

---

## Introduction of Programmable Electronic Devices in nuclear safety systems: a new challenge in assessment.

*Jean GASSINO*

Institut de Radioprotection et de Sûreté Nucléaire, DSR/SAMS  
92262 Fontenay-aux-Roses Cedex, France

---

### **Abstract:**

Programmable Electronic Devices (PEDs) are integrated circuits such as FPGAs<sup>1</sup>, CPLDs<sup>2</sup> or ASICs<sup>3</sup>, increasingly used in the industry because their high capacity allows embedding complex functions. As a result, projects already exist to apply them to safety critical nuclear applications such as protection systems.

In addition to electronic aspects such as the influence of temperature, voltage or length of internal paths on timings, programming PEDs has many similarities with developing software. Both technologies allow implementing the same functions using for example complex conditional and iterative processing.

Experience feedback from the industry shows that PED programs are subject to design errors in the same way as software: a wrong instruction is wrong in both cases. A weak or insufficiently known design cannot be qualified by 100% testing or other post-design approaches, thus the ability to be verified must drive the whole development process.

So PED based systems cannot be developed and licensed as traditional “hardwired” systems. They need to be specified, designed and verified specifically to build and demonstrate their safety. An IRSN expert contributes to the development of a standard within the International Electrotechnical Commission (IEC) to tackle this issue.

## **1 INTRODUCTION**

Programmable Electronic Devices have increased in capacity since the 1980s to such an extent that they currently include in a single integrated circuit up to a billion transistors organized in millions of logical operators and memories. This allows them to perform many logic computations traditionally devoted to microprocessors.

Depending on production volumes, different kinds of PEDs are available to the designer:

- for small or intermediate volumes (e.g. less than 10.000 parts) such as those in the nuclear field, FPGAs or CPLDs are typically used. They include a high number of programmable logic blocks which communicate together through programmable connections. The intended application is implemented in the circuit by configuring these programmable elements, i.e. assigning a specific function to each one;
- for higher production volumes, ASIC may be more cost effective: such a circuit is directly manufactured according to a specific design, so only the required structures are actually built. This reduces the silicon area and therefore the cost of each part, but non-recurring costs (e.g. to produce specific masks) may reach millions of euros.

---

<sup>1</sup> Field Programmable Gate Array

<sup>2</sup> Complex Programmable Logic Device

<sup>3</sup> Application Specific Integrated Circuit

Many consumer electronic devices are based on PEDs because their capacity and versatility allow the designer to pack in a single circuit functions that previously required many ones. For example, a DVD player needed a microprocessor to perform tasks such as human machine interface, a digital signal processor for heavy computations such as video decoding, memory to store intermediate results, and “glue logic” to interface these parts together. Now, the designer may replace these circuits by a single high-capacity PED and so reduce costs. Hardware reliability is also improved as the total area of silicon as well as the number of soldering joints decrease, which lowers the potential for random failures.

PEDs are being introduced in safety critical nuclear applications, including computing the main logic of protection systems. This important trend has been recognized during the IAEA workshop on “The Applications of FPGAs in Nuclear Power Plants” held in October 2008 [1], where utilities, designers, researchers and regulators exchanged their views on this topic. For example, EDF presented in [2] engineering and research projects for refurbishments, upgrades and new applications with FPGA and concluded that this technology “*can provide very significant benefits to utilities*”.

## 2 DIFFERENCE BETWEEN PED AND HARDWIRED LOGIC

The traditional hardwired logic, used for example in the protection systems of the 900 MWe reactor series in France, is based on simple components such as resistors, diodes, transistors and relays. These components are soldered on an epoxy plate called printed circuit, and interconnected by copper wires running on the plate.

The complexity of hardwired logic designs is low because each of the simple components must be physically placed and “hardwired” on the printed circuit, which strongly limits the number of gates<sup>4</sup> available to the designer. Additionally their testability is good because all connections are physically reachable. Therefore, they can be extensively tested so that design errors are practically eliminated.

