predeveloped functions that serve as building blocks. Unlike third party IP cores, these are likely to be developed under strict requirements, if they are a part of a platform intended for NPP applications. Plans to use third-party certified SIL3 block libraries are underway.

3.9 V&V: design entry

In addition to inspection and review of the design, or diagrams and code, the format of the design allows simulation. Behavioural simulation is an important verification method at this stage. Details of the hardware are not available and therefore proper functional simulation is not yet possible. The test vectors, or the inputs and expected outputs, need to be designed carefully to cover the specifications and both normal and exceptional operating conditions. Additionally, faults and errors should be included in the tests, for example by use of fault injection methods. This allows wider test coverage than just the manipulation of input values. It also enables the testing of the effects of SEEs and other hardware failures and the correct behaviour of fault detection and recovery functions.

Both intermediate results (internal states of the design and inputs/outputs of functional blocks) and final outputs should be verified. Separate simulations can

be performed for the functional blocks or modules (or cores) to first verify their correct behaviour before examining the whole.

Only very simple designs can be tested with 100% coverage. As the size of the design increases, 100% testing of all possible behaviour quickly becomes too laborious and time consuming to be practical or even possible. Therefore, additional methods should be used to gain confidence in the correctness of the design. Such methods can be, for example, static analyses (dynamic analysis refers to simulation and static analysis to methods that do not activate or "run" the design with inputs), including formal methods (e.g. model checking), complexity metrics, and code reviews.

3.10 Implementation: logic synthesis and place and route

The description of the functions given as, for example, HDL code is first compiled into a netlist and then into a binary file with which the device can be configured.

A **netlist** is a device-independent description of the connections and gate structure of the logic. The generation of a netlist is usually referred to as *logic synthesis*. A typical format for netlists is electronic design interchange format (EDIF). Depending on the target technology, the netlist can also contain more complex functions, such as counters, in addition to the basic gates (AND, OR, etc.). Some tools generate both device-specific and device-independent descriptions.

Place and route operation maps the design onto a specific device architecture, that is, it creates the physical layout and is typically the phase referred to as *implementation*. This is usually done with the device manufacturer's proprietary software tools. The logic gates are mapped onto the hardware on the chip and the signals are routed between them. The place and route process is not deterministic, automatically providing the same best layout. Even the definition of "best" is ambiguous. The solution of the place and route problem is computationally hard and approximations and compromises are made. The higher the gate count, the harder the problem, and tools implementing non-deterministic methods, such as genetic algorithms can be used. The developer can also have a great influence on the outcome through configuration of the tool. Things such as signal speeds between areas of the chip, timing properties, availability of routing lines and gates, and power consumption all need to be considered. The output of place and route is the configuration file, which can be referred to as a place list, fuse map, or burn list.

3.11 V&V: logic synthesis and place and route

Verification is performed after both logic synthesis and place and route. As the design becomes closer to the final implementation, the details available in the simulations increase. After synthesis the properties of the technology can be included, and after place and route the details of the device and also the layout and routing information are available. An important aspect of correct functioning for parallel computing structures are the timing properties of the design, that is, prop-

agation delays and the arrival sequence of signals, and their correctness must be verified. See Timing Analysis below.

In addition to conformance to requirements, it should be carefully verified that the behaviour does not change as a result of transforming the design into new formats. Therefore, for simulation testing, the test vectors should be the same as in the previous steps. In addition to careless or non-systematic use of the tools, the tools themselves can be a source of changes and errors. Among other things, the configuration files of the tools must be documented and verified to be correct.

3.12 Implementation: configuring a chip and putting a programmed chip onto a board

The final step in the design process is the physical implementation. An FPGA device is configured using the configuration file produced by the place and route tool. The file is uploaded to the memory of the device (SRAM and flash) or a configuration tool is used to activate the desired connections in the device (antifuse). The FPGA device is connected to the circuit board, which is placed in a rack or computer cabinet to which power and signal wiring are connected. A single application can consist of multiple FPGA devices, possibly on multiple circuit boards.

3.13 V&V: physical implementation

Though the focus here is on the functions, the hardware itself must be verified. When the device is chosen, it is tested and verified to be suitable for the application (physical properties, memory technology, capacity, etc.). This includes the circuit boards and cabinets.

The configured device can be tested on its own to verify the functioning of the device. At this stage, the internal state of the device is unlikely to be fully observable (or there are delays in receiving the data through the output pins) and not all of the data of the test vectors of the previous stages can be verified. The inputs and outputs of the device should still be the same as in the earlier stages.

The correct functioning is also verified for the system with the FPGA device(s) on the circuit board(s). This testing includes the functioning of the other hardware on the circuit board.

The testing can be done using real hardware-generated input stimuli (intentionally generated or recorded from the actual operating environment), emulated hardware (which allows feedback structures), or input vectors from earlier stages. Real hardware-generated inputs are preferred to artificially generated inputs as the results then better correspond to the actual operating environment.

3.14 Other verification and validation issues

In the literature, it is quite commonly agreed that FPGA design should use a life cycle similar to software, and different variations have been proposed to take into account the special features of FPGAs. See, for example, [IEC 62566]. This means that the V&V activities should be performed alongside the design. The V&V plans themselves should also go through a V&V process. The coverage of testing (physical and simulated), the quality and detail of reviews, and the use of other methods (static analyses, formal methods) should be evaluated to justify the adequacy of the V&V plans.

The non-random nature of errors originating in design calls for additional considerations for “proven in use” arguments on reliability. If a particular operating situation has not been encountered in use, the fault has remained hidden. Thus, when using systems or components that have been proven in use in other applications, one should carefully consider the possible differences in operating conditions. For example, a piece of software used successfully on the Ariane 4 rocket failed when used on Ariane 5. It was tested as part of the Ariane 5 equipment, but the simulated flight trajectory was that of Ariane 4 and the particular error-inducing input was never encountered.

The definition of “100% testing” should refer to fully testing all input and internal state combinations. From a hardware perspective, “100%” could also mean exercising every gate to verify its correct operation, but this does not correspond with correct behaviour and reliable execution of the intended control logic.

3.14.1 Importance of independent V&V

The competence of the personnel performing the V&V is an issue to consider. In addition to technical expertise, their involvement with the design work needs to be considered. The first verification of the design can be made by the designers themselves but the final verification before acceptance should be performed by an independent party not directly involved with the design process. The designers can easily become blind to their own mistakes and no longer be able to question the assumptions made during the design process. In [Gaisler Research, 2002], a case is mentioned where there were internal verification teams independent of the designers, but where it still took outside experts to point out the errors (the comment “Oh my god!” by one of the outside experts gives an idea of the character of the errors discovered).

3.14.2 Formal methods

Formal verification can target the translation of the design between formats (equivalence checking) or the behaviour and functionality. While simulation tools check the equivalence or behaviour for the defined input test vectors, formal

methods verify them for all possible inputs. The methods apply a mathematical approach to generate a formal proof. The terminology used includes proof and assertion of properties. See, for example, the descriptions of a number of formal methods in C.2.4 of IEC 61508 [IEC 61508] part 7 and the references therein. Equivalence checking is quite commonly provided by the development tools along with the simulation tools. See, for example, [Xilinx/FAQ]. Formal verification of correct behaviour is the far more difficult task.

A large portion of the products of the design process are in text format in a natural language (English). This includes the requirements and descriptions of the higher-level systems and the context the system operates in. Formal methods that provide a mathematical proof of the correct behaviour cannot handle such material. A formal proof requires that the properties to be verified are also defined formally. At a later stage in the design process, the HDLs are standardised languages but not formal languages, and as such not entirely suitable for formal verification. There are tools for verification of HDL code, such as [IBM/RuleBase], but the properties to be asserted are still written separately using a different language instead of being an integral part of the system specification. Then there is the huge number of possible inputs and internal states that may present a problem through required computational resources and time. Note, too, that this analysis focuses on the correctness of the logic itself and excludes consideration of, for example, layout and timing.

Design tools and development environments are progressing towards providing more effective integrated verification methods, such as the Esterel language [Hammarberg, Nadjm-Tehrani, 2005]. Approaches that take into account the physical functionality and timing properties have also been developed. For example, the Property Specification Language, PSL [IEEE 1850-2010], can be used with multiple HDLs and it allows both simulation and formal verification of the design's properties. Design work may be progressing towards a situation where the synthesis is based on property and assertion statements instead of HDL code. Specifications of the properties to verify or assertion statements can be a source of errors themselves as they are typically written manually, as in the 20 000 lines of PSL code of [Druilhe et al., 2010].

