NuDE 2.0: A Model-based Software Development Environment for the PLC & FPGA based Digital Systems in Nuclear Power Plants

Junbeom Yoo , Eui-Sub Kim , Dong-Ah Lee
Computer Science and Enginering
Konkuk Univeristy
Republic of Korea
Email: {jbyoo, atang34, ldalove}@konkuk.ac.kr

Jong-Gyun Choi , Young Jun Lee , Jang-Soo Lee
MMIS Lab.
Korea Atomic Energy Research Institute
Republic of Korea
Email: {choijg, yjlee426, jslee}@kaeri.re.kr

***Abstract*—*NuDE 2.0* (Nuclear Development Environment) is a model-based software development environment for safety-critical digital systems in nuclear power plants. It makes possible to develop PLC-based systems as well as FPGA-based systems simultaneously. The *NuDE* starts from a formal requirement specification specialized for the nuclear domain, and synthesizes C and Verilog codes for PLC and FPGA, respectively, through a series of model transformation. It also provides various methods for formal verification and safety analysis with support of automatic CASE tools. We expect that the *NuDE* can be adopted as an effective method of bridging the gap between the PLC and FPGA-based developments as well as a means of gaining diversity.**

I. Introduction

A safety-grade PLC (Programmable Logic Controller) has been widely used as an implementation platform of safety-critical digital systems in nuclear power plants, such as RPS (Reactor Protection System) and ESF-CCS (Engineered Safety Features - Components Control System). While complexity of newly developing systems and maintenance cost of the old ones have increased rapidly, research es of an alternative platform for the PLC have begun. The solution of [1], [2], [3] proposes to use FPGA (Field-Programmable Gate Array) which can provide powerful computation with lower hardware cost.

The platform change from PLC to FPGA, however, is not so straightforward. It gives rise to a paradigm shift from the CPU-based software development to FPGA-based hardware development. PLC software engineers in nuclear domain should give up all experience, knowledge and practices accumulated over decades, and start a new FPGA-based hardware development from the scratch. For example, it is required to model design specifications with HDLs (Hardware Description Languages) such as Verilog and VHDL in place of FBD (Function Block Diagram) and LD (Ladder Diagram) [4]. The underlying working mechanisms of HDLs and the PLC programming languages are also quite different from each other. Besides the loss from the semantic gap, the platform change may result in potential causes leading to safety-related problems in regard to under-trained software engineers. Therefore, it is now strongly required to transit to the new development approach safely and seamlessly.

The *NuDE* (Nuclear Development Environment) [5], [6] is a formal model-based software development environment for safety-critical digital systems in nuclear power plants. It starts from a formal requirement specification written in NuSCR [7], and finally synthesizes C codes for PLCs through a series of model transformation. It also supports various levels of formal verification and safety analysis to check the safety and correctness of the transformed models. All verification and analysis are also supported by automatic CASE tools.

While the *NuDE* originally intended for the software development of the PLC platform, it is now extended for the FPGA platform. The *NuDE 2.0* can develop the PLC and FPGA software simultaneously from the same requirement or design specifications. We expect that the *NuDE* can reduce the semantic gap between the PLC-based and FPGA-based developments (*i.e.,* software *vs.* hardware), while keeping all experience and knowledge accumulated for decades. It can also be used as a means of gaining diversity of software design and implementation. The organization of the paper is as follows: Section 2 provides background information such as the typical PLC and FPGA software development processes. Section 3 introduces the *NuDE 2.0* and various supporting tools briefly, due to the limitations of space. Section 4 concludes the paper and provides remarks on future research extension.

II. The Software Development Processes for Safety-Critical Digital Systems

This section explains the typical software development processes for the PLC and FPGA-based platforms in order to understand how these are different from each other and get rational of the proposed integrated development.

978-1-4799-4833-8/14/$31.00

A. The PLC-based Software Development

Fig.1 shows the typical software development process for the PLC platform. A number of safety-critical digital systems in nuclear power plants such as RPS and ESF-CCS has been developed with the platform. SRS (Software Requirements Specification) is first written in natural languages, and then design specification is modeled with the PLC programming languages such as FBD or LD, manually. Commercial PLC vendors provide software engineering tools to support mechanical translation from the FBD/LD programs into ANSI C and executable codes.