On the other hand a PED may include several million interconnected gates, which prevents testing it for all possible cases. Additionally, the inclusion of all gates within a single integrated circuit hides their connections, making the design less observable and testable if it has not been developed with testability in mind.

## 3 TYPICAL DEVELOPMENT FLOW

Programming the high number of gates and connections embedded within a PED is not feasible by hand.

The development process typically uses description languages and design tools running on workstations. It has both the characteristics of software design (e.g. the use of high level languages allows creating complex structures quickly and easily) and the characteristics of traditional electronic design (e.g. aspects such as clock skew, metastability, glitches or influence of voltage, temperature and length of internal paths on timings).

---

<sup>4</sup> A gate implements a simple logic function such as AND, OR, NOT, bit storage...

### 3.1 Design

Starting from the requirements, a preliminary design phase is typically performed to outline a solution; then a Hardware Description Language (HDL) such as VHDL<sup>5</sup> or Verilog<sup>6</sup> is used to produce a Register Transfer Level (RTL) description of this solution. This high level description is quite independent of the electronic technology that will be used to implement the target circuit.

This description is a synchronous parallel model, describing the intended behaviour by means of signals transformed by combinatorial functions and sequentially transferred between registers triggered by one or more clocks.

The RTL description has structural aspects, showing the logical relations between modules which can be designed specifically or taken from libraries. It also has behavioural aspects, making it possible to describe the function of a module by means of algorithmic descriptions similar to those used in software. The following figure gives an example of VHDL code.

```
adder : process (clk) begin
  if rising_edge (clk) then
    if enable = '1' then
      sumx <= inp1x + inp2x;
    end if;
  end if;
end process adder;
```

Figure 1: example of VHDL code

Tools used during this phase include editors, code analyzers and simulators to run the description on test cases.

Logic errors may be introduced during this phase, exactly as they may be introduced in software programs written for example in C or ADA. Of course, errors may also exist in the requirements.

### 3.2 Implementation: synthesis, place and route

The logic synthesis transforms the RTL description into a network of interconnected gates such as AND, OR, counters, registers, etc. The set of available gates depends on the target electronic technology.

The place and route phase defines the physical location of the gates on the silicon die and interconnects them taking into account the technological constraints (e.g. existence and capacity of routing channels, numbers of metal layers available for connection) as well as the application constraints (e.g. maximum propagation delay between two given gates).

As the gate count increases more and more connections have to be routed across the die, while requirements for speed usually impose to keep short some (or many) paths. These constraints may lead to modify the placement of some gates or even to duplicate them to shorten paths and this in turn impacts the whole place and route scheme. Finding the "best" solution is a hard problem (in the sense of computability), so only approximations may be found by the tools, at the cost of complex and fast-evolving algorithms.

<sup>5</sup> VHDL (Very high speed integrated circuits HDL) is standardized by IEEE 1076

<sup>6</sup> Verilog is standardized by IEEE 1364

Although these tools are comparable to software compilers (and actually the term “silicon compiler” is sometimes used), they need high skill from the designer who has to direct the implementation process by providing information on the required performances such as clock frequency, delays between all relevant signal pairs, power consumption, silicon area and on how critical signals such as clocks are to be distributed.

This necessary information can be really complex in case of large circuits. Its elaboration can thus be difficult and an error or omission (e.g. in specifying a maximum delay between two given signals) may result in generating a circuit suffering from subtle faults, almost impossible to detect by test.

### 3.3 Production

The result of the implementation phase is used to produce the final circuit. When an FPGA or CPLD is selected, a programming file is generated by the tools and used to physically configure the gates and connections of the circuit. When an ASIC technology is selected, additional steps are needed to produce the masks which in turn will be used to manufacture the circuits.

### 3.4 Verification tools

In parallel to the design, a "test-bench" is usually developed with the same language: the HDL description of the circuit is included in a broader program, which sends it inputs and reads its outputs.

This test-bench is used to simulate and test the HDL description, and can be associated to various tools for test coverage measurement.

The test-bench can be used again after synthesis and place/route, by running the same tests on the resulting descriptions, to ensure that they behave identically at least on these cases.

Static analysis tools are being introduced to propose complementary verification approaches. As for software, emerging techniques and tools allow proving whether some particular properties hold or not on an HDL description. Examples of static analyses are: checking specific properties, assertion based verification, checking the equivalence between products of different design phases or Static Timing Analysis.

### 3.5 New trend: Electronic System Level

The requirements of the circuit to be designed are more and more frequently captured by means of a high level description of the whole system which includes it. This Electronic System Level (ESL) description includes all hardware and software components: each one is represented by a behavioural model, and these models exchange information through communication channels in order to simulate the system.

This ESL description level uses languages such as SystemC or System Verilog, which are very close to software languages: e.g. SystemC is the C++ software language with additional libraries.

This ESL description is typically run on functional test cases in order to estimate the relevance of different system architectures, select the best one, and finally set up the requirements of each component, including the circuit to be implemented in the PED, in terms of behaviour and interface.

Finally this ESL description can be used to generate, to some extent automatically, the HDL description of the circuit to be implemented in the PED. This approach is known as “high level synthesis” or “behavioural synthesis”.

## 4 DIGITAL FAILURES

Two kinds of faults may induce failures in digital systems:

- random faults due to the wear-out of physical structures (such as soldering joints, silicon layers, etc.), phenomena like electro-migration or impact of high-energy particles or radiations. They can usually be modelled by statistical approaches,
- errors in the implemented logic, which may induce failures in cases determined by this logic but unknown due to development deficiencies. Typical logic errors are due to unplanned event coincidences or input sequences, invalid operations in special states, etc.

The effects of random failures are efficiently dealt with in safety systems at component and architecture levels, by means such as redundancy, self-supervision and periodic testing. Therefore, logic errors remain the major concern to the reliability of digital systems because their deterministic behaviour defeats the redundant architectures.

As seen before, current PED technology offers high capacity (up to several million operators in one circuit) and a powerful toolset to easily create and manipulate large and complex structures. Therefore, the potential for logic errors in the design is high.

It should be noted that 100% test of PED is often claimed either by the manufacturer of the raw integrated circuit or by the designer after the programming operation.

This means that the functionality of each individual gate within the circuit has been tested (e.g. an inverter correctly inverts the signal) but this does not address the correctness of the logic design. For example a given AND operator has been tested to provide a correct AND function, but this does not mean that an AND function in this place is appropriate for the considered application.

So this kind of 100% test addresses random faults of the circuit, not logic errors.

## 5 PED ARE NOT INHERENTLY ERROR-FREE

A frequent misunderstanding is that PEDs should be exempt from logic errors because, unlike microprocessor programs, their function is frozen in hardware. Consequently, they should be considered as pure conventional hardware (“hardwired”) and therefore free from errors.

Unfortunately, both analysis and experience feedback demonstrate that this is wrong, and that the potential for errors of PEDs is actually comparable to the one of software.

### 5.1 Analysis

Hardware languages such as VHDL or Verilog have many similarities with software languages such as ADA or C. For example, both ADA and VHDL have been developed on the demand of the US Department of Defense which required VHDL to borrow as much as possible from ADA to ease learning it. As a result, they have similar concepts and syntax.

It is often said that software languages are sequential (the instructions are executed one after the other) while HDLs are inherently parallel because they describe structures which operate in parallel.

This is somewhat true, and it would in fact make HDL more complex than software languages, but actually software also allows tasks to be run in parallel, either at language level (e.g. ADA includes parallelism) or through multitasking or multiprocessing architectures. Therefore synchronisation issues exist and must be properly handled in both technologies.

