Development of an FPGA-Based Controller for Safety Critical Application

A. Xing[1], J. de Grosbois[1], V. Sklyar[2], P. Archer[1] and A. Awwal[1]
[1] Atomic Energy of Canada Limited, Mississauga, Ontario, Canada
[2] Research and Production Corp. (RPC) Radiy, Kirovograd, Ukraine

Abstract

In implementing safety functions, Field Programmable Gate Arrays (FPGA) technology offers a distinct combination of benefits and advantages over microprocessor-based systems. FPGAs can be designed such that the final product is purely hardware, without any overhead runtime software, bringing the design closer to a conventional hardware-based solution. On the other hand, FPGAs can implement more complex safety logic that would generally require microprocessor-based safety systems. There are now qualified FPGA-based platforms available on the market with a credible use history in safety applications in nuclear power plants. Atomic Energy of Canada (AECL®), in collaboration with RPC Radiy, has initiated a development program to define a vigorous FPGA engineering process suitable for implementing safety critical functions at the application development level. This paper provides an update on the FPGA development program along with the proposed design model using function block diagrams for the development of safety controllers in CANDU® applications.

1. Introduction

FPGAs have been widely used at the component level in the design domain of system suppliers. In recent years, use of FPGAs at the application level is gaining increased attention worldwide for application in nuclear power plant instrumentation and control (I&C) systems, especially for safety critical applications. FPGA-based systems are beginning to appear in new plant I&C designs, as well as retrofits of operating plants [1]. In Ukrainian nuclear stations, the Reactor Trip System (RTS) and Engineered Safety Feature Actuation Systems (ESFAS) of VVERs (Russian pressurized water reactors) have been implemented using FPGAs. In Wolf Creek NPP, United States, the Main Steam and Feedwater Isolation System (MSFIS) logic has been implemented using FPGAs. The MSFIS application has been approved by the USNRC.

FPGA technology is not new to CANDU Nuclear Power Plants (NPP). However, use of FPGA in CANDU reactors is mostly at the embedded electronic component level. For example:

- Obsolete control computer disc drive replacements have used FPGAs;
- Clone replacements for obsolete control computers have been produced using FPGAs;
- The refurbishment shutdown system programmable digital controllers have contained FPGAs.

FPGAs embedded in generic re-usable components are delivered as part of a pre-developed product and are generally not user-programmed and fall within the scope of product/platform qualification. Their wide spread use helps to assure their design correctness and reliability. FPGA-based safety controllers on the other hand must be programmed specifically for the safety logic functions they will perform. The following discussion will concentrate on the use of

FPGAs in CANDU NPP safety systems at the application logic level, the level at which designs would be performed by AECL using a FPGA-based safety controller platform such as the RPC Radiy product. The process for implementing application level logic in FPGAs is analogous to safety-critical software development and requires a specific process and measures to ensure quality.

In retrofit projects, FPGAs have the potential to replace obsolete portions of existing CANDU safety and safety-related systems (including obsolete PLC/microprocessor-based systems and systems presently implemented in relay logic). The implementation and commissioning of a safety system using relay logic has its own drawbacks; e.g. it requires considerable wiring and commissioning time, and even small logic changes may delay the construction schedule, etc. FPGA systems can be pre-developed and pre-tested to a high level of confidence in an office environment, prior to shipping to site. Problems can therefore be eliminated by the designers early in the system design lifecycle. One area of application is to replace hardwired fuelling machine protection interlock logic (currently implemented using either computer or relay equipment) with FPGAs.