An approach following the strategy of divide and conquer is proposed in [Lach et al., 2005 and 2006]. The approach involves verifying (100% testing or some other method) small units and using them as building blocks. Using reliable blocks gives confidence in the whole of the design but it can also be used to model the system based on the interaction of well-defined elements, which then would enable formal analysis. Practically all system platforms follow the strategy of developing a set of blocks or functions that are then used as building blocks for the applications, whether or not formal analysis is used for V&V.

A formal analysis does not have to go all the way and cover the entire system or be complete in all aspects. Verifying important parts of a design (repeating structures, critical functions) can increase confidence in the design and provide evidence to support a safety case. On the other hand, the use of formal methods provides a diverse approach to V&V. Even if the analysis is incomplete, not finding

errors can also be considered as evidence. On the other hand, finding errors allows them to be corrected. For the purposes of analysis, the design can be approximated with a simpler design that over-estimates the possible functionality. Then the absence of certain behaviour of the simplified design implies absence of that behaviour in the actual design. This approach, however, requires expert judgement and is somewhat lacking in transparency.

3.15 Tools

The design process is heavily dependent on software tools starting from the management of requirements to the testing of the device. The selection of these tools can have a significant impact on the success and outcome of the design process.

At the requirements specification stage, tools can be used to manage the requirements and to establish traceability from higher system-level requirements down to design decisions and the final design. Tools can also generate requirements from ESL descriptions. For V&V, tools supporting reviews can and should be used.

In the design stages, the main tools are HDL (or another description method) editors and translators to generate the netlist and place and route design. Tools to manage the schematic and flow diagrams of the first design stages can also be used. The simulation tools used for testing and verification of the designs also have a significant role. Simulation tools need to support the generation and management of test vectors, and enable their systematic execution and data collection, and the analysis of the results. In addition to simulation testing, static analyses can be performed, for example the evaluation of coverage metrics of simulation test vectors and HDL code style analysis. The timing properties need to be analysed by tools for all but the simplest designs.

Often the tools provided by the device vendors and third parties are not intended for the design of applications with strict safety requirements. Their focus is more on a fast, cost-efficient development process. With new device generations come new tools. Even though the tools are widely used by developers of various application fields, the tools are not rigorously tested and their development process is rarely suitable for high-reliability software products.

The tools do not simply translate the design into another format, that is, HDL is first synthesised to netlist and then placed and routed to a configuration file. The translation is not unique and the tools can perform optimisation of the design using various rules and algorithms according to some criteria. Even if the behaviour is superficially the same, there can be differences, in particular, under abnormal circumstances. For example, the optimisation of the layout done by the place and route tools can lead to some logic being duplicated due to routing resources and signal speeds while some logic can be entirely removed as redundant. It may be efficient to re-compute the same logic multiple times in different parts of the chip instead of establishing long wire connections, but in the case of a random error, the result may be different for different parts and the whole system may end up in

an internally inconsistent state. Similarly, removing redundant structures can be disastrous for the *intentional* redundancy (e.g. TMR) used for reliability reasons, such as the mitigation of effects of SEUs [Gaisler Research, 2002].

3.16 Emulated processor

One proposed approach to using FPGAs is to replace obsolete microprocessor based systems by emulating the processor with an FPGA. The original system concept and design are retained and the actual logic is not redesigned. This should ease the V&V effort. While the old microprocessors are no longer available, their functionality and software execution can nowadays be implemented in an FPGA. In France, an extensive modernisation project is underway using an emulated Motorola 6800 microprocessor [Bach, Tavolara, 2010]. This approach has the advantage of addressing more than one system because the same emulator can be applied to other systems using the same processor. The processor emulator is implemented as an IP core, which should allow it to be transferred to other FPGA devices. This type of replacement also needs to consider hardware aspects. The new technology does not necessarily have the same physical characteristics of, for example, radiation tolerance, aging, and electromagnetic compatibility.

3.17 Timing analysis

The elements of a circuit are clocked to operate at specific times, that is, they react to their input signals and change value. The timing is controlled using a clock signal “ticking” at a specified frequency. See “Timing, Clock Skew, and Race Condition” in the previous chapter. After place and route has specified the physical layout, timing analysis can be performed. The components and wiring produce delays to the signal propagation and the analysis is based on the physical properties of the device derived from the laws of physics and measured by testing. Given the layout and details of the design and properties of the device, the purpose is to verify that the logic functions correctly with respect to timing properties. The propagation and arrival of clock signals at different gates must be checked. All flip-flops on the chip must have sufficient set-up and hold times for changing data signals. Thus, in addition to clock and data signals, the relative arrival of the clock signal and data signals must also be considered. This also calls for checking the worst-case scenarios of uncertainty and performance degradation. However, timing analysis considers only signal propagation, not the values of the signals.

The timing properties can also be analysed without the details provided by place and route but instead based on the netlist. Delays of the components are taken into account, but as the layout information, being the location and wiring between the components, is lacking, the accuracy of the analysis is inferior.

Static timing analysis (STA) is an analytical analysis (not a simulation test limited to specific input cases). The tools for STA are very efficient for synchronous designs (see below) and the sufficiency of the timing margins of the entire design

can be verified. The results cover all possible signal paths and are independent of the signal values.

Timing analysis is also performed as part of the simulation of the circuit (dynamic analysis). In the simulation of the physical functioning of the device, the signal values are also considered and only cases included in the test vectors are covered.

3.18 Synchronous vs. asynchronous design

The timing of the flip-flops on the circuit is controlled with a square wave clock signal. It is possible to use multiple clock signals and either the rising or falling clock signal edge within an FPGA chip so that the flip-flops react at different times. It is even possible to use the output signal of another element as the clock signal. In a synchronous design, all components are clocked by the same edge of the same clock signal. Otherwise, the circuit is asynchronous.

Synchronous design is the simpler, more straightforward approach. On the other hand, asynchronous design has its advantages in some situations. It can facilitate better speed and lower power consumption or enable the solution of special problems [Erickson, 2000].

Existing design tools can easily verify worst-case timing properties (set up and hold time satisfaction) and performance requirements of synchronous designs. Complex asynchronous designs are an entirely different matter. Their analysis of worst-case conditions can be very difficult or even impossible in practice. Tools are mostly intended for synchronous designs, thus leaving the designer to manually verify asynchronous designs. Asynchronous designs are also more sensitive to changes in timing performance of components due to, for example, temperature and aging.

It is possible to have separate, non-interacting circuits with different clocks (in different clock domains) on the same chip. This is effectively a synchronous design but it would be simpler if the circuits were on different chips altogether.

3.19 Hardware vs. software aspects of design

It should be noted that the hardware aspects are strongly involved in the design process. Compared to software-based system design, timing with component response times and signal propagation delay is an additional concern. Similarly, the physical layout of the logic on the chip and parallel processing with dedicated hardware for each function are missing in software design. With FPGAs, the designer comes face to face with clock skew, SEE, and other hardware issues.

In comparison with conventional hardware design, the complexity of the functionality is likely to be significantly greater. The approach of using an HDL or design methods and tools with a higher level of abstraction also shifts the focus of the work. In the nuclear domain, this calls for new types of training and skills from the designers.

3.20 Standards

In the literature, the most often mentioned standards relevant to FPGA-based system development include:

- general safety standard IEC 61508
- upcoming IEC 62566
- IAEA standards
- the IEEE (7-4.3.2, 1012) software standards along with IEC 60880
- on the hardware side, the DO-254 standard for airborne electric hardware, which is mentioned along with the IEC hardware standards for NPPs.

IEC 62566 [IEC 62566] is expected to be published in 2012. Originally, it was to address the use of complex electronic components more widely, but it was then focused to "*Development of HDL-programmed integrated circuits for systems performing category A functions.*" The standard and its development have been presented in meetings related to FPGA use in NPPs and the standard is likely to become a relevant one.

The HDL standards are not usually mentioned in the literature. The focus of interest in a high-reliability context is more on limiting the features of the language that are used, as with ESA VHDL guidelines [ESA, 1994].

4. Risks and advantages

Most of the topics relevant to the risks related to the use of FPGAs are covered elsewhere in this report. For example, many of the risks related to design processes are covered in the chapter on design. Here, only some of the issues are reconsidered along with some risk mitigation approaches. This also applies to some extent to the advantages of using FPGAs, which are summarised here. Security issues, as in prevention of intentional malicious acts, and related risks are considered in this chapter and only briefly mentioned elsewhere.

Currently, there is not much experience of the use of FPGAs in NPP automation systems. This includes utilities, vendors, and authorities. For now, a project implementing FPGA technology therefore faces risks related to all phases of the project. The applicability and suitability for the intended purpose are not guaranteed, delays in the project are likely, and the approval of the authorities may be difficult to obtain.