As a result, the same algorithms including complex features such as conditional or iterative processing may be implemented either in software or in HDL. This is confirmed by several facts:

- the traditional HDL development process includes verification by simulation, performed by software on a microprocessor-based workstation; it is guaranteed by the language standard that the simulated behaviour matches the one of the actual PED. This implies that any HDL parallel program may be executed on a sequential microprocessor to produce the same results<sup>7</sup>, and in fact this sequential execution is the reference for the next design steps;
- on the other hand, a microprocessor is itself a set of gates, which means that a PED or a part of it may be configured to be a microprocessor. Indeed, this capability is used more and more frequently to embed a microprocessor and other logic functions in the same PED, as explained before in the case of the DVD player. So, software programs may be executed by a PED.

As PEDs and microprocessors can emulate each other and their languages allow implementing the same algorithms with similar syntaxes, we see no rationale to claim that PED designs are error-free only because they are “hardware”.

The instruction “ $A=B+C$ ;” is valid both in C language (for microprocessors) and in Verilog (for PEDs). If the application needed in fact “ $A=B-C$ ;” then the design would be wrong for both.

## 5.2 Experience feedback

As PEDs have been used in aerospace for years, significant feedback is available from this field. Other ones such as the automotive industry have also used PEDs, but their experience feedback is not so available.

In [3], NASA analyses failures due to design errors in logic circuits and observed during space flights or final ground tests. This report describes a wide range of errors, from “hardware” mistakes (such as incorrect use of special FPGA pins e.g. “mode pin”, or inadequate programming of the clock tree resulting in excessive skew) to “logic” mistakes such as bad behaviour at start-up or incorrect cases in state machines.

In 1997, the report already pointed out the fact that:

- *“a fundamental issue is how the complexity is managed to permit reliable design despite the increased size and complexity”*,
- *“logic errors are still common in space-flight projects, with bad circuits making it into flight hardware, on the ground and on-orbit”*.

In [4], the US Federal Aviation Administration (FAA) focuses on technologies used e.g. in fly-by-wire applications. It states that:

---

<sup>7</sup> The simulated execution may be slower (or could be faster) than the actual one, depending on the workstation computing power and on the simulated PED. But in any case, the simulation computes and displays the correct dates when the PED outputs are produced, thus the results are exact from both logical and temporal aspects.

- *“Not only are there the normal hardware integrity issues for safety-critical systems, but now all the issues of software correctness apply also”,*
- *“Part of the problem is due to the sheer complexity”,*
- *“Error-free parts can no more be guaranteed than one can promise error-free software”.*

To summarize, the report points out that, although digital technology is not new, the increasing complexity of the devices introduces new problems, and refers to *“New threats from an old technology”*.

[5] compiles problems encountered by European Space Agency (ESA) with FPGAs. Here again a wide range of design and methodology errors are described. Among the later category, it is worth mentioning:

- design and verification by the same individual,
- inadequate verification with inapt stimuli,
- managers and reviewers (review by people not technically qualified),

which are inadequacies already known in weak software development processes.

[6] describes the failure of the control unit of a Soyuz spacecraft during re-entry, leading to a ballistic fall rather than the normal closed-loop descent. The failure originated in a logic error in an electronic circuit, which was triggered under some rare conditions of pitch, roll and yaw. It is important to mention that the faulty circuit was used successfully for 24 years, which confirms that “proven in use” evidence must be analyzed with high care for PEDs as well as for software when safety is involved.

## 6 NEED FOR A SPECIFIC SAFETY APPROACH

More than 25 years with safety critical programmed systems in nuclear plants show that specific approaches are needed when programmed components are involved: extremely safe systems cannot be built like ordinary industrial products, even with additional testing or used in diversified combinations.

As it is the case in software, PED outputs depend on many conditional statements which make their behaviour discontinuous: if we know the outputs corresponding to two different input values, we cannot deduce from this the output corresponding to an intermediate input value because, due to conditional statements, the actual output may be very different from the interpolated one.

Therefore the full testing of an ordinary product would require the actual execution of all relevant cases, because such a product lacks the demonstrated design properties needed to reduce the execution cases to a limited number of classes.