For new builds (such as Enhanced CANDU 6®), the FPGA technology provides a diverse digital platform safety critical systems, especially for safety systems that have lower complexity. FPGA systems can be designed such that they do not have the overhead of an operating system and runtime software, thus resulting in a simpler and more thoroughly testable system. Another beneficial feature of FPGAs is their ability to process separate functions independently and in parallel on the same integrated circuit. This feature makes it possible to implement self-testing and diagnostics independently from the primary functions and to process these functions in parallel with the primary functions, without interfering with the primary functions, or adding processing overhead. Applications can even be segmented into smaller blocks and implemented on separate FPGA chips or even separate boards reducing complexity level and facilitating verification/analysis/testing. This is in contrast to microprocessor-based solutions, where achieving independence between functions running on the same microprocessor is much more difficult due to the shared operating system and other common software or hardware services/resources that functions rely on to operate correctly. The high processing speed of FPGAs is yet another advantage over microprocessor-based systems primarily due to the hardware implementation with parallel processing capability. For example, fast neutronic trips are achievable by implementing trip function logic on a FPGA platform and taking advantage of its parallel processing capability, avoiding the excessive time delay normally seen on microprocessor-based computers. In addition, using FPGA-based systems makes it easier to deal with the obsolescence issue, compared to relay-based systems or microprocessor-based systems. In case the specific FPGA integrated circuit becomes obsolete and unavailable, the Register Transfer Level (RTL) representation of the existing design is easily portable to a new FPGA circuit. FPGA-based systems are also often inherently more immune to cyber security threats in comparison with microprocessor-based systems. Logic modifications to FPGA-based systems require complete reprogramming using FPGA-specific engineering tools making online cyber attracts less likely.

Working closely with RPC Radiy, AECL is currently embarking on an FPGA-based safety system development pilot project. The project consists of prototyping CANDU shutdown system logic on the proven RPC Radiy FPGA-based safety controller platform. The FPGA-

based controller is connected to a testbed in an environment similar to what AECL typically uses for Validation and Software Reliability testing of CANDU safety system computers. During this project, a rigorous FPGA development process is being defined and followed. Areas where FPGA-based systems can actually improve safety (e.g., more thorough diagnostics and redundancies) will also be studied. The development project serves as a design process validation activity and its objective is to demonstrate that an FPGA development process can be defined to meet regulatory expectations and comply with international IEC standards for implementing safety critical applications.

A generic channelized trip computer architecture is shown in Figure 1. For retrofits of existing stations, FPGA based controllers can be used to implement neutronic trips replacing ageing analog components without affecting existing process trips. For new builds, a single FPGA based trip computer can be used to implement both process trip parameters and neutronic trip parameters. In both cases, computerizing the neutronic trips will facilitate automated flux detector calibrations.

