A Translator Verification Technique for FPGA Software Development in Nuclear Power Plants

Jaeyeob Kim [a *], Eui-Sub Kim [a], Junbeom Yoo [a]
[a]*Division of Computer Science and Engineering, Konkuk University*
1 Hwayang-dong, Gwangjin-gu, Seoul, 143-701, Republic of Korea
[*]*Corresponding author: radic2510@gmail.com*

1. Introduction

The PLC (Programmable Logic Controller) is an industry computer widely used to implement the safety-critical system such as RPS (Reactor Protection System) in NPP (Nuclear Power Plants). Increasing complexity of newly developed systems and maintenance cost, however, are now demanding more powerful and cost-effective implementation such as FPGAs (Filed-Programmable Gate Array) [1]. Although the FPGAs give a high performance than PLCs, the platform change from PLC to FPGA impose all PLC software engineers give up their experience, knowledge and practices accumulated over decades, and start a new FPGA-based hardware development from scratch.

We have researched to fine the solution to this problem reducing the risk and preserving the experience and knowledge [2, 3, 4]. One solution is to use the 'FBDtoVerilog' translator, which translates the FBD programs into behavior-preserving Verilog programs. In general, the PLCs are usually designed with an FBD, while the FPGAs are described with a HDL (Hardware Description Language) such as Verilog or VHDL. Once PLC designer designed the FBD programs, the 'FBDtoVerilog' translates the FBD into Verilog, mechanically. The designers, therefore, need not consider the rest of FPGA development process (e.g., Synthesis and Place&Routing) and can preserve the accumulated experience and knowledge.

Even if we assure that the translation from FBD to Verilog is correct, it must be verified rigorously and thoroughly since it is used in nuclear power plants, which is one of the most safety critical systems. While the designer develops the FPGA software with the FBD program translated by the translator, there are other translation tools such as synthesis tool and place&routing tool. This paper also focuses to verify them rigorously and thoroughly.

There are several verification techniques [5] for correctness of translator, but they are hard to apply because of the outrageous cost and performance time. Instead, this paper tries to use an indirect verification technique for demonstrating the correctness of translator using the co-simulation technique. We intend to prove only against specific inputs which are under development for a target I&C system, not against all possible input cases. If the proposed technique succeeds, then we can assure the translator worked correctly at least for the inputted programs.

We had developed the supporting tools for co-simulation such as 'Scenario Generator,' 'FBD Simulator,' 'FBD-Verilog Comparator' and so on. We, however, should implement more than three independent tools, individually. The independent execution of them can cause several disadvantages such as human errors generated by mistake and time consuming for change the tools. We thus developed the integrated tool to support the co-simulation. It contains the previous developed tools such as 'Scenario Generator,' 'FBD Simulator' and 'FBD-Verilog Comparator,' and implements the independent tools, internally and automatically. Thus, the designer can largely focus on the development of software.

This paper is organized as follows. Section 2 introduces FBD, Verilog HDL, EDIF, 'Scenario Generator', 'FBD Simulator' and co-simulation. Section 3 explains the developed integrated tool and process in details. Section 4 shows the efficiency of the developed tool with a case study using the Korea Nuclear RPS logic. Section 5 concludes the paper and provides remarks on future research extension.

2. Related work

2.1 FBD (Function Block Diagram)

FBD (Function Block Diagram) is one of five standard PLC programming languages defined in the IEC 61131-3 standard [6]. It is a graphical language for programmable logic controller design that can describe the function between input variables and output variables. A function is described as a set of elementary blocks. Input and output variables are connected to blocks by connection lines. FBD consists of an arbitrary number of function blocks connected together with links or wires similar to that of a circuit diagram. FBD has been widely used for developing software controllers of plants and machines because of its graphical notations and usefulness in implementing data flow based applications.

2.2 Verilog HDL

Verilog [7] is one of the most common HDLs used by IC (Integrated Circuit) designers. Designs modeled in

Verilog are technology independent, easy to develop and debug, and considered more readable than schematics. For this reason, Verilog is being increasingly used to specify software logic for process control system. Also, a variety of EDA (Electronic Design Automation) tools support the process of gate-level synthesis. Therefore, the development using Verilog has advantage that regardless of the manufacturing process, such as a semiconductor chip or a FPGA device, to develop and to design the circuit in concentration.

2.3 EDIF (Electronic Design Interchange Format)

EDIF [8] is a vendor-neutral format in which to store Electronic netlists and schematics. It was one of the first attempts to establish a neutral data exchange format for the EDA industry. The goal was to establish a common format from which the proprietary formats of the EDA systems could be derived. When customers needed to transfer data from one system to another, it was necessary to write translators from one format to other.