The current output of a PED depends on its current inputs (typically a few hundreds for an FPGA) and of its internal memories (100.000 and more) which in fact depends on the past input values. This means that the number of relevant cases could reach  $2^{100000}$  (approximately  $10^{30000}$ ) or more, which makes 100% test unreachable<sup>8</sup>.

---

<sup>8</sup> Testing requires not only running the program for each test case, but also pre-defining each expected output value to compare it to the actual one. This is a difficult and time-consuming task as no convincing automated solution exists. Building automatically the expected values would need another program having similar requirements, but this approach is subject to correlated errors in both programs as introduced in paragraph 6.1. about N-versions.

## 6.1 Limitations of program diversification

Program diversification has been proposed to reach high reliability by using several less reliable products. In particular, “N-version” consists of N programs independently developed from the same requirements and used in an N-redundant architecture.

The underlying assumption is that these programs will not fail for the same case, and therefore the overall architecture will be very reliable.

Unfortunately the famous Knight and Leveson’s experiment about N-version<sup>9</sup> [8] showed that errors in 27 programs implementing the same requirements were not independent. In fact, the error independence hypothesis was rejected at 99% confidence level.

More, the analysis of the data collected during this experiment shows that redundant systems built with 3 “intermediately good” versions (based on 2-out-of-3 vote) are often not as good as the best versions used alone.

It may be the case that some parts of a requirement are more difficult to implement than others and therefore more subject to errors, or that designers tend to make similar mistakes, despite the fact that in Knight and Leveson’s experiment they had different backgrounds and geographical locations (no correlation was found between the programmers’ locations and their mistakes).

This result means that an extremely safe programmed system cannot be built by simply combining less safe ones.

## 6.2 Limitations of statistical approaches

Statistical approaches are sometimes proposed to build a high confidence level in a given product without having to know and to analyse its design details.

They start from the statement that a program has a probability of failure  $p$  for each execution case. Then different methods are applied, for example deriving from  $p$  the probability of success  $(1-p)^N$  for the execution of  $N$  independent cases: so the probability for having at least one failure when executing  $N$  independent cases is  $C = 1 - (1-p)^N$ .

This formula allows determining the number of cases  $N$  which, if executed without failure, shows a given upper-limit for  $p$  at a given confidence level  $C$ .

E.g. reaching  $C = 0.95$  confidence that  $p$  is less or equal to  $10^{-4}$  requires the execution of  $N = 29900$  cases without failure.

But while physical experiments such as throwing a coin may actually be random, e.g. due to the laws of physics (for example we definitely cannot know and control exactly both position and speed of the coin) it is not the case of a program, which is a logic formula.

A program will systematically output a wrong result in some determined cases and the correct result in the other ones.

This means that the « probability » for a program to fail is 1 for some determined cases and 0 for the other ones, which is different from a random experiment such as throwing a coin with constant probability (e.g. 0.5 if it is symmetric) for all throws.

So there is no constant probability of failure for a program, and therefore we think that applying the theorems of statistics as if such probability existed needs more justification.

---

<sup>9</sup> Knight and Leveson have evaluated the assumption of statistical independence of failures in programs developed independently: 27 programs were developed from the same specification and tested. The results showed that this assumption must be rejected for this experiment.

Statistical testing proposes to randomly select the tests within an “operational profile”, which is a set of “use cases” representing the different inputs seen by the system in operation and weighted according to their frequency of use.

But it seems very difficult to predict and to weight accurately all possible cases for e.g. a protection system, which has hundreds of inputs and whose outputs depend not only on the current combination of inputs but also on their histories<sup>10</sup>.

It has been suggested to record this profile on a previous, already operating, system. As the current outputs depend on the history of inputs back to a point which may be very old but is anyway unknown from the outside, it is not clear how this record can be split in cases guaranteed to be independent.

Also, we think that such a recording technique would not solve the issue for protection systems which, hopefully, quite never see the important cases (accidents) during the recorded operation. So the recorded operational profile and therefore the tests could include many irrelevant cases but few or none of the important ones.