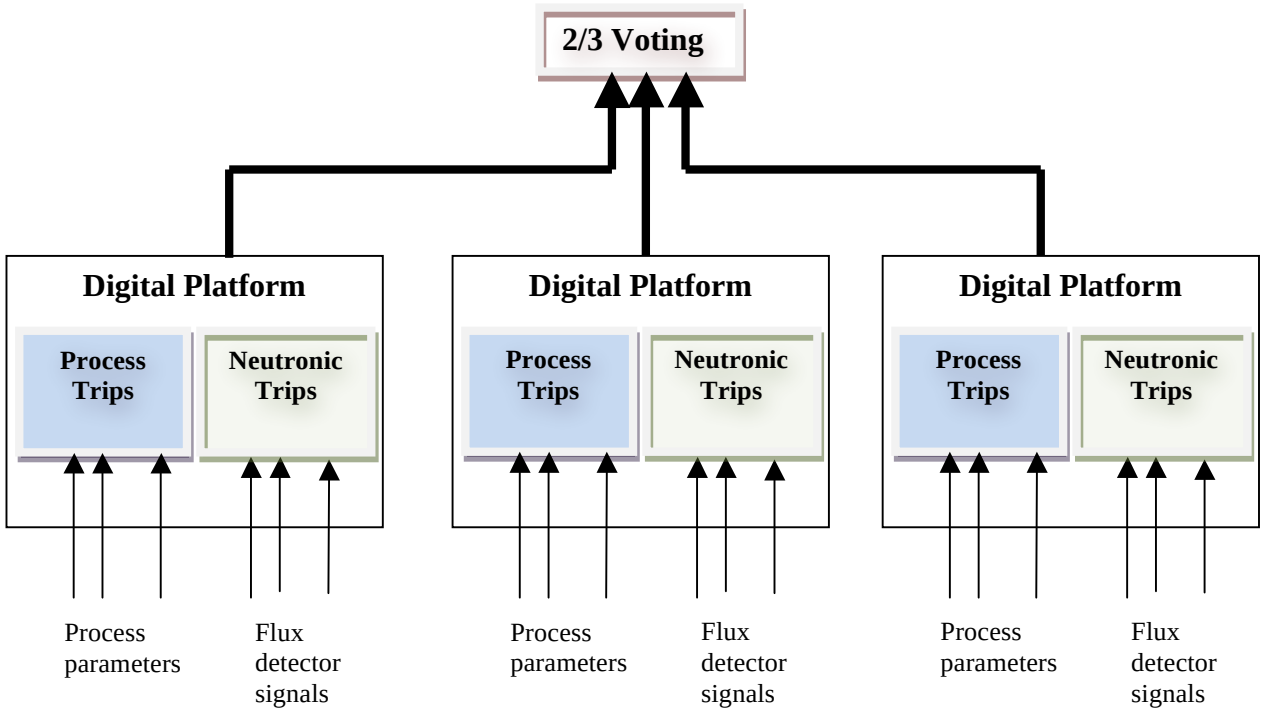

Figure 1: Channelized Trip Computer Architecture

2. FPGA Application Development Lifecycle

Although the FPGA end product is a hardware circuit, the FPGA application logic development process for safety systems shares a lot of similarities with the microprocessor-based software development process for safety critical software. These similarities make FPGA an attractive option to provide a diverse I&C platform in nuclear safety system applications. Experience gained in the microprocessor-based software development process is readily applicable to the FPGA application logic development process. However, special attention must be given to certain FPGA development stages which involve hardware design activities.

First of all, the three types of FPGA application code map equivalently to the three types of microprocessor-based application software as shown in Table 1.

Table 1: Comparison of FPGA-based Application Components with Microprocessor-based Software Components

FPGA-Based Application Logic	**Microprocessor-Based Application Software**
Application specific logic (e.g., reactor trip logic, interlock/protective logic)	Application specific software (e.g., reactor trip logic, interlock/protective logic)
Pre-developed electronic designs • Intellectual Property (IP) blocks such as analog I/O blocks, communication blocks, • System level electronic designs, • Soft cores.	Pre-developed software: • Runtime library, • Diagnostic/service routines, • Operating system/executive or system kernel.
Software tools (used in design, synthesis, place&route, etc)	Software tools (e.g., Compiler, Application oriented programming tools)

Pre-developed logic (software or hardware designs) and software tools are generally under the product qualification scope. Application specific logic/software is developed following a well-defined IEC Class 1 safety compliant development process.

At the application level, the class 1 FPGA development process on a FPGA-based controller product is quite similar to a typical microprocessor-based safety critical software development process, especially at the upper levels of the commonly referenced development V-model. The main differences between the two development processes exist at the bottom of the V-model. The proposed FPGA application level development process is illustrated in Figure 2 and further explanation follows.

The FPGA *Requirements Specification* phase states all the requirements that apply to the final FPGA circuit and the specification is considered as an input document to the FPGA development process. The FPGA *high-level design* phase includes decisions on the major design choices such as the balance between combinatorial logic and sequential design, decomposition into modules, the needed library functions and IP cores, etc. This phase is circuit independent and is analogous (i.e. similar level of abstraction) to the conventional software requirements specification (SRS) which is also platform independent. The FPGA *detailed design* phase develops a detailed description of the logic processing to be performed by the FPGA component, much like software design description and source code describing the logic processing to be performed by a microprocessor. This phase is analogous to the conventional software design description (SDD) and source code. The FPGA design is usually expressed using Hardware Design Language (HDL) which is at the same abstract level as ADA or C/C++ programming languages.

The *implementation* phase contains two steps: synthesis and place & route. The synthesis step translates the HDL code into an equivalent description that is expressed in terms of resources provided by the selected FPGA circuit. This circuit dependent description is called a netlist and is somewhat analogous to the conventional compiled object machine code that can be directly

interpreted or executed by a microprocessor. However, the FPGA implementation phase does contain a hardware design step, referred to as "place & route", to determine the physical position (in the circuit) of FPGA resources specified in the netlist. The output of the "place & route" is the bitstream file that is used to realize the final FPGA circuit.