One comparison between the risks of software and FPGA implementations can be found in [Kharchenko, Sklyar, 2008, pp. 64–69], in which the risks of FPGAs are considered lesser or at most equal.

The security issues related to preventing malicious changes to system behaviour are also a safety feature against *unintentional* changes. The integrity of the system needs to be considered in relation to, for example, maintenance and testing. In addition to technical approaches, administrative approaches should also be used.

In many sources, it is considered that the best choice for safety-critical systems such as those in nuclear power plants is antifuse technology. The non-volatility and non-reprogrammability prevent accidental and malicious changes to functions and antifuse has the best resistance to radiation. Flash technology has much the same properties when using a chip with security features and destroying the password after configuring the chip. When selecting the device, the complexity and range of additional features of the device should also be considered, as they may complicate the V&V through, for example, functions that need to be verified to be inactivated.

Some examples of the use of risk mitigation techniques, such as diversity, can be found in the descriptions of the current systems in Chapter 5.

4.1 Faults, tolerance, and mitigation

Four methods of protection against errors and faults include prevention, tolerance, removal, and forecasting. Fault prevention is focused on the development process, including hardware selection, equipment placement, and configuration of the system. A fault tolerant system is able to continue operation after a fault has occurred. Typical methods to achieve this are fault detection and restoring the system to a correct or safe state. Fault removal is part of the development process (V&V and corrective actions) and system maintenance during operation. Fault prediction considers the potential faults and their consequences and tries to find ways to avoid them or alleviate their effects. See, for example, [Ziener, 2010] section 1.2.4 and the references therein.

Methods for fault detection and repair are presented and discussed in [Ziener, 2010] section 2.3 and [Stott et al., 2010a] and their references provide an extensive list of literature on fault tolerance in relation to FPGA technology.

4.1.1 Triple modular redundancy

A commonly used approach to mitigate the effects of single event upsets (SEU) is to triplicate the computation and then use a majority vote among the outputs for the final result. Even if one of the computing units is affected by an SEU and produces an incorrect output, the final result is correct and it is much less likely that two units would be affected at the same time. This approach is known as triple modular redundancy (TMR) and is an example of voting among redundant inputs (the number of inputs and rules for voting can vary).

The technique can be used on a very small scale to triplicate individual flip-flops or gates, on a larger scale to triplicate the logic on a device or devices, or on plant-wide level to build entire redundant systems. From the perspective of FPGA-based system development, the redundancy of modules (of different sizes) and multiple devices on a circuit board are of most interest. For redundant devices, the term triple *device* redundancy is used.

In [Wang et al., 2010], it is argued that partitioning the design into smaller modules improves reliability. Having the whole design triplicated with a voter only at the end before output provides inferior reliability performance than partitioning the design into smaller modules, triplicating each of them, and voting for the intermediate results. This can be taken down to the level of individual gates and flip-flops.

The option of using TMR only on some of the gates is considered in [Xiaoxuan, Samudrala, 2009]. The gates are selected on the basis of how likely they are to produce an incorrect output as a result of an SEU. The probability depends on the type of the gate and the probability distributions of the inputs (the probability of the signal being “1” or “0”). With fewer redundant gates, the size of the design can be reduced compared to full TMR.

Some design tools allow automatic use of TMR. The designer can choose which parts are triplicated. Individual gates or flip-flops can be triplicated or larger entities can be selected.

One difference between combinatorial logic and sequential logic is the persistence of faults. If the memory is corrupted, the incorrect state will result in incorrect output for all future outputs. Resetting or synchronising the logic by feedback from the voted result can be used. If the modules are large, the errors may take a while to reach the output, which may allow other errors to appear and corrupt the other partitions.

The fault may affect the voting mechanism and thereby ruin the whole TMR scheme. However, SEU-resistant voters can be built using “tristate buffers”. Voter reliability is considered in [Carmichael, 2006]. Voters can also be triplicated. Figure 7 depicts different TMR schemes.

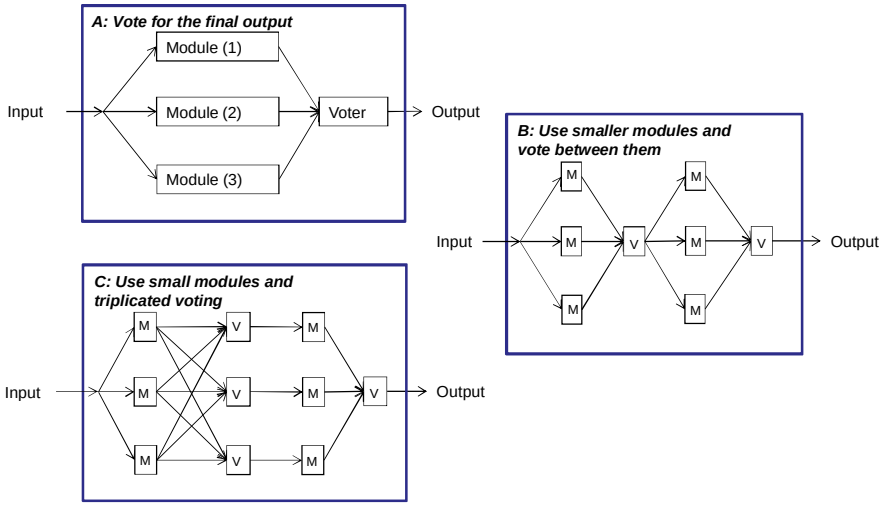

Figure 7. Triple modular redundancy schemes.

4.1.2 EDAC and hamming codes

Various error detection and correction (EDAC) schemes use additional redundancy information in the data to detect errors and correct them. In communication, the signal can be split into blocks, to which parity bits are added. The values of the parity bits are determined by the actual signal and an error occurring during transmission can be detected by the discrepancy it causes. Different schemes and numbers of parity bits offer different error detection and correction capacities, that is, the number of incorrect bits that can be detected or corrected.

In FPGAs, the EDAC schemes are used mostly for verification of the content of SRAM memory, which is the sensitive part of the device. For SRAM devices, this includes the configuration memory.

4.1.3 Diversity

The approach of using diverse implementations can be used against random faults in a way similar to TMR. However, its use is generally directed more as a measure against design and other systematic or common cause errors. Although making multiple independent versions based on the same specifications does not guarantee the independence of errors [Knight, Levenson, 1986], it may still help to reduce the risk of undetected errors. Using FPGA and software-based solutions may improve diversity due to the differences in design process and implementation, for example, description languages and cyclic versus parallel processing.

The diversity can be in the form of an independent development process, and even some of the lower-level requirements derived from higher-level system specifications could be different. More typical approaches include the use of diverse:

- personnel (design teams, organisations, managers, testers, etc.)
- HDLs (or other behavioural descriptions)
- tools (editors, synthesisers/compilers, simulation, analysis) and settings or configurations for tools
- function libraries and IP cores
- hardware (devices, device models, circuit boards, device manufacturers).

The number of FPGA device manufacturers is small (Altera and Xilinx are the largest; Actel seems to have an interest in the NPP applications), and therefore device diversity may be limited to different models, which may be developed by the same design teams and share parts of the design.

Different diversity approaches are discussed in, for example, Chapter 6 of [NRC, 2010a]. On a higher level, additional diversity can be achieved using different technologies, such as FPGA, microprocessor/software, and analog. Beyond the level of implementing control logic, diversity can include separate process instrumentation (measurement and actuators), but that is beyond the topics of this report.

4.2 Advantages of FPGA technology

One driver behind the interest in FPGAs is the replacement of old systems based on analog technology that are becoming obsolete. They are difficult to maintain and the situation is becoming worse with regard to component availability. The flexibility of the technology in different applications is limited. In addition, analog components suffer from drift of characteristics that affect functions.

The FPGA design tools provide extensive functionality to support the design process. High abstraction level languages and block libraries and cores provide a good foundation on which to build individual applications. The testing tools help with verification at every step of the design.

The primary advantages of FPGA technology when compared to microprocessor-based solutions in NPP applications are considered to be the reduction of complexity and the fewer risks of the technology becoming obsolescent [Fink et al., 2010]. Performance characteristics related to processing speed and response times are seen as an advantage [Bach, Tavolara, 2010]. In addition, the flexibility of FPGAs makes them applicable for various purposes.

Application-specific integrated circuits (ASIC) are similar to FPGAs in the sense that the function is pure hardware without additional layers of platform software (operating system, drivers, etc.). Unlike FPGAs, the functions of ASICs are fixed and determined at production. Only the necessary circuitry is placed on the chip. Because the manufacturing process cannot be used to mass produce “blank” devices that are then configured for the applications, the non-recurring engineering costs (in particular, setting up the manufacturing process) are huge compared to FPGAs. Typically, ASIC is a good choice if the production quantities are in the millions of devices, which clearly is not the case in the nuclear industry. On the other hand, ASICs provide better processing speed and power consumption properties.