Finally, we have not yet seen a mean to build a complete and accurate profile, in terms of both cases and weights. Thus we think that the profiles used in practice partly depend on subjective choices and so does the following random test selection performed within them.

In our opinion, it should therefore be proved that this selection is better or at least as good as the deterministic test selection made for safety critical systems, which means tests duly justified to cover all functionalities in all modes, all signals, vetoes, life indicators, etc., having verified structural coverage and built by independent people and/or organizations.

Until such a proof has been provided we prefer to stick to deterministic approaches to demonstrate the required correctness of the logic (deterministic) aspects of safety critical nuclear systems.

## 7 WORK ON A SAFETY APPROACH

As it is not possible to demonstrate high reliability for an ordinarily-designed PED, applying PEDs to a nuclear safety system needs compliance with strict process and technical rules. This includes the specification of its requirements, the selection of pre-existing circuits, libraries and tools, the design, the verification, the validation and the procedures for operation and maintenance.

No standard directly applicable to the use of PEDs in nuclear safety systems has been found, mainly because PED development has both software and traditional hardware aspects.

In particular the hardware standard of avionics (DO-254) has been considered because its application field has similarities with the nuclear one. It provides guidelines for the overall hardware life-cycle, but does not focus on PEDs. It recognizes that PED development involves characteristic technical activities, but does not provide specific guidance for them.

So, in [7] prepared for the US NRC, the Oak Ridge National Laboratory states that “*there is no ready-to-use regulatory guidance directly applicable to the FPGA-based safety-critical system design*”.

---

<sup>10</sup> The duration to be taken into account may be long. Let us consider a safety output (e.g. reactor trip) which depends on a value stored in an internal memory. Depending on the internal logic, this memory stores the value of a computation when a certain internal condition C holds, and then keeps it unchanged as long as the condition C is not met again. Then the output depends on the value taken by the computation the last time the internal condition C was met. This may require taking into account a long duration, may be back to the last plant outage.

Therefore a project was launched in 2007 by the International Electrotechnical Commission (IEC) to develop an international standard - IEC 62566 - to cover the "*Selection and use of complex electronic components for systems performing category A functions*". The standard mainly covers PEDs as discussed in this paper, and it should be ready by 2011. It aims at complementing existing documents such as the general hardware standard IEC 60987 to provide guidance for the specific case of safety critical PEDs.

An IRSN expert coordinates the project team that prepares the successive IEC 62566 drafts and implements the comments made by the IEC community. This team also includes experts from utilities, manufacturers and research laboratories.

The IEC rules do not allow disclosing the details of the project in its current status, but it may be mentioned that both process and product aspects are taken into account through the whole life-cycle to classically avoid introducing faults, eliminate the faults introduced anyway and tolerate the effects of the remaining ones.

This implies process aspects to structure the life-cycle in documented and auditable phases such that design decisions are exposed (no trick), and to implement independent verification and validation with well-defined technical objectives, as well as technical reviews.

Product aspects include guidance on functional and safety requirements of the PED to ensure they are as clear and complete as possible, use of standardized HDL languages as well as qualified and mutually consistent tools (e.g. for synthesis, place/route and verification), restriction to HDL structures having well-defined implementation and behaviour, completeness of the design (e.g. handling of all cases in terms of logic and timing), testability thanks to requirements such as synchronous and deterministic behaviour, verification of logic and electronic aspects after each relevant refinement, requirements on self-supervision or defensive design.

More information may be found in the presentation [9] given at the workshop [1], or on the "62566" page on IEC's website. Participation in further discussions is of course welcome.

## 8 CONCLUSION

Programmable Electronic Devices such as FPGAs, CPLDs and ASICs offer flexibility, computing power and high integration to the designers of programmed systems, and therefore an increasing number of functions move from software to PED in the industry.

This trend exists, although with some delay, in the nuclear field where the use of PEDs in place of some software-based systems may additionally solve issues such as long term availability of a given microprocessor type (often designed for the fast evolving consumer market) and the increasing complexity of some software architectures.

Anyway, PEDs differ from the traditional "hardwired logic" because they now embed several million configurable gates allowing them to be programmed in the same way as microprocessors and to execute identical functions. They are thus subject to the same logic errors as software and these errors may trigger deterministic failures affecting all redundancies of a system.

PEDs, of course, have also the characteristics of electronics including aspects such as clock skew, power drops or timing variability due to physical parameters.

No easy solution seems to exist to build and demonstrate high reliability for PEDs. Due to the potentially huge number of execution cases, exhaustive testing by "brute force" of a weak or insufficiently known design is out of reach.

We think that statistical approaches do not capture well the behaviour of the deterministic logic. Also we have explained why, in our opinion, the methods based on random selection of

tests do not yet identify and weight all possible temporal evolutions of the combined inputs of the system, with sufficient accuracy to predict or assess high reliability levels.

Associating some less reliable PEDs, independently designed from the same requirement, to build a more reliable N-version system is subject to the observed correlation of failures in independently developed programs.

So, failures due to design errors in safety critical PEDs must be avoided by special care taken throughout their entire life-cycle, including specification of requirements, selection of pre-existing components and libraries, design, verification, validation and procedures for operation and maintenance.

Both the process and the product characteristics must be addressed. As post-design approaches for verification of a deficient or incompletely known design are blocked by uncontrolled complexity, the verification must be accounted for from the very beginning of the development.

As these measures are also essential to the independent assessment of PEDs and no directly applicable guidance exists yet, IRSN has committed itself to working on this topic.

Thus, the International Electrotechnical Commission has launched a project to develop the future IEC 62566 standard, essentially devoted to the application of PEDs to category A (i.e. the most critical) nuclear functions.

The project team is coordinated by an IRSN expert and includes other experts from utilities, manufacturers and research laboratories to maximize the relevance of the ongoing work, which is submitted to and discussed with the IEC community.

As no major obstacle to a technical consensus about PEDs has been found yet, we are confident that the application of these devices in the nuclear field will be successful.

## 9 ACRONYMS

ASIC: Application Specific Integrated Circuit  
CPLD: Complex Programmable Logic Device  
ESL: Electronic System Level  
FPGA: Field Programmable Gate Array  
HDL: Hardware Description Language  
IEC: International Electrotechnical Commission  
PED: Programmable Electronic Devices  
RTL: Register Transfer Level  
VHDL: Very high speed integrated circuits HDL

## 10 REFERENCES

- [1] First Workshop on The Applications of FPGAs in Nuclear Power Plants. Co-organized by IAEA and Electricité de France. 8-10 October 2008, Chatou, France
- [2] EDF's Projects with FPGAs. Presentation at the First Workshop on The Applications of FPGAs in Nuclear Power Plants). Thuy Nguyen, Patrick Salaün, Frédéric Daumas, 2008
- [3] Logic Design Pathology and Space Flight Electronics. R. Katz, R. Barto, K. Erickson. NASA Goddard Space Flight Center, 1997

- [4] Design, Test, and Certification Issues for Complex Integrated Circuits. Federal Aviation Administration, DOT FAA AR-95/31, 1996
- [5] Lessons Learned from FPGA Developments, European Space Agency Contract Report. Gaisler Research, 2002
- [6] Russian SNAFU, crew absolved of any blame in anomaly laid to 24-year-old component design. Aviation week and space technology. June 2003
- [7] Survey of Field Programmable Gate Array Design Guides and Experience Relevant to Nuclear Power Plant Applications. Oak Ridge National Laboratory, July 2007
- [8] An Experimental Evaluation of the Assumption of Independence in Multi-version Programming (IEEE Transactions on Software Engineering). John C. Knight, Nancy G. Leveson, 1986
- [9] Objectives of the new Standard IEC 62566. (Presentation at the First Workshop on The Applications of FPGAs in Nuclear Power Plants). J. Gassino, 2008