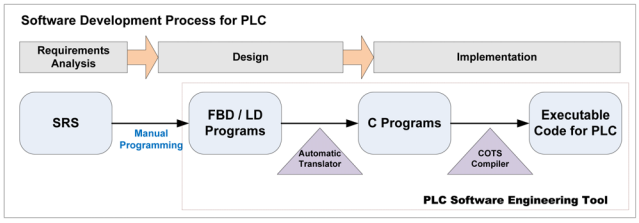

Fig. 1. A typical software development life-cycle for PLC platforms

Vendors such as *AREVA*, *invensys* and *ponu-Tech* have provided safety-level PLCs and their own software engineering tool-sets for safety-critical systems in nuclear power plants. '*SPACE*' [8] is the tool for AREVA's PLC '*TELEPERM XS*' [9], while '*TriStation 1131*' [10] is the one of *invensys*. KNICS [11] and *ponu-Tech* in Korea have recently developed a safety-level PLC '*POSAFE-Q*' and its software engineering tool-set '*pSET*' [12]. The tool-set provides a graphical editor for FBD and LD programming languages and generates ANSI-C programs mechanically.

B. The FPGA-based Software Development

Fig.2 depicts the typical FPGA development process [13] including FPGA software. Software requirements and designs are specified first, similar to the PLC-based development. The HDL code phase programs the design in HDLs such as Verilog or VHDL, manually. (It is common that the code programmed with HDLs is regarded as '*software*'.) After programming Verilog (or VHDL) programs, an FPGA is produced mechanically through several steps, such as synthesis, optimization, placement & routing, design verification, configuration and downloading. Software synthesis tools provided by FPGA vendors such as '*Xilinx ISE Design Suite*' , '*Altera Quartus II*' and '*Actel Libero IDE*' support all steps seamlessly and mechanically. They also provide systematic verification and simulation facilities for each synthesis step.

FPGA, however, is not yet used for implementing safety-critical digital controllers in nuclear power plants, since its development process and techniques are not familiar with software engineers in nuclear domain and not acknowledged safe by government authorities. Safety and correctness of the FPGA software development process (*i.e,* synthesis process and tool) up to the level of the safety-grade PLC should be demonstrated sufficiently.