Dedicated separate hardware for all functions provides the advantages of computational efficiency, but from the reliability point of view, a more important aspect is the separation of functions. There is no need for resource allocation such as memory, processor time, or data transfer on a bus. This reduces the risk of functions interfering with each other or with the operating system or other platform functions.

A rather strong opinion presented in [Salaün et al., 2009] is that standardised configurable hardware components provide the advantages but not the problems of both conventional hardware and microprocessor technology.

The complexity of systems implemented using conventional hardware, a conventional microprocessor, and different approaches to using FPGA are compared in [Fink et al., 2010]. Complexity in this case is defined as the “presence of software and hardware components that are not directly responsible for performing the primary I&C functions but cannot be shown to be independent from the primary functions, and thus complicate the design and its assessment for reliability and safety justification.” Conventional hardware is considered the least complex as there are only the necessary components and functions. Conventional microprocessor implementation is the most complex. FPGA falls in between those two: architecture using only simple hardware logic is close to conventional hardware, an embedded microprocessor without an operating system in between, and an embedded microprocessor running an operating system close to conventional microprocessor. The capabilities of the technologies correspond to the complexity, with increased complexity offering increased capability to implement more varied functions and alternative solutions.

Security issues are also seen to be less with FPGAs than with microprocessor-based systems. Having access or the possibility to alter the functions is more difficult. With antifuse, the functions cannot be changed at all without physically replacing the device. Hence, it is very difficult to force FPGAs to execute malicious software (viruses, worms) alongside the intended functions.

Though not an advantage of the technology as such, simply by being different from microprocessors, FPGAs provide an opportunity for a diverse implementation.

4.3 Experience from other fields

The aerospace industry is an area where FPGAs have been used in high-reliability applications. Of particular concern for both high altitude and space applications are the effects of radiation, as without the shielding effect of the atmosphere the radiation environment is harsh. In addition, the hardware reliability of space applications has a special consideration for maintenance. It may be impossible to replace a malfunctioning component even if you know exactly where it is. In [Gaisler Research, 2002], twelve problems are presented as case examples, and ways to avoid them are suggested. The focus is on the most common problems that can be avoided with careful design. Gaisler Research [Gaisler] has produced other studies dealing with FPGA use in space applications. In addition, [klabs.org] contains a large collection of material related to FPGAs and ASICs in space applications.

Views presented by Actel in an e-magazine article [Mason, O'Neill, 2005] and a white paper [Actel, 2003] consider reliability issues in automotive and space applications. The focus is mostly on hardware as Actel is an FPGA manufacturer.

4.4 Security

Security issues can be broadly divided into two categories: information going out and information coming in. Outbound information in the form of intellectual property (IP theft) is mostly the concern of the vendor and designer. On the other hand, knowledge of the details of the system can also pose a threat as they may facilitate an attack on the system. Security issues related to inbound information can be, for example, in the form of inserted harmful functions in the system. Countermeasures include technical and administrative approaches. For example, the data can be encrypted and verified when the chip is configured and only certain persons can have access to the device or data.

4.4.1 Problem areas

According to [Huffmire et al., 2008], the main security problems of reconfigurable hardware are: design tool subversion, composition, trusted foundries, and bitstream protection.

The design tools can insert malicious features into the design. Several different tools could be used at various stages of the design flow. The inserted functions can give unauthorised access to the device, cause incorrect behaviour, or even destroy the device by creating a short circuit.

The composition problem results from the final design being a collection of interacting cores. There are a number of different functions that can influence each other on the device. The IP cores also need to be accounted for, as they may contain intentional or unintentional harmful features. One core could monitor or alter another and be in contact, possibly over a network, with an entirely different system.

The trusted foundry problem is a lesser problem for FPGAs than for ASICs. When the device is not configured by the device manufacturer at the foundry, IP theft can be avoided at that stage. However, the design could be read and stolen from the device. On the other hand, malicious features can be inserted onto the device during manufacture or transport to the customer.

Bitstream protection is of concern in both IP security and having the device configured with the correct bitstream. In particular, this is a problem of SRAM technology because the configuration is loaded from a separate memory device or possibly over a network. Therefore, the bitstream is easier to access and change.

4.4.2 Managing security problems

As a solution to the above security problems, life-cycle management and secure architecture are suggested.

From a tool perspective, **life-cycle management** means verifying the trustworthiness of all tools. FPGA technology progresses quite fast and new tools appear frequently. The extent to which the tools are tested and the amount of accumulated proven-in-use experience should be taken into account when selecting tools. Each version of the tools should be evaluated, because new versions may introduce new flaws. Hardware life-cycle considerations include production and delivery of the device. The device could be tampered with already at the foundry or during transportation. These are in addition to installation, configuration, operation, and maintenance.

Secure architecture includes features such as memory protection, spatial isolation tags, and secure communication. Memory protection enforces proper sharing of memory resources between cores, so that they do not interact unintentionally. Spatial isolation physically separates the cores onto different areas of the chip. This way, it is easier to verify that they do not interfere with each other's operation. Tags can be used as metadata attached to pieces of information to track them. Secure communication via memory can be covered by memory protection. Direct communication between cores should be analysed and checked to verify that only specific connections are possible. Communication over a shared bus has greater security issues.

Encrypting the bitstream or using a **fingerprint** or **watermark** helps to protect against IP theft and also prevents the device from being configured with an incorrect bitstream. The device can also have built-in self-test (BIST) features to check the correctness of the configuration and the integrity of the data.

Antifuse devices can have a security fuse/antifuse, which is activated after configuration is complete and prevents further modification of the configuration. Verifi-

cation of the configuration (in any technology) can be implemented so that a configuration is input and the device simply responds whether the configuration matches that on the device. In this way, the configuration cannot be read from the device (readback) but can be verified.

Administrative measures can be used to prevent unauthorised personnel from having access to the device. Access to and possession of the tools needed to configure the device can also be controlled [NRC, 2009b, Bach, Tavolara, 2010].

4.4.3 Attack methods

In [Drimer, 2008] Chapter 3, a number of different methods of attacking an FPGA device and its design are covered. The focus is on IP security. The following is an abstract of the topics covered.

Bitstream reverse engineering attempts to reconstruct the higher-level description (a functionally equivalent description) of the design based on the configuration bitstream. It is, in a way, a reversal of the place and route and netlist synthesis.

Counterfeits and clones are easy to produce if the configuration bitstream for a particular device is available. **Overbuilding** refers to a contracted manufacturer producing more devices than ordered and then selling them on their own and thus avoiding development costs.

Readback retrieves the state and configuration of an FPGA device in operation. The entire state might not be accessible, but it is still very useful for verification and debugging. An attacker can use readback to copy the device, alter it, or reverse engineer it.

Side-channel attacks try to gather information from the measurable external phenomena and deduce something about the inner workings of the device. Power consumption analysis tracks the electric power usage of the device. Electromagnetic emanation analysis measures the changes in electromagnetic fields cause by current changes inside the device. Timing analysis measures changes in execution times. For example, discovering a password that is input one character at a time is possible when the measurement results differ between inputting a correct character and an incorrect character.

Invasive attacks physically alter the device. The packaging is removed and the exposed chip is analysed. For example, an aging mechanism also leaves imprints on volatile memory, providing clues to what the configuration was.

Brute force tries to overcome encryption by trying all key values. It can also try to force a system into an error state through unexpected input values.

Crippling attacks reconfigure the device with malicious programming. Even in the presence of encryption, it may be possible to at least cause the device to stop functioning. **Fault injection** uses methods such as input clock frequency and voltage variations and electromagnetic fields to alter the state of the system or configuration. **Ionising radiation** can be used in a similar way.

Relay attacks use communication between the target of the attack and an authorised party by establishing a connection between them. For example, commu-

nication between a magnetic key card and a lock could be relayed over a distance. **Replay attacks** record communication with the device and attempt to overcome security by sending the same, or possibly modified, transmission or parts of it again.

Social engineering targets personnel to overcome security measures. Phone calls and emails that trick the receiver into giving away sensitive information, as well as bribery and stealing laptops and mobile phones, fall into this category.

4.4.4 Defences

Security measures fall into three categories: social, active, and reactive [Drimer, 2008]. Social methods are laws and agreements and associated penalties. Active methods prevent attacks using, for example, encryption and locked doors. Reactive methods expose theft, or security breach using, for example, watermarked files.

Encryption of the bitstream is more efficient if the decryption is done inside the FPGA chip. This, however, requires a slightly more advanced and complex device. The safety of the key is then another matter, but if there is no intent to reconfigure the device the key can be destroyed.