2.4 FBD Simulator

The 'FBD Simulator' is simulator for FBD. It has been developed in order to verify the correctness of the 'FBDtoVerilog'. After FBD is input, it automatically classifies the POU (Program of Unit) in the FBD. And it presents input, output, and local list. It is available to generate input values and simulate at the same time manually on cycle. Also it presents graphs of changes of value simultaneously.

2.5 Scenario Generator

The 'Scenario Generator' [9] is a tool that automatically generates a scenario. FBD can be only the input file for this tool. The 'Scenario Generator' can generate a scenario that reflects the features of the domain such as range of values, and to automatically generate an infinite number of scenarios. Generated scenario can be used interchangeably in the simulator because scenario has the only input value used in simulation.

2.6 Co-Simulation

Co-simulation is indirect verification technique of translator. It simulates programs with same scenario and compares results of simulation for confirming correctness. Comparison process simply compares results that programs output values which are occurred with the same elapsed time. On the other hand, it is called Behavior equivalence checking. Confirmation of correctness with co-simulation can make to enhance the reliability of the program so as to ensure that program which is input and translated program, at least, perform the same function.

3. The Integrated Tool for Demonstrating the Correctness of Translator

The whole process for demonstrating the correctness of translators with co-simulation is depicted in Fig 1. Before performing the co-simulation, each program (i.e., FBD, Verilog and EDIF) should be prepared with 'FBDtoVerilog' and synthesis tool. First, it reads three programs such as a FBD, a Verilog and an EDIF. Next, it automatically generates same scenarios for the FBD, the Verilog and the EDIF programs, respectively. Finally, it also automatically simulates the FBD, the Verilog and the EDIF programs, and compares each simulation result for checking whether each simulation results are equivalent or not. If all simulation outputs are equivalent, it produces the 'True,' otherwise it produces the counter example with graphical chart. The graphical chart makes for designer to know how two programs reach to the different state through tracing the sequence of variables. We developed the integrated tool to support the proposed technique.

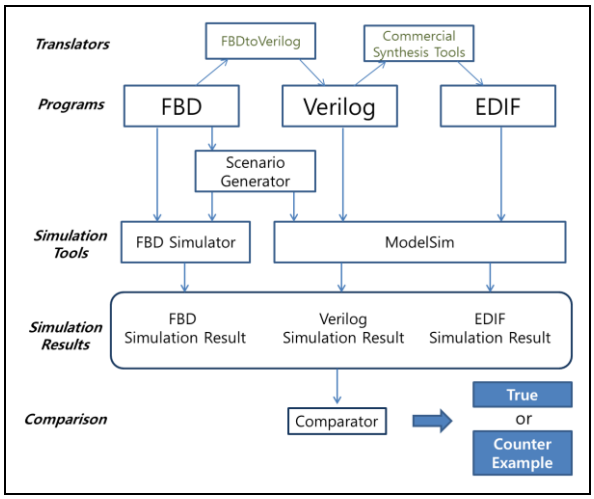

Fig. 1. The overall process for co-simulation

3.1 The Input programs of the integrated tool

This section explains each input programs for the integrated tool, which receive the three programs (i.e., FBD, Verilog and EDIF). Fig. 2 shows the input part in integrated tool, which consists of three parts for FBD, Verilog and EDIF. The FBD programs should follow the de facto standard of PLCopen TC6 [10], not permit a vendor-specific FBD format since the imported two tools, 'FBD Simulator' and 'Scenario Generator,' only support the PLCopen format. The Verilog programs should be translated by the 'FBDtoVerilog' translator. Once the designers make the FBD programs, they can naturally obtain the Verilog programs since the 'FBDtoVerilog' translator automatically translates the FBD programs into Verilog programs. After both co-simulations of FBD and Verilog programs were finished,

we can assure the translator works correctly at least for the FBD programs when simulation results are equivalent. The EDIF programs should be synthesized by the synthesis tool (e.g., 'Synopsis Synplify Pro' [11]). We also aim to verify the 3rd-party synthesis tools. If also the simulation results are equivalent, then we can assure the synthesis tool works correctly at least for the Verilog programs.

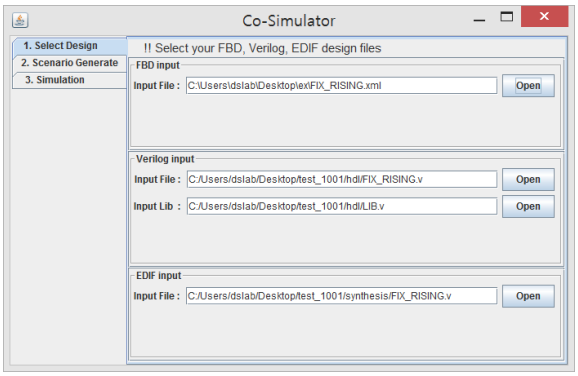

Fig. 2. The input part of integrated tool

3.2 The Scenario Generation

This section explains the scenario generation process. Fig. 3 show the scenario generation part in integrated tool, which consists of three parts for FBD model input, pou selection and setting table for constraints of scenarios. It is important to assure the scenarios we use are sufficient for demonstrating the behavioral equivalence.

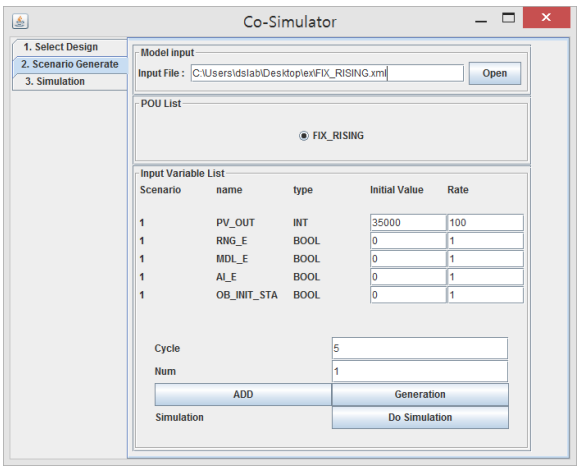

Fig. 3. The scenario generation part of integrated tool

We therefore use the 'Scenario Generator [9],' which we developed for generating various and more practical scenarios for executing FBD and Verilog programs simultaneously. It can automatically generate a number of scenarios to cover as many as possible cases. It also takes several constraints on input values, e.g., initial values, rate of change and maximum/minimum values to reflect the domain features. It then randomly generates a number of scenarios within predefined constraints on input values. We are now planning to extend it with more elaborate and systematic generation strategy, based on theories such as structural coverage criteria for co-simulation.

3.3 The Simulation & Comparison

This section explains the simulation and comparison process in integrated tool. Fig. 4 shows the screenshot of simulation and comparison result. The integrated tool performs the three simulations (i.e., FBD, Verilog and EDIF) with two simulators (i.e. 'FBD Simulator' and 'ModelSim' [12]). Also, it performs with the two comparisons between FBD with Verilog and Verilog with EDIF using two comparators (i.e., 'FBD-Verilog Comparator' and 'Verilog-EDIF Comparator'). The integrated tool, however, performs the two processes internally and automatically. Thus, it is not necessary for designers to consider the simulation and comparison processes. It can reduce the human errors generated during performing its processes and the performance time.

Internally, the FBD programs are simulated with 'FBD Simulator,' which saves the simulation result into .txt file. On the other hands, the Verilog and EDIF programs are simulated with 'ModelSim,' which originally generates the simulation result into wave form, but we convert to the .lst file (.txt file) from the wave form with command of 'ModelSim.' We can now easily compare each simulation results of FBD, Verilog and EDIF programs because we obtained each simulation result with .txt file. If each co-simulation results are equivalent, then we can assure that the translation from FBD into Verilog and the synthesis from Verilog into EDIF worked correctly. If each co-simulation results are not equivalent, on the other hand, then we can't assure that the translation or the synthesis worked correctly. And then, tool produces the counter example with graphical chart.

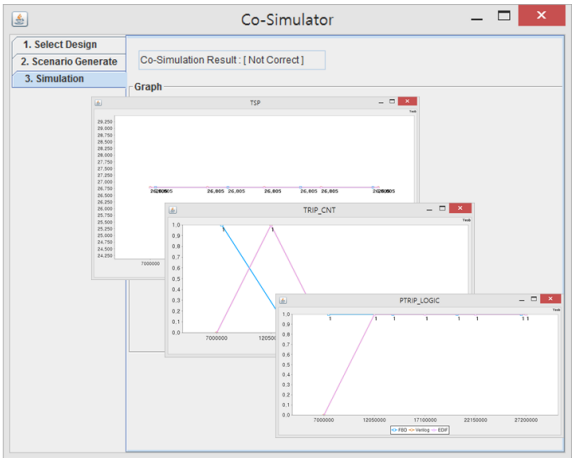

Fig. 4. The simulation and comparison part of integrated tool

4. Case Study

We tried to apply to an integrated tool using the FIX-RISING and FIX-FALLING of KNIC RPS BP. First, we translated FBD programs, FIX-RISING and FIX-FALLING, into Verilog programs using the 'FBDtoVerilog' and translated Verilog programs into EDIF using the 'Synplify Pro.' Then, we inputted programs to the integrated tool. In the 'Scenario Generator' part, we generated 100 scenarios for each program. Table I described the initial values of each scenario. Then, an integrated tool simulated programs using scenarios and compared simulation results.