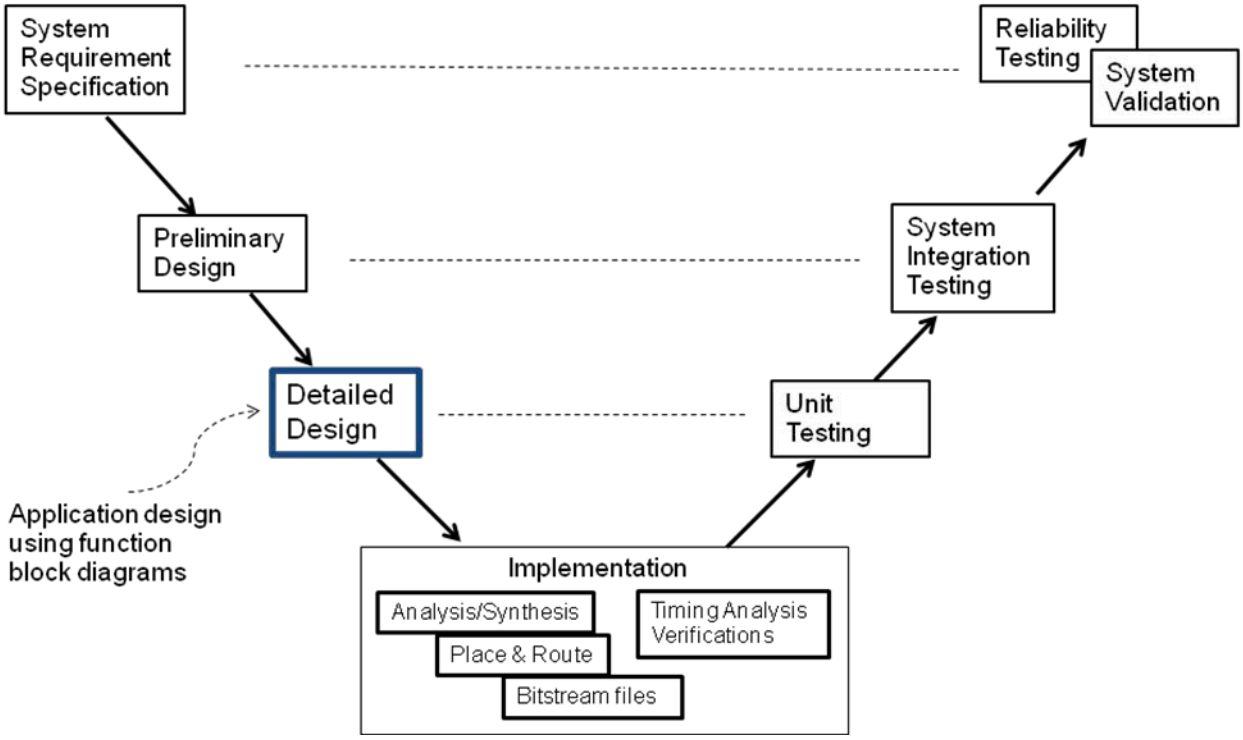

Figure 2: Proposed FPGA Application Level Development Process

Therefore, the FPGA development process maps very well to the conventional software development process up to the implementation / synthesis level as shown in Table 2. It should be noted that in a conventional application oriented microprocessor-based programming environment, e.g., using IEC 61131-3 function block diagrams, source code is automatically generated from the software design (in other word, SDD and source code are often indistinguishable).

Table 2: Comparison of FPGA development Activities with Conventional Software development Activities

FPGA Development Activities		**Conventional Software Development Activities**
High-Level Design		Software Requirement Specifications
Detailed Design		Software Design Description Source Code
Implementation	Synthesis	Compiled Object Code, Link
	Place & Route	-

With Radiy's FPGA based platform, AECL develops the application logic following the process shown in Figure 2 which is similar to the conventional micro-processor based software

development lifecycle. The output of the implementation phase is the application specific bitstream file which is used to complete the realization of the application FPGA on the control module (Platform specific configurations are typically performed by the vendor). The development process then continues by carrying out the remaining verification and validation activities.

To formally define a FPGA application logic level development process, additional procedures are required to address hardware design specific activities (e.g., place & route, and timing analysis). The FPGA development process may also require additional verification and or testing activities. The set of procedures that defines the FPGA development process will address all relevant requirements such as those specified in the draft IEC 62566 standard [2].

3. FPGA Application DESIGN Using Function Block Diagrams

Detailed Design is a key development phase in the FPGA development process. It formally specifies the application logic in a well-defined format, typically using Hardware Design Language (HDL). At the beginning of the FPGA development program, a design model based on graphical specification language was proposed and is demonstrated in Figure 3.

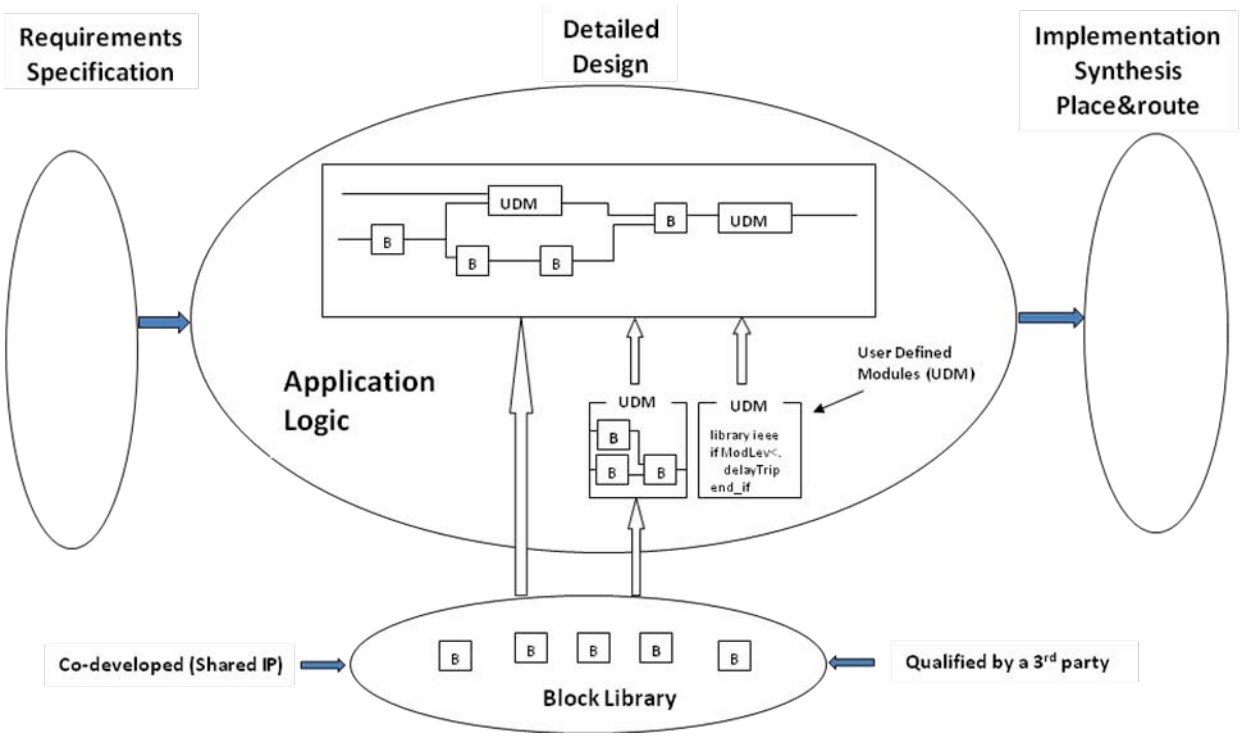

Figure 3: Detailed Design Logic Abstraction Model

The proposed model for the detailed design phase follows a structured three-level logic abstraction model: library blocks (lowest level), user defined modules (middle level) and application logic (highest level). This three-level model is considered suitable and desirable for safety critical software design using a graphical language. This model provides the required logic encapsulation without compromising logic reviewability and verifiability. As shown in

Figure 3, application logic is implemented graphically in the highest level using function blocks. Function blocks used in the application logic can be basic blocks from the block library (lowest level), or user defined modules **(**middle level). A Class 1 library of functions blocks has been be defined and is currently being developed in conjunction with Radiy. The block library will be qualified, by a 3rd party, for safety critical applications. User defined modules (mostly for encapsulation of commonly used logic) are considered part of the application and will be verified/validated as part of the application logic.

4. Project Management

For a FPGA application development project, the overall project organization is also similar to a microprocessor-based software development project, with strong focus on

- planning,
- appropriate types and levels of skills,
- quality assurance,
- configuration management,
- application development,
- independent review, verification and validation, and
- documentation of the development activities.

These processes are typically addressed by a quality project plan (QPP) with references to specific procedures that govern each of the development, verification and validation activities.

5. Challenges Associated with FPGAs in Nuclear Safety Applications

The FPGA development process shares a lot of similarities with the conventional software development process. However, realization in hardware results in unique characteristics that differentiate FPGA-based implementation from conventional microprocessor-based system. Realization of a logic design in an integrated FPGA circuit is inherently parallel and likewise there is no notion of synthesis and place & route in a conventional CPU-based software implementation. Additional attention is given to avoid potential race conditions, and this is easily addressed with the use of proven and robust verification tools and the use of synchronous designs. It is also important that experienced FPGA hardware engineers be involved throughout the design, implementation and verification process.

Although the final product is a hardware circuit, the FPGA application logic development relies on a comprehensive set of well proven and qualified software tools. Thorough verification of FPGA-based systems will be an important part of the licensibility for safety critical systems. Adequate verification methods have to be exercised for every step of the FPGA design process. Where tools are used to generate the outputs of some of the steps, for example the “netlist” from the HDL, either a reverse engineering tool or a thorough manual procedure will have to be used to verify that output. Otherwise the tool needs to be qualified to the appropriate level for the safety application. AECL is also collaborating with McMaster University on systematic model-based cross-verification methods using various tools to provide requisite confidence in the integrity of safety logic design. Formal verification methods for the elemental VHDL function block library will be explored as part of this development project. Advances in formal methods

that have been made in the academic world and applied in other industries may be applied to facilitate FPGA verification activities.

Safety system applications are not computation intensive and calculation accuracy is generally not an issue with floating point calculations. However, floating point support generally requires emulated soft core in the FPGA platform. Fixed point calculation can be used to simplify the design. Calculation accuracy with fixed point calculation will need to be confirmed.

Failure modes and reliability assessments of an FPGA-based safety system is also an important issue. FPGA-based systems will have to be demonstrated to meet the stringent reliability targets. Certification to IEC-61508 [3-5] Safety Integrity Level (SIL) 3 would constitute a good basis for reliability claims. Appropriate defenses against likely failure modes will be the key to achieving high reliability targets. Currently, RPC Radiy is in the process of obtaining a SIL 3 certification of their product. The proposed development approach will need to consider any constraints identified in the product safety manual once available.

6. Concluding Remarks

The FPGA technology provides a diverse digital platform for safety critical systems in nuclear applications. It provides fast response time equivalent to a hardwired solution, and programming flexibility as with microprocessor based PLCs. In the past, AECL has developed two rigorous software processes for the development of safety controllers, which have been reviewed and accepted by the regulator. Although some FPGA specific details need to be addressed, it is believed that a FPGA application logic development process can be defined to achieve the same level of rigor as the accepted software processes currently used for the development of microprocessor based safety controllers. This will be demonstrated with the FPGA development program. The well defined and licensable FPGA application logic development process will benefit both new builds and retrofit projects and enable the use of proven FPGA-based safety controllers such as the RPC Radiy platform to further improve existing safety system designs.

7. References

1. I. Bakhmach, V. Kharchenko, A. Siora, V. Sklyar and V. Tokarev **"**Advanced I&C Systems for NPPs Based on FPGA Technology: European Experience", Proceedings of the 17th International Conference on Nuclear Engineering ICONE17 July 12-16, 2009, Brussels, Belgium.
2. IEC 62566, "Nuclear Power Plants Instrumentation and Control Important to Safety, Selection and Use of Complex Electronic Components for Systems Performing Category A Functions, Committee Draft for Vote (CDV), 2010-01-29.
3. IEC-61508-1, "Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems – Part 1: General Requirements", 1998.
4. IEC-61508-2, "Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems – Part 2: Requirements for Electrical/Electronic/Programmable Electronic Safety-related Systems", 2000.
5. IEC-61508-3, "Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems – Part 3: Software Requirements", 1998.