Design theft deterrent methods check that the FPGA device is installed onto an authentic circuit board.

Watermarking and **fingerprinting** the configuration bitstream allow identification of the source of the file. Additional information is added to the configuration file so that the functions are not altered, but it is very unlikely that another independent design process would come up with the exact same file.

Physically unclonable functions are based on uncontrollable but measurable variation in, for example, the manufacturing process. A device can be identified by, for example, the details of the timing of signal lines that are, by design, supposed to be identical, or by the initial state to which RAM memory settles on power up. The identification is “physically imprinted” onto the device.

5. Current systems

The following is based on publicly available literature, mostly conference papers. The goal has been to gather information about both currently available systems and those that are still under development. Some of the systems are developed only for research purposes to evaluate suitability and build expertise.

In Figure 8, some components of the Radiy platform are presented to give an idea of the physical structure of the system: the cabinet houses the chassis into which the circuit board containing the FPGA device is inserted.

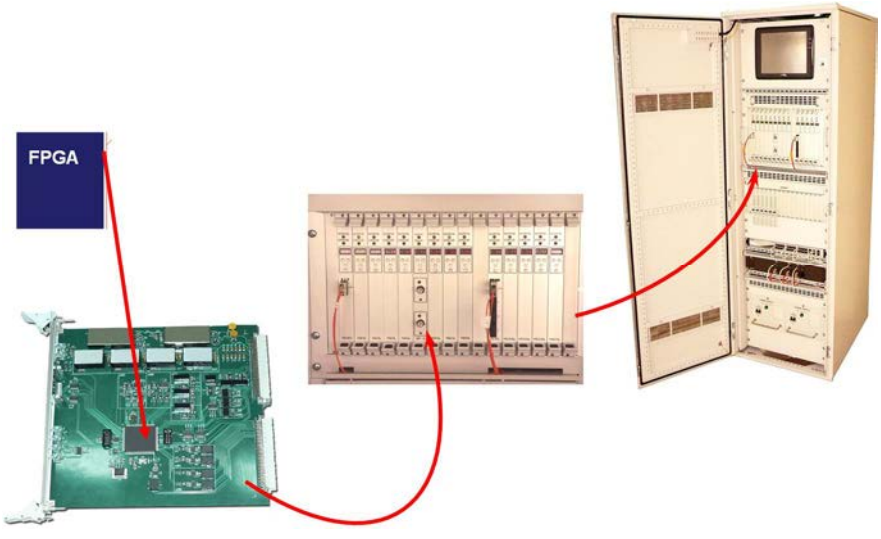

Figure 8. Hardware of the Radiy platform. Image source: Sklyar, V., Case of Application: Implementation of a Reactor Trip Function, 4th Workshop on the Applications of FPGA in NPPs, France, 2011.

5.1 CANDU

An experimental system has been developed and tested using a hardware-in-the-loop simulation [She, Jiang, 2009]. The system implements the logic for shutdown system no. 1 (SDS1) for the CANadian Deuterium Uranium (CANDU) reactor. The system is an example of replacing existing technology with FPGAs. The purpose of this study was to show the feasibility of FPGA technology and the study was performed by academic researchers.

In another project [Xing et al., 2010], AECL (Atomic Energy of Canada Ltd.) and RPC (Research and Production Company) Radiy are working to define an application development process based on Radiy's FPGA-based platform. A pilot project implementing CANDU shutdown system logic is being carried out. The aim is to have an FPGA development process that meets regulatory requirements and IEC standards for safety-critical system development.

5.2 Lungmen

This, still in design, FPGA-based reactor protection system (RPS) is described in detail in [Lu et al., 2010]. The system is designed for the Lungmen NPP, which is still under construction. The system is to be tested using a full-scope engineering simulator with detailed models of the plant systems and reactor core.

The RPS is a part of the safety system logic and control (SSLC) which has four separate divisions. RPS has a role in both safety and power generation. The tasks for safety features include automatic scram under specified conditions and enabling manual scram. For power generation, the tasks include tripping certain reactor internal pumps under specific conditions and providing signals for alert and indication purposes.

A flash-based Actel SmartFusion chip is used. It is an advanced chip with an internal memory, embedded microcontroller unit, configurable signal processing UARTs (universal asynchronous receiver/transmitter), analog-to-digital and digital-to-analog converters, and internal monitoring.

The RPS has four circuit boards, one in each of the four divisions. On each board, there are two sets of hardware logic for all safety functions. Diversity of the logic is achieved by using two different logic input formats: VHDL, Verilog, or schematic capture.

5.3 Wolf Creek

In 2009, the U.S. NRC granted a licence for an FPGA-based system at the Wolf Creek Generating Station [Dittman, 2010, NRC, 2009b]. The newly licensed system replaces the existing MSFIS (main steam and feedwater isolation system). The system is based on CS Innovations' ALS (advanced logic system) and flash technology based FPGAs.

The primary tasks of the MSFIS are to receive signals from the SSPS (solid state protection system) or manual signals, and in response to send signals to individual valves. It does not receive measurement signals from sensors or determine actions. The simplicity of the system played a significant role in gaining the licence. The fact that the system replaces an existing system also contributed to licensing.

Each circuit board has an FPGA chip with two diverse logic cores [NRC, 2009b, p. 16] that work in parallel and were synthesised using different directives for the synthesiser. This allows for error detection and self check. If the results differ, the board sets output to failsafe values and halts operation.

Because flash technology is reprogrammable, precautions needed to be taken to prevent inadvertent or malicious changes in the functions. The maintenance computer and the platform are both designed to prevent field programming of the system. Plant personnel are prohibited from on-site possession of the equipment necessary for reprogramming, and only the vendor has the authority to alter the operational behaviour of the system. The FPGA firmware also verifies that the configuration is valid and generates an alarm if it is not.

5.4 RPC Radiy

The Ukrainian Research and Production Company (RPC) Radiy has developed an FPGA-based platform for implementing safety-critical I&C applications [Kharchenko, Sklyar, 2008, Bakhmach et al., 2010a, 2010b, Yastrebenetsky et al., 2009, Bakhmach et al., 2009]. It can be scaled and tailored to fit different needs. Systems have been installed in several Ukrainian and Bulgarian NPPs. More than 50 safety systems have been installed since 2003.

The system consists of higher and lower levels. The higher level consists of computer cabinets, workstations and their software. The lower level consists of standardised cabinets and modules along with the electronic designs for the FPGAs. Essentially, the division is into the high-level man-machine interface and diagnostics and low-level signal processing and control algorithms.

Based on the platform, the following types of systems have been implemented: reactor trip system (RTS), reactor power control and limitation system (RPCLS), engineering safety feature actuation system (ESFAS), power equipment for rods control system (PERCS), regulation, and monitoring, control, and protection system for research reactors (RMCPS). Time schedules for design, implementation, and commissioning of an FPGA-based ESFAS are presented in [Bakhmach et al., 2010a].

The qualification approach divides the FPGA component into two entities. Physically, the chip is electronic hardware and should be qualified accordingly. The electronic design, such as HDL code, should be qualified against functional requirements. The design process follows a V-shape life cycle adapted from software by including FPGA-specific stages as described in the upcoming standard IEC 62556.

RPC Radiy is co-operating with AECL in a pilot project to develop an FPGA development process (see CANDU above).

5.4.1 IERICS mission on FPGA-based digital I&C platform and systems

An independent engineering review of instrumentation and control systems (IERICS) [IAEA, 2011] mission was carried out concerning the FPGA-based systems produced by RPC Radiy [IAEA, 2010]. The purpose of IERICS missions is to peer review I&C systems (from those still in development to deployed systems in operation). The review teams consist of invited technical experts with different backgrounds. This was the second IERICS mission. The review was conducted during a week-long visit to Ukraine in December 2010. In addition, there was a preparatory meeting and a final follow-up/closing meeting.

5.5 Rolls-Royce and Electricité de France

Rolls-Royce has been contracted by Electricité de France (EDF) to carry out an extensive modernisation project of the 900 MWe units. The rod control systems (RCS) and reactor in-core measurement systems (RIC) of all 34 of the 900 MWe units owned and operated by EDF are to be replaced. The project has reached the deployment phase [Bach, Tavolara, 2010].

The current systems are from the 1970s and are considered difficult to maintain and to cause unnecessary loss of production due to faults. The replacement of the obsolete technology with FPGA technology is being carried out by the instrumentation and control department of Rolls-Royce. The new systems are to be placed in the same cabinets that the old systems occupied. The replacement takes place during the outages during the ten-yearly inspections of the plants. These outages are longer than normal and offer a suitable window of opportunity for larger maintenance and modernisation operations. The project started in 2005 with requirements specifications. In 2010, the refurbishment of the first unit was finished and the schedule is to have all done by 2020.