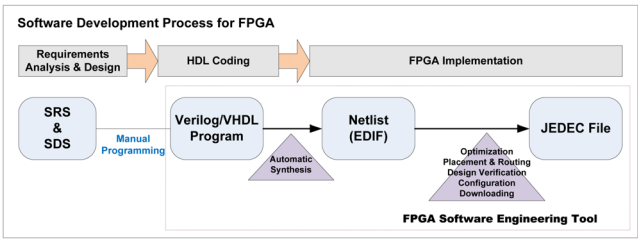

Fig. 2. A typical software development life-cycle for FPGA platforms

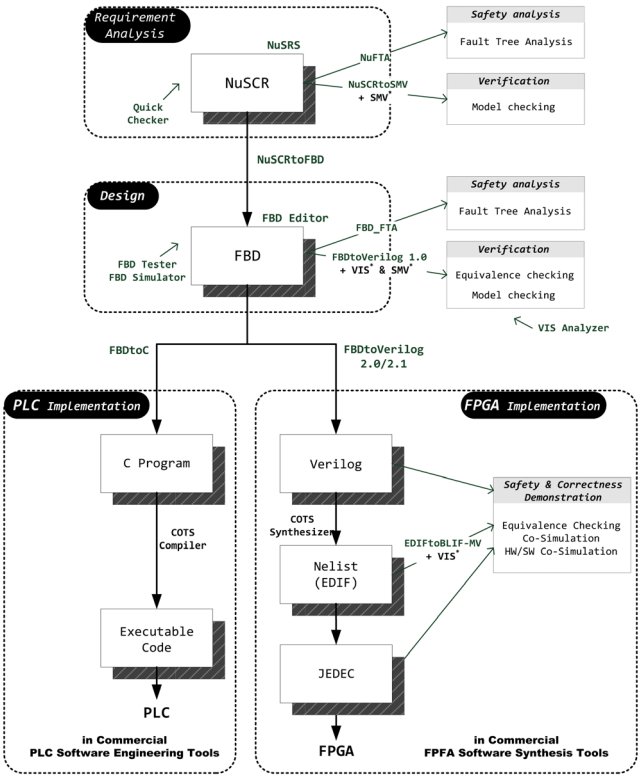

Fig. 3. An overview of the *NuDE* 2.0 framework

III. The *NuDE 2.0*

The *NuDE 2.0* (Nuclear Development Environment) is a formal model-based software development environment, specialized for safety-critical digital systems in nuclear power plants. It can develop the software implemented with safety-grade hardware platforms such as PLC and FPGA, seamlessly and simultaneously. The *NuDE* was originally developed for the PLC platform, but now being extended for FPGA. It can generate implementation codes for PLC and FPGA simultaneously from the same NuSCR requirements specification or FBD design program. It also encompasses the whole SDLC (Software Development Life-Cycle) while seamlessly supporting various formal verification and safety

analysis as well as the MBD (Model-Based Development)-based code generation. Fig.3 depicts the whole process in details, and the following subsections briefly explain each phase around supporting tools.

A. The Requirements Analysis Phase

Fig.4 is an example of the *NuSCR* specification modeled in '*NuSRS 2.0*.' *NuSCR* [7] is a data-flow based requirements specification language, specialized for the safety-critical systems in the nuclear domain. The NuSCR modeling environment, *NuSRS 2.0*, includes static grammar checker '*Quick Checker*' and the '*NuSCRtoSMV*' [14] translator to generate the SMV input program and execute the Cadence SMV model checker [15], seamlessly. '*NuFTA*' [16] also generates software fault trees for the NuSCR specification mechanically. The NuSCR formal requirements specification is then translated into a behaviorally-equivalent FBD program by '*NuSCRtoFBD* 4.0' [17].

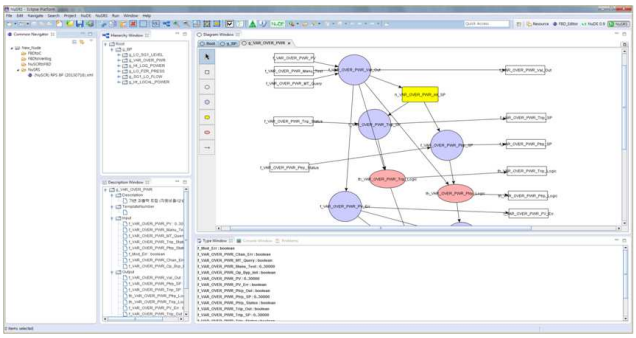

Fig. 4. An NuSCR formal specification modeled in NuSRS 2.0

B. The Design Phase

'*FBD Editor*' in Fig.5 shows the FBD program, which is mechanically translated from an NuSCR specification, through *NuSCRtoFBD 4.0*. We can also model it directly on the tool [18]. '*FBD Simulator*' executes an FBD program with predefined inputs or randomly, while '*FBD Tester*' [19] enables us to do test the FBD programs directly with data-flow based coverage criteria for FBDs. Formal verification with the VIS verification system [20] and the SMV model checker is also possible through the '*FBDtoVerilog 1.0*' translator [21]. Since thhe FBD design phase often include hardware-dependent modifications on the FBDs, the formal verification are required additionally. The *NuDE* also provides '*VIS Analyzer*' [22] to assist the VIS verification graphically and seamlessly. '*FBD FTA*' is a fault tree generation and analysis tool for FBD programs.

The FBD program modeled in the '*FBD Editor*' can be transformed into different implementation codes for PLC and FPGA. '*FBDtoC*' [23] translates FBDs into behaviorally-equivalