VHDL Verification of FPGA based ESF-CCS for Nuclear Power Plant I&C System

Restu MAERANI[1], and Jae Cheon JUNG[2]

1. Department of NPP Engineering, KINGS, Ulsan, 45014, Indonesia (maerani@email.kings.ac.kr)
2. Department of NPP Engineering, KINGS, Ulsan, 45014, Republic of Korea (jcjung@kings.ac.kr)

Abstract: Verification becomes the focus of activities during the integration phase of design life cycle in the development of the system. Verification methods that will not take much cost and time should be properly selected, accordance with the Measurement of Effectiveness (MOEs) need. Verification is one phase that must be done after completing the implementation process. Since Instrumentation & Control (I&C) system has a role as a very crucial to the control protection system in Nuclear Power Plant (NPP), then software verification is very essential and shall to be achieved for safety critical issue in system level. According to IEEE 1076-2008 standard, VHDL is a language that is easy to read by machines and humans; and make it easier for process development, verification, synthesis and testing for hardware reliability in the design. Because this design uses VHDL code for Field Programmable Gate Array (FPGA) based Engineered Safety features – Component Control System (ESF-CCS) and by referring to the NUREG/CR-7006 during VHDL verification on behavioral simulation process, it should be equivalent with the post layout simulation. Furthermore, Vivado will be used as the VHDL verifier, where the VHDL code itself is created, in order to simplify the process of verification with this design life cycle phase on re-engineering process. By using this methodology, the testing process will automatically can be represented as one of verification process from several software verification methods that can be developed.

Keyword: Verification, FPGA, Software, Testing

1 Introduction

In the digital Instrumentation and Control System (I&C) technology of nuclear power plants, is expected to have the ability to meet the safety criteria of nuclear reactors in order to avoid 3 problems that can occur; software common-cause failure, failure interaction between operator and Man Machine Interface System (MMIS) and the non-detectability of software failure [1]. The decision to choose FPGA rather than using PLC is due to reduce the system complexity, safety from cyber-attack and economic calculation if measured from the Measurement of effectiveness (MOE).

Development of Field Programmable Gate Array (FPGA) based Safety Injection Actuation System (SIAS) should refer to reverse engineering process to developed Group Controller (GC) and Loop Controller (LC) modules of Engineered Safety features – Component Control System (ESF-CCS) functions. Starting from project planning and definition concept phase, referring to the requirement, Design Basis Document (DBE), design specification and design requirements so that design implementation, phase of test and verification can be develop for re-engineering the system. VHDL verification is to verify software which is used to generate the code for the operational of the ESF-CCS system.

2 Theory

Safety Critical Software for I&C System

Instrumentation and control systems (I&C), systems and components at Nuclear Power plants (NPP) can be grouped into two categories: essential goods for safety and non-essential items for safety [2]. The reason why develop FPGA based system for the technology of I&C system is because [3]:

- FPGAs attractive for safety applications, where on-line reconfigurations would be unacceptable.
- Design and V&V for FPGA based platforms for safety applications can be made much simpler and less costly than for microprocessor based ones.

Software Classification and Category

- Protection (Safety Critical)
  Software whose function is necessary to directly perform RPS control actions, ESFAS control actions, and safe shutdown control actions. "Protection" class software designation is applied when the highest level of software quality assurance is needed.
- Important to Safety (ITS)
  Software whose function is relied on to monitor or test protection functions, or software that monitors plant critical safety functions.
- Important to Availability (ITA)
  Software that is relied on to maintain operation of

plant systems and equipment that are critical to operate the plant.

- General Purpose
  Software that performs some functions other than that described in the previous classifications. This software includes tools that are used to develop software in other classifications, but is not installed in the online plant system.

System Requirements

System requirements for ESF Actuation Response Time are:

- For both cases, low pressurizer pressure and High Containment pressure, total response time of safety injection initiation signal should be less than 40 seconds [4].
- The response time between input of a GC and output of a Loop Controller (LC) due to Plant Protection System (PPS) initiation signal shall be less than 240 milliseconds. This number includes the data acquisition time, logic processing, and communication time (Group Controller (GC) + High Speed Link (HSL) + LC)[4].
- The response time between input of a Control Panel Multiplexer (CPM) and output of a LC due to manual Engineered ESFAS actuation shall be less than 240 milliseconds. This number includes the data acquisition time, logic processing time, and communication time (CPM + HSL + GC + HSL + LC) [5].

System Architectural Design Verification

The quality of an architectural description refers to its capability to meet the needs and concerns of the stakeholders for whom it was constructed. Such concerns typically include understandability, consistency, completeness, and analyzability of the description [6].

The objectives of Architectural Design Verification are to assure that [7].

- The architecture satisfies the system requirements,
- The system architecture is realizable,
- The selected architecture is based on specified selection criteria,
- The basis for verifying the system elements is defined
- The basis for integration of the system elements is established.

IEEE 1012 defines several tasks to be carried out in this process.

Software Design

The purpose of the Software Detailed Design Process is to provide a design for the software that implements and can be verified against the requirements and the software architecture and is sufficiently detailed to permit coding and testing [8].

In software design, software requirements are transformed into an architecture and a detailed design for each software component. The design includes databases and system interfaces (e.g., hardware, operator/user, software components, and subsystems).

The design of FPGA-based systems is expressed in Hardware Definition Language (HDL). In this project, VHSIC Hardware Description Language (VHDL) which is defined in [9] was used. EPRI [10] provides guidelines for the design of FPGA-based systems. HDL coding rules are detailed in NUREG 7006 [11]. These rules are aimed at ensuring that the hardware design practices and HDL code results in a system that is reliable, robust, traceable and maintainable.

Further requirements, specific to the design process are detailed in IEC 62566 [12]. This is currently the only standard that deals with FPGA-based safety I&C systems. However, it is not endorsed by the U.S NRC and therefore serves only as a reference.

Software Design Verification

The objective of Software Design Verification is to demonstrate that the design is a correct, accurate, and complete transformation of the software requirements and that no unintended features are introduced.

The first verification task involves ensuring the design, expressed in VHDL, is syntactically correct. The design was developed using the Xilinx tool which has the capability to verify that the code is correctly written. In order to verify that the design is functionally correct, test benches were designed.

Verification

Verification is the task of determining if the implementation of a model has been done correctly. For ordinary software systems, testing is conducted to verify that the system generates absolutely predictable

outcomes based on test data.

Simulation models have an expected rather than an absolute behavior, and may have widely differing

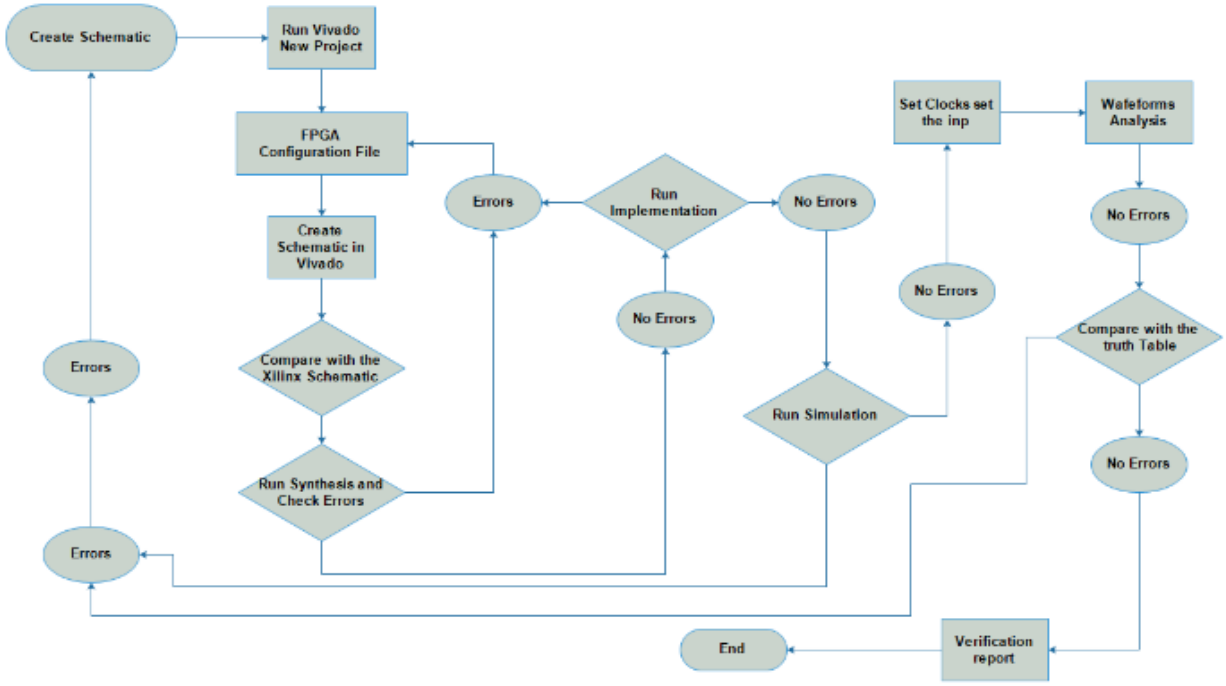


Fig. 1. VHDL Code Verification Flow Chart

results depending on configuration and input data. The kind of testing used in development of software systems is used to get a simulation model in functional order, but additional testing is required for verification and validation of the simulation model [13].

Fundamentals of Verification:

1. Tracing the model flow via flow diagram or otherwise to determine if the model is taking all possible actions in the course of a run.
2. Model outputs making sense for various combinations
3. Dumping data to see if the values are what they should be at simulation end.
4. Observing model behavior Validation

Test Case Development

Test cases for this system were developed, based on [14] as follows:

- Requirements based test case selection
- Normal Range Test Cases: The objective of normal range test cases is to demonstrate the ability of the software to respond to normal inputs and conditions.

Software Implementation Verification

The purpose of the development and implementation is to realize a specified system element. This process transforms specified behavior, interfaces and implementation constraints into fabrication actions that create a system element according to the practices of the selected implementation technology. The system element is constructed or adapted by processing the materials and/or information appropriate to the selected implementation technology and by employing appropriate technical specialties or disciplines. This process results in a system element that satisfies specified design requirements through verification and stakeholder requirements through validation [15].

3 Methodology

VHDL code Testing

Every VHDL implementation goes through extensive verification. How do we verify the circuit behaves as expected? We basically provide stimulus to the circuit at its input port and check its output. We change the input and check the output again. We continue doing it until we exhaust all possible inputs. If output under all conditions is as expected, the circuit stands verified.

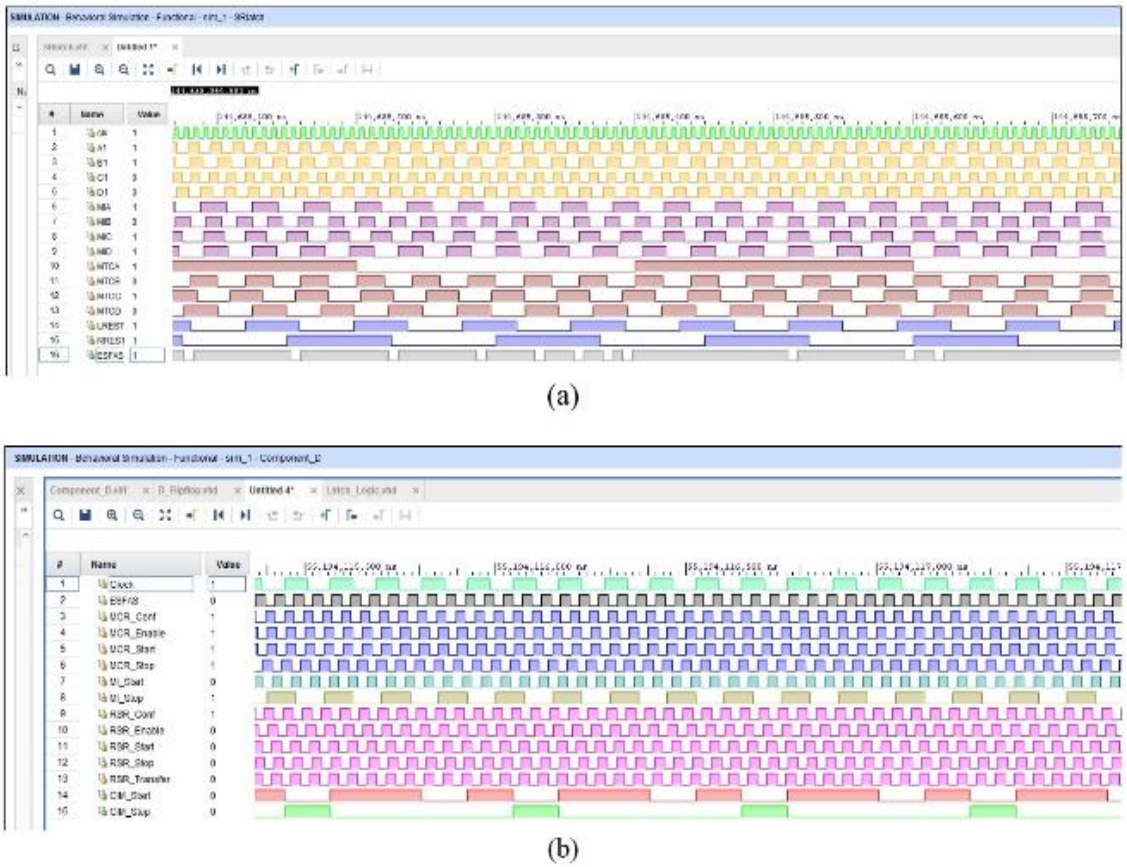
In VHDL code verification process the following steps had been done to verify the code:

1. Create writable folder outside VIVADO
2. Create a VHDL code using XILINX.
3. Run VIVADO and add XILINX file as a source.
4. Run synthesis.
5. Run Implementation.
6. Run Schematic.
7. Run Simulation.
8. Use generated VHDL to simulate the schematic for verification of all logic gates and hardware functions.
9. Defines the unit of time which we will use to change the values of the inputs x and y at certain periods of time. The code changes the values of x and y at defined intervals of time.
   - Specify Input/output ports of VHDL
   - Set Clock, Reset values and HDL start time
10. Run the simulation under all possible scenarios
11. Check the output waveform and compare it with the designed truth table
12. Verify the possibility to generate bit stream file that can deploy for FPGA
13. Write VHDL Verification Code

According to FPGA Basys3 board hardware features it has limited digital inputs/output ports (24 I/O/digital ports). According to system design there are more than 25 Input/output signals

The proper design is to use two FPGA Basys3 to implement the whole SIAS to actuate Pump.

The first board process the signal from PPS, MTC, Minimum Inventory (MI) system level and rest signals. The second board to process the signals coming from selective 2 out of 4,signals related to SIAS from Main Control Room (MCR) and the signals coming from



(a)

(b)

Fig. 2. Simulation Waveform (a) System Level, (b) Component Level

Remote Shutdown Room (SRS) to generate start /stop signal to pump A.

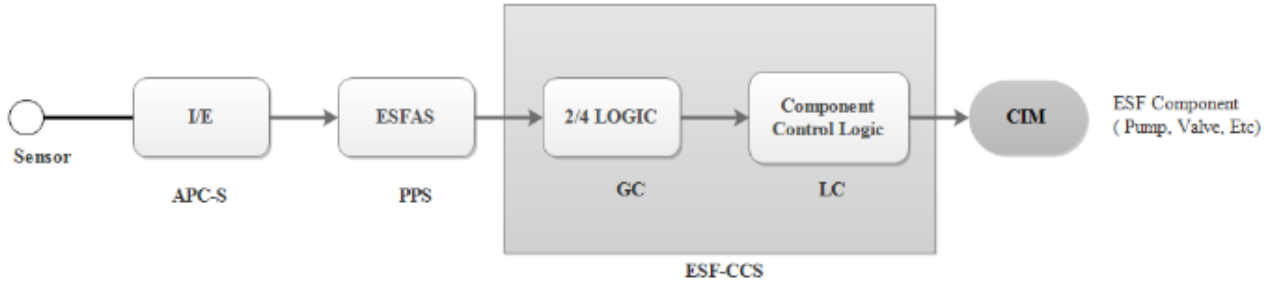


Fig. 3. ESF - CCS Operation

VHDL Code Verification of Selective 2 out of 4 Logic.

After completion of synthesis verification process, implementation verification process start to verify the possibility of the hardwired logic gates structure and the configuration file to deploy to FPGA board and timing constrains for gate synchronization

One of the missions to verify if the same code after synthesis and implementation verifying processes will generate the designed logical synthesis using VIVADO

After completion of synthesis verification process, implementation verification process start to verify the possibility of the hardwired logic gates structure and the configuration file to deploy the FPGA board and timing constrains for gate synchronization

4 Conclusion

The purpose of verification activity is to prove the completeness of the design, to make sure that the design can be tested its reliability, has complete and same specification and requirement from the design specification, system criteria and technical requirements that allowed. This method also we call it with white box testing which is examine the structure of the code by looking at the code itself. Unit testing is generally done within a class or a component.

VHDL code verification technique are for the software implementation level to test the program source code. It targets the faults that the programmer may introduce in the source code such as coding errors or bugs [16].

Acknowledgement

This research was supported by the 2017 Research Fund of the KEPCO International Nuclear Graduate School (KINGS), Republic of Korea.

References

[1] H.-W. Huang, C. Shih, S. Yih, and M.-H. Chen, “Integrated software safety analysis method for digital I&C systems,” *Ann. Nucl. Energy*, vol. 35, no. 8, pp. 1471–1483, 2008.

[2] M. S. Farias, R. H. S. Martins, P. I. N. Teixeira, and P. V. R. Carvalho, “FPGA-based I&C systems in nuclear plants,” *Chem. Eng. Trans.*, vol. 53, pp. 283–288, 2016.

[3] IAEA, “Application of Field Programmable Gate Arrays in Instrumentation and Control Systems of Nuclear Power Plants,” *Iaea Nucl. Energy Ser. Publ. NP-T-3.17*, vol. 1995–7807, no. Viena, 2016.

[4] US.NRC, “Chapter 7. Instrumentation and Controls,” in *The AP1000 Design Control Document Tier 2*, no. Dcd, .

[5] “Chapter 6. Engineered Safety Features,” in *APR1400 SSAR Document*, .

[6] “IEEE Recommended Practice for Architectural Description of Software-Intensive Systems IEEE-SA Standards Board,” 2000.

[7] IEEE, “Standard for System and Software Verification and Validation,” in *IEEE Std 1012-2012*, 2012.

[8] IEEE, “Systems and software engineering - Software Life Cycle Processes,” in *IEEE Std 12207-2008*, vol. Second Edi, 2012.

[9] IEEE, “Std 1076-2008, Standard VHDL Language Reference Manual,” 2009.

[10] R. Fink, C. Killian, and T. Nguyen, “Recommended Approaches and Design Criteria for Application of Field Programmable Gate Arrays in Nuclear Power Plant Instrumentation and Control Systems,” 2011.

[11] RTCA, "Software Consideration in Airbone System and Equipment Certification," in *RTCA/DO-178B*, 1992.

[12] IEC, "Nucear Power Plants - Instrumentation and Control Important to Safety - Development of HDL - Programmed Integrated Circuits for Systems Performing Category A functions," in *International Standard IEC 62566*, 2012.

[13] *Verification and Validation of Simulation Models*. 2009.

[14] RTCA, "Software Considerations in Airborne Systems and Equipment Certification," *RTCA/DO-17B*, 1992.

[15] IEEE, "System and Software Engineering-System Life Cycle Process," in *IEEE Std 15288-2008*, vol. Second edi, 2012.

[16] M. Kooli, F. Kaddachi, G. Di Natale, A. Bosio, P. Benoit, and L. Torres, "Computing reliability: On the differences between software testing and software fault injection techniques," *Microprocess. Microsyst.*, vol. 50, pp. 102–112, 2017.