The main tasks of the RCS are to generate control signals to move the control rods, verify that the actual rod position corresponds to the controls, interface with the human system interface to allow manual control of rods, and interface with the PLC-based control and diagnostic unit.

The chosen technology is flash based. Thus, the devices are reprogrammable but changes require special tools and are possible only at Rolls Royce premises. The main reasons for selecting FPGA technology were the fast response time and the perceived avoidance of future problems with obsolescence.

5.6 Toshiba

Toshiba has developed a number of FPGA based systems intended for safety-related functions in the boiling water reactor (BWR) and advanced boiling water reactor (ABWR) designs [Kojima et al., 2010, Miyazaki et al., 2009]. The systems include monitoring (e.g. power range neutron monitor) and actuation signal generation (reactor trip and isolation system).

The technology used is antifuse from Actel Corporation, and the decision on its use was based on non-rewritability and reliability considerations. The design life cycle, V&V, and qualification are based on IEEE software standards. VHDL was used.

The developed systems are: power range neutron monitor (PRM) for BWR and power range neutron monitor (PRNM), startup range neutron monitor (SRNM), and reactor trip and isolation system (RTIS) for ABWR. The design process was developed and tested with the PRM and then used for the PRNM, SRNM, and RTIS.

The design method is based on a library of functional elements (FE) that perform simple functions (e.g. addition, comparison, and data communication) and that can be verified using full pattern testing. The FPGA is programmed using these as building blocks. The V&V process then considers the connections between the FEs. Evaluation is based on the toggle coverage ratio, which is a metric that measures the possible different input/output combinations (ones and zeroes) of the FEs that are activated in the testing relative to all possible combinations.

6. Summary

This report has tried to give the reader a basic understanding of FPGA technology in the context of nuclear power plant I&C systems with safety requirements. The introduction started by describing the hardware, then the application design workflow and related safety issues were considered, and finally examples of the currently existing systems were given.

FPGAs are programmable electronic devices with capacity for very complex computations. The devices can be mass produced as blank devices without fixed functions and then programmed (configured) to execute desired functions. FPGAs are used widely in various application areas including safety and reliability-critical areas such as the aviation and automotive industries.

The work on bringing FPGAs into wider use in nuclear power plants is ongoing. New systems and platforms are under development while the first applications are already in use. This trend is driven by the perceived advantages of FPGAs over software and microprocessor-based systems and the need to replaces obsolete systems based on analog technology. FPGAs are seen to have superior performance characteristics while being simpler and easier to license than software-based systems.

FPGAs are integrated circuits based on semiconductor technology and have much the same reliability issues as other similar hardware. These include aging and performance deterioration of the device, random errors due to, for example, radiation and power fluctuations, and heating due to current leakage. Care needs to be taken to choose a suitable device with sufficient reliability characteristics. Typically, simpler devices of smaller capacity can achieve greater reliability due to more robust technology, while also helping to maintain lower application complexity through lack of excess built-in functions. Special devices, such as those using silicon on insulator technology, should also be considered.

Application development based on FPGAs has much in common with software development. It is highly dependent on software tools, which can be sources of errors themselves but also create an illusion of simplicity by allowing easy construction of complex behaviour. The initial stages of development are also rather similar in requirements specification, architectural design, and use of hardware description languages similar to programming languages. The final product, however, is a hardware device without an operating system that would execute a pro-

gram one step at a time. Instead, all operations run in parallel with dedicated hardware for each function. Thus, the last stages of application design have a much more hardware-oriented character. The designer needs to take into account things such as the layout of transistors on the silicon chip, signal propagation and timing properties, and power consumption.

There are two important forums of discussion and dissemination of experience on FPGAs in the nuclear domain. The *Workshop on the Use of Field Programmable Gate Arrays in Nuclear Power Plants* is an annual, somewhat informal workshop held since 2008. It has been organised twice in France (2008, 2011) and once each in Ukraine (2009) and Canada (2010). In 2012, the workshop is planned to be held in China. It has been organised in cooperation between IAEA and the industry, including utilities and system providers. In 2011, there were more than 80 participants representing the industry, authorities, and research organisations. The *International Conference on Nuclear Plant Instrumentation, Control, and Human-Machine Interface Technologies (NPIC & HMIT)* is an event organised somewhat irregularly with the 6th and 7th conferences in 2009 and 2010 and the 8th scheduled for July 2012. This is a large event, with FPGAs representing only a small fraction of the topics. Still, in 2010, there were two sessions devoted to FPGAs and two sessions are planned for 2012.

References

Actel 2003. Reliability Considerations for Automotive FPGAs. White Paper. Available at: http://www.actel.com/documents/AutoWP.pdf (accessed: 5 December, 2011).

Altera 2009. Understanding Metastability in FPGAs. Available at: http://www.altera.com/literature/wp/wp-01082-quartus-ii-metastability.pdf (accessed: 14 September, 2011).

Alvarado, R., Herrell, D. 2009. Approach to designing FPGA-based digital I&C systems for nuclear applications. NPIC&HMIT 2009, 5–9 April, Knoxville, Tennessee.

Bach, J., Tavolara, I. 2010. Use of FPGA Technology in Implementation of the Logic of the Modernized Rod Control System (RCS) of the 900 MW EDF Fleet. In: NPIC&HMIT 2010, Las Vegas, Nevada, pp. 1326–1336.

Bakhmach, I., Kharchenko, V., Siora, A., Sklyar, V., Andrashov, A. 2010a. Experience of I&C Systems Modernization Using FPGA Technology. NPIC&HMIT 2010, 7–11 November, Las Vegas, Nevada, pp. 1345–1352.

Bakhmach, I., Kharchenko, V., Siora, A., Sklyar, V., Tokarev, V. 2010b. Design and qualification of I&C Systems on the Basis of FPGA Technologies. NPIC&HMIT 2010, 7–11 November, Las Vegas, Nevada, pp. 916–924.

Bakhmach, I., Siora, A., Tokarev, V., Reshetitsky, S., Bezsalyi, V. 2009. Implementation principles of FPGA-based ESFAS for Kozloduy NPP. In: NPIC&HMIT 2009, 5–9 April, Knoxville, Tennessee.

Bernstein, J.B., Gurfinkel, M., Li, X., Walters, J., Shapira, Y., Talmor, M. 2006. Electronic circuit reliability modeling. Microelectronics and Reliability, Vol. 46, No. 12, pp. 1957–1979.

Bobrek, M., Wood, R.T., Bouldin, D., Waterman, M.E. 2009. FPGA design practices for I&C in nuclear power plants. In: NPIC&HMIT 2009, 5–9 April, Knoxville, Tennessee.

Boussinot, F., de Simone, R. 1991. The Esterel Language. Proceedings of the IEEE, Vol. 79, Issue 9, pp. 1293–1304.

Brandt, J., Schneider, K. 2008. How different are Esterel and SystemC. In: Villar, E., (ed.). Embedded Systems Specification and Design Languages, Selected contributions from FDL'07, Lecture Notes in Electrical Engineering, Vol. 10, DOI: 10.1007/978-1-4020-8297-9, pp. 3–3, 2008. Available at: http://www.springerlink.com/content/978-1-4020-8297-9 or

http://rsg.informatik.uni-kl.de/publications/datarsg/BrSc07b.pdf (accessed: 20 September, 2011).

Carmichael, C. 2006. Triple module redundancy design techniques for Virtex FPGAs. Available at: http://www.xilinx.com/support/documentation/application_notes/xapp197.pdf (accessed: 1 December, 2011).

Dittman, B.F. 2010. Licensing field-programmable gate arrays in safety systems. In: NPIC&HMIT 2010, 7–11 November, Las Vegas, Nevada, pp. 966–976.

Drimer, S. 2008. Volatile FPGA design security – a survey. Computer Laboratory, University of Cambridge, 2008. Available at: http://www.cl.cam.ac.uk/~sd410/papers/fpga_security.pdf (accessed: 5 December, 2011).

Driscoll, K., Hall, B., Sivencrona, H., Zumsteg, P. 2003. Byzantine fault tolerance, from theory to reality. In: 22nd International Conference on Computer Safety, Reliability, and Security, SAFECOMP 2003, Edinburgh, UK. Published in Vol. 2788 of Lecture Notes in Computer Science, Springer, pp. 235–248.

Druilhe, A., Daumas, F., Nguyen, T. 2010. Formal verification of an FPGA emulation of the Motorola 6800 microprocessor. In: NPIC&HMIT 2010, 5–9 April, Las Vegas, Nevada, pp. 1316–1325.

EPRI 2009. Guidelines on the use of field programmable gate arrays in nuclear power plant I&C systems. Electric Power Research Institute. Product ID: 1019181, December, 2009. Available at: http://my.epri.com/portal/server.pt?Abstract_id=000000000001019181 (accessed: 7 December, 2011).

EPRI 2011. Recommended approaches and design criteria for application of field programmable gate arrays in nuclear power plant instrumentation and control systems. Electric Power Research Institute. Product ID: 1022983, June 2011. Available at: http://my.epri.com/portal/server.pt?Abstract_id=000000000001022983 (accessed: 7 December, 2011).

Erickson, K. 2000. Asynchronous FPGA risks. In: Proceedings of the 4th Military and Aerospace Applications of Programmable Devices and Technologies International Conference (MAPLD '00), Laurel, Md, USA, 2000. Available at: http://klabs.org/richcontent/MAPLDCon00/Papers/Session_A/A5_Erickson_P.pdf (accessed: 14 September, 2011).

ESA 1994. VHDL Modelling Guidelines. ASIC/001, Issue 1, Prepared by Sinander, P. and approved by Creasey, R. and Coirault, R. European Space Agency. Available at: http://www.eda.org/rassp/vhdl/guidelines/ModelGuide.pdf (accessed: 16 March, 2012).

Fang, T., Osterman, M., Pecht, M. 2006. Statistical analysis of tin whisker growth. Microelectronics and Reliability, Volume 46, Issues 5–6, pp. 846–849, Available at: http://www.sciencedirect.com/science/article/pii/S0026271405001319 (accessed: 14 September, 2011).

Fink, R.T., Killian, C.D., Nguyen, T., Druilhe, A., Daumas, F., Naser, J.A. 2010. Guidelines and a primer on application of field-programmable gate arrays in nuclear plant I&C systems. In: NPIC&HMIT 2010, 7–11 November, Las Vegas, Nevada, pp. 1305–1315.

Gaisler. http://www.gaisler.com (accessed: 20 March 2012).

Gaisler Research 2002. Lessons learned from FPGA developments. Technical report, FPGA-001-01, Version 0.2. 2002. Available at: http://microelectronics.esa.int/asic/fpga_001_01-0-2.pdf (accessed: 14 September, 2011).

Gassino, J. 2009. Introduction of Programmable Electronic Devices in nuclear safety systems: a new challenge in assessment. In: EUROSAFE 2009: Safety Implications of an Increased Demand for Nuclear Energy, Brussels, 2–3 November.

Hall, T.S. 2004. Field-programmable analog arrays: A floating-gate approach. Ph.D. dissertation, Georgia Institute of Technology, 2004. Available at: http://smartech.gatech.edu/xmlui/handle/1853/5071 (accessed: 14 September, 2011).

Hammarberg, J., Nadjm-Tehrani, S. 2005. Formal verification of fault tolerance in safety-critical reconfigurable modules. International Journal on Software Tools for Technology Transfer (STTT), Vol. 7, No. 3, pp. 268–279, 2005. Available at: http://www.springerlink.com/content/eep7dmtmec38v02g/ (accessed: 16 September, 2011).

Huffmire, T., Brotherton, B., Sherwood, T., Kastner, R., Levin, T., Nguyen, T.D., Irvine, C. 2008. Managing security in FPGA-based embedded systems. Design & Test of Computers, IEEE, Vol. 25, No. 6, pp. 590–598.

IAEA 2010. IERICS Review Team visits the Ukraine. http://www.iaea.org/OurWork/ST/NE/20101217.html (accessed: 20 March, 2012).

IAEA 2011. Preparing and Conducting Review Missions of Instrumentation and Control Systems in Nuclear Power Plants, IAEA-TECDOC-1662, 2011. Available at: http://www-pub.iaea.org/MTCD/Publications/PDF/te_1662_web.pdf (accessed: 5 December, 2011).

IBM RuleBase homepage,
http://www.research.ibm.com/haifa/projects/verification/RB_Homepage (accessed: 3 October, 2011).

IEC 61508. Functional safety of electrical/electronic/programmable electronic safety-related systems. Available at: http://www.iec.ch/functionalsafety (accessed: 16 September, 2011).

IEC 62566. Nuclear power plants – Instrumentation and control important to safety – Development of HDL-programmed integrated circuits for systems performing category A functions. Available at: http://webstore.iec.ch/webstore/webstore.nsf/Artnum_PK/46039 (accessed: 20 March, 2012).

IEC/IEEE 62530-2011. IEEE SystemVerilog – Unified hardware design, specification, and verification language. Available at: http://standards.ieee.org/findstds/standard/62530-2011.html (accessed: 16 September, 2011).

IEEE 1076-2008. IEEE Standard VHDL Language Reference Manual. Available at: http://standards.ieee.org/findstds/standard/1076-2008.html (accessed: 14 September, 2011).

IEEE 1364-2005. IEEE Standard Verilog Hardware Description Language. Available at: http://standards.ieee.org/findstds/standard/1364-2005.html (accessed: 14 September, 2011).

IEEE 1666-2005. IEEE Standard SystemC(R) Language Reference Manual. Available at: http://standards.ieee.org/findstds/standard/1666-2005.html (accessed: 19 September, 2011).

IEEE 1850-2010. IEEE Standard for Property Specification Language (PSL). Available at: http://standards.ieee.org/findstds/standard/1850-2010.html (accessed: 16 September, 2011).

iRoC Technologies 2004. Radiation Results of the SER Test of Actel, Xilinx and Altera FPGA instances. 2004. Available at: http://www.actel.com/documents/RadResultsIROCreport.pdf (accessed: 14 September, 2011).

Kharchenko, V., Sklyar, V. (Eds). 2008. FPGA-based NPP Instrumentation and Control Systems: Development and Safety Assessment, RPC Radiy, National Aerospace University “KhAI”, State STC on Nuclear and Radiation Safety, Kharkiv, Ukraine.

klabs.org. FPGAs and ASICs. NASA Office of Logic Design. A scientific study of the problems of digital engineering for space flight systems, with a view to their practical solution. Available at: http://klabs.org/fpgas.htm (accessed: 16 March, 2012).

Knight, J.C., Leveson, N. G. 1986. Experimental evaluation of the assumption of independence in multiversion software. IEEE Trans Software Engineering, Vol. 12, No. 1, pp. 96–109. Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.29.363 (accessed: 1 December, 2011).

Kojima, A., Kato, M., Tahira, M., Tadashi, M., Oda, N., Goto, Y., Hayashi, T., Sato, T., Igawa, S. 2010. Qualification of Toshiba's FPGA-based safety-related systems. In: NPIC&HMIT 2010, Las Vegas, Nevada, 7–11 November, pp. 935–943.

Lach, J., Bingham, S., Elks, C., Lenhart, T., Nguyen, T., Salaun P. 2006. Accessible formal verification for safety-critical hardware design. Annual Reliability and Maintainability Symposium, RAMS '06, 23–26 June.

Lach, J., Bingham, S., Elks, C., Lenhart, T., Nguyen, T., Salaun P. 2005. Accessible formal verification for safety-critical hardware design. MAPLD International Conference, 2005. Presentation slides available at: http://klabs.org/mapld05/presento/241_lach_p.ppt (accessed: 14 September, 2011).

Lahtinen, J. 2012. Application of the perspective-based reading technique in the nuclear I&C context. CORSICA work report 2011. VTT Technology 9. VTT Technical Research Centre of Finland, Espoo. Available at: http://www.vtt.fi/inf/pdf/technology/2012/T9.pdf (to be published).

Lu, J.-J., Chou, H.-P., Wong, K.-W. 2010. Conceptual design of FPGA-based RPS for the Lungmen Nuclear Power Plant. In: NPIC&HMIT 2010, 7–11 November, Las Vegas, Nevada, pp. 944–953.

Mason, M., O'Neill, K. 2005. FPGA reliability in space-flight and automotive applications. FPGA Journal, 2005. Available at: http://www.eejournal.com/archives/articles/20050906_actel (accessed: 5 December, 2011).

MathWorks. HDL Code Generation and Verification. Available at: http://www.mathworks.se/hdl-code-generation-verification/index.html (15 September, 2011).

Miyazaki, T., Oda, N., Goto, Y., Hayashi, T., Sato, T., Igawa, S. 2009. Qualification of FPGA-based safety-related PRM system. In: NPIC&HMIT 2009, 5–9 April, Knoxville, Tennessee.

Morris, W. 2003. Latchup in CMOS, IEEE. In: Proceedings of the 41st Annual International Reliability Physics Symposium, Dallas, Texas, pp. 76–84.