The integrated tool shows the result of simulation of all inputted program (e.g., FBD, Verilog and EDIF) were equivalent. As a result, we can assure that the 'FBDtoVerilog' and 'Synplify Pro' worked correctly at least for the inputted programs. Also, comparison with the case of not using an integrated tool indicates that the case of using integrated tool more than 10 times faster. Thus, it is possible to save time for verification when using an integrated tool.

Table I: The Co-Simulation results

	FIX-RISING	FIX-FALLING	Total
Scenarios	100	100	200
Initial Values	27,000-28,000 (stepwise: 100)	12,000-13,000 (stepwise: 100)	All Correct
Rate of Change	10-100 (stepwise: 10)	10-100 (stepwise: 10)	
Cycles	100	100	

5. Conclusion

We developed the integrated tool in order to automatically perform the co-simulation. It integrates the several developed tool such as 'Scenario Generator,' 'FBD Simulator' and 'FBD-Verilog Comparator' and executes them internally and automatically. It has an advantage of reducing the time for verification and preventing human errors. We intend to demonstrate the correctness of translator such as 'FBDtoVerilog' and the commercial synthesis tools. As a result, we demonstrated them indirectly with case study that the 'FBDtoVerilog' and 'Synplify Pro' worked correctly at least for the inputted programs.

We are planning to extend the integrated tool to perform a JEDEC for verifying place&routing tool, which translates an EDIF into a JEDEC. We are also planning to elaborate the scenarios on the basis of adequate coverage criteria such as structural coverage or FBD testing coverage [13] in order to increase the confidence of verification.

Acknowledgements

This research was partially supported by a grant from the Korea Ministry of Strategy, under the development of the integrated framework of I&C conformity assessment, sustainable monitoring, and emergency response for nuclear facilities, and also partially supported by a grant from the Korea Atomic Energy Research Institute, the development of the core software technologies of the integrated development environment for FPGA-based controllers.

REFERENCES

[1] J. Yoo, J.-H. Lee and J.-S. Lee, A Research on Seamless Platform Change of Reactor Protection System from PLC to FPGA, Nuclear Engineering and Technology, Vol.45, No.4, pp.477-488, 2013.
[2] D.-A. Lee, E.-S. Kim, J. Yoo, J.-S. Lee, and J. G. Choi, FBDtoVerilog 2.0: An automatic translation of FBD into Verilog to develop FPGA, International Conference on Information Science & Application (ICISA), pp. 447-450, 2014.
[3] J.-Y. Kim, D.-A. Lee, Y.-J. Seo, J. Yoo, NuSCRtoFBD 4.0: Automatic Generation of FBD Program for FPGA development from NuSCR Formal Specification, Korea Computer Congress (KCC), pp. 1986-1988, 2014.
[4] J. Yoo, E.-S. Kim, D.-A. Lee and J.-G. Choi, An Integrated Software Development Framework for PLC & FPGA based Digital I&Cs, International Symposium on Future I&C for Nuclear Power Plants/ International Symposium on Symbiotic Nuclear Power System (ISOFIC/ISSNP), 2014.
[5] E.-S. Kim, D.-A. Lee, J. Yoo, A Survey of Verification Techniques for Translator, Code generator and Compiler, Korea Conference on Software Engineering (KCSE), Vol.15, No1, pp. 43-51, 2013.
[6] IEC, IEC 61131-3, International standard for Programmable controllers – part 3: Programming languages, 2013.
[7] IEEE Computer Society, IEEE Std 1364-2005, IEEE Standard Verilog Hardware Description Language, 2006.
[8] Electronic Industries Association, Electronic Design Interchange Format Version 2.0.0 ANSI/EIA-548-1988, 1988.
[9] E.-S. Kim, D.-A. Lee, J. Yoo, The Scenario Generator for Verifying the Correctness of FBDtoVerilog Translator, Korea Information Processing Society, Vol.21, No.1, p. 599-602, 2014.
[10] PLCopen, PLCopen for efficiency in automation, http://www.plcopen.org
[11] Synopsis, Synplify Pro, http://www.synopsys.com/Tools/Implementation/FPGAImplementation/FPGASynthesis/Pages/SynplifyPro.aspx
[12] Mentors Graphics, ModelSim, http://www.mentor.com/products/fpga/model/
[13] E. Jee, S. Kim, S. Cha, I. Lee, Automated test coverage measurement for reactor protection system software implemented in function block diagram, Springer Berlin Heidelberg, 2010.