NASA. Tin Whisker (and other Metal Whisker) Homepage. Available at: https://nepp.nasa.gov/WHISKER (accessed: 14 September, 2011).

NI (National Instruments) 2008. FPGAs Under the Hood. 2008. Available at: http://zone.ni.com/devzone/cda/tut/p/id/6983 (accessed: 14 September, 2011).

NRC 2003. Emerging Technologies in Instrumentation and Controls, NUREG/CR-6812, 2003. Available at: http://pbadupws.nrc.gov/docs/ML0319/ML031920412.pdf (accessed: 7 January, 2011).

NRC 2006. Emerging Technologies in Instrumentation and Controls: An Update, NUREG/CR-6888, 2006. Available at: http://pbadupws.nrc.gov/docs/ML0608/ML060870216.pdf (accessed: 7 October, 2011).

NRC 2009a. Instrumentation and Controls in Nuclear Power Plants: An Emerging Technologies Update, NUREG/CR-6992, October 2009. Available at: http://www.nrc.gov/reading-rm/doc-collections/nuregs/contract/cr6992/cr6992.pdf (accessed: 1 December, 2011).

NRC 2009b. Wolf Creek Generating Station – Issuance of Amendment re: Modification of the Main Steam and Feedwater Isolation System Controls (TAC NO. MD4839). 2009. Available at: http://www.cs-innovation.com/docs/WCNOC_MSFIS_SER.pdf (accessed: 14 September, 2011).

NRC 2010a. Diversity Strategies for Nuclear Power Plant Instrumentation and Control Systems, NUREG/CR-7007, (ORNUTM-2009/302), February 2010. Available at: http://pbadupws.nrc.gov/docs/ML1008/ML100880143.pdf (accessed: 5 December, 2011).

NRC 2010b. Review Guidelines for Field-Programmable Gate Arrays in Nuclear Power Plant Safety Systems. U.S. NRC, NUREG/CR-7006, (ORNL/TM-2009/20), 2010. Available at: http://www.nrc.gov/reading-rm/doc-collections/nuregs/contract/cr7006 (accessed: 14 September, 2011).

OSCI, Open SystemC Initiative. Available at: www.systemc.org (accessed: 19 September, 2011).

Quinn, H., Graham, P. 2005. Terrestrial-based radiation upsets: A cautionary tale. In: Proceedings of the 13th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'05), April 18–20, 2005, Napa, California, USA, pp. 193–202.

Salaün, P., Daumas, F., Nguyen, T., Esmenjaud, C. 2009. FPGA/ASIC: A promising technology for future of I&C systems in power industry. In: NPIC&HMIT 2009, 5–9 April, Knoxville, Tennessee.

Schwank, J.R., Shaneyfelt, M.R., Fleetwood, D.M., Felix, J.A., Dodd, P.E., Paillet, P., Ferlet-Cavrois, V. 2008. Radiation effects in MOS Oxides. IEEE Transactions on Nuclear Science, Vol. 55, No. 4, pp. 1833–1853.

Sexton, F.W. 2003. Destructive single-event effects in semiconductor devices and ICs. IEEE Transactions on Nuclear Science, Vol. 50, No. 3, pp. 603–621.

She, J., Jiang, J. 2009. application of FPGA to shutdown system No. 1 in CANDU. In: NPIC&HMIT 2009, 5–9 April, Knoxville, Tennessee.

Smith, G. 2010. FPGAs 101; Everything You Need to Get Started. Elsevier, 245 p. 2010. Available at: http://www.sciencedirect.com/science/book/9781856177061 (accessed: 14 September, 2011).

Stott, E., Sedcole, P., Cheung, P. 2010a. Fault tolerance and reliability in field-programmable gate arrays. Computers & Digital Techniques, IET, Vol. 4, No. 3, pp. 196–210.

Stott, E.A., Wong, J.S.J., Sedcole, P., Cheung, P.Y.K. 2010b. Degradation in FPGAs: measurement and modelling. FPGA '10: Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays, ACM, USA, 2010. Available at: http://doi.acm.org/10.1145/1723112.1723152 (accessed: 14 September, 2011).

System Verilog. See: http://www.systemverilog.org (accessed: 19 September, 2011).

Wang, J. 2003. Radiation effects in FPGAs. Proc. 9th Workshop on Electronics for LHC Experiments, 2003. Available at: http://cdsweb.cern.ch/record/712037 (accessed: 14 September, 2011).

Wang, X., Holbert, K.E. Clark, L.T. 2010. Using TMR to mitigate SEUs for digital instrumentation and control in nuclear power plants. In: NPIC&HMIT 2010, 7–11 November, Las Vegas, Nevada, pp. 925–934.

Xiaoxuan, S., Samudrala, P.K. 2009. Selective triple modular redundancy for single event upset (SEU) mitigation. In: NASA/ESA Conference on Adaptive Hardware and Systems, AHS 2009. Pp. 344–350.

Xilinx/FAQs. Formal Verification – Frequently Asked Questions (FAQs). Available at: http://www.xilinx.com/support/answers/25007.htm (accessed: 15 September, 2011).

Xing, A., de Grosbois, J., Archer, P., Awwal, A., Sklyar, V. 2010. FPGA-Based Controller in CANDU® Nuclear Safety-Reactor Applications. In: NPIC&HMIT 2010, 7–11 November, Las Vegas, Nevada, pp. 1337–1344.

Yastrebenetsky, M., Sklyar, V., Rozen, Y., Vinogradskaya, S. 2009. Safety assessment of FPGA-based ESFAS for Kozloduy NPP. In: NPIC&HMIT 2009, 5–9 April, Knoxville, Tennessee.

Ziener, D.M. 2010. Techniques for increasing security and reliability of IP cores embedded in FPGA and ASIC designs. Dissertation, University of Erlangen-Nuremberg, Verlag Dr. Hut, Munich, Germany, July, 2010. Available at: http://www.dr.hut-verlag.de/978-3-86853-657-7.html or http://www12.informatik.uni-erlangen.de/people/ziener/pub/DanielZienerDissertation.pdf (accessed: 14 September, 2011).

Series title and number
VTT Technology 10

Title	**The current state of FPGA technology in the nuclear domain**
Author(s)	Jukka Ranta
Abstract	Field programmable gate arrays are a form of programmable electronic device used in various applications including automation systems. In recent years, there has been a growing interest in the use of FPGA-based systems also for safety automation of nuclear power plants. The interest is driven by the need for reliable new alternatives to replace, on one hand, the aging technology currently in use and, on the other hand, microprocessor and software-based systems, which are seen as overly complex from the safety evaluation point of view. This report presents an overview of FPGA technology, including hardware aspects, the application development process, risks and advantages of the technology, and introduces some of the current systems. FPGAs contain an interesting combination of features from software-based and fully hardware-based systems. Application development has a great deal in common with software development, but the final product is a hardware component without the operating system and other platform functions on which software would execute. Currently the number of FPGA-based applications used for safety functions of nuclear power plants is rather limited, but it is growing. So far there is little experience or common solid understanding between different parties on how FPGAs should be evaluated and handled in the licensing process.
ISBN, ISSN	ISBN 978-951-38-7622-7 (URL: http://www.vtt.fi/publications/index.jsp) ISSN 2242-122X (URL: http://www.vtt.fi/publications/index.jsp)
Date	March 2012
Language	English
Pages	62 p.
Name of the project	CORSICA
Commissioned by	
Keywords	Nuclear power, instrumentation and control, I&C, FPGA
Publisher	VTT Technical Research Centre of Finland P.O. Box 1000, FI-02044 VTT, Finland, Tel. 020 722 111

The current state of FPGA technology in the nuclear domain

Field programmable gate arrays are a form of programmable electronic device used in various applications including automation systems. In recent years, there has been a growing interest in the use of FPGA-based systems also for safety automation of nuclear power plants. The interest is driven by the need for reliable new alternatives to replace, on one hand, the aging technology currently in use and, on the other hand, microprocessor and software-based systems, which are seen as overly complex from the safety evaluation point of view.

This report presents an overview of FPGA technology, including hardware aspects, the application development process, risks and advantages of the technology, and introduces some of the current systems.

FPGAs contain an interesting combination of features from software-based and fully hardware-based systems. Application development has a great deal in common with software development, but the final product is a hardware component without the operating system and other platform functions on which software would execute.

Currently the number of FPGA-based applications used for safety functions of nuclear power plants is rather limited, but it is growing. So far there is little experience or common solid understanding between different parties on how FPGAs should be evaluated and handled in the licensing process.

ISBN 978-951-38-7622-7 (URL: http://www.vtt.fi/publications/index.jsp)
ISSN 2242-122X (URL: http://www.vtt.fi/publications/index.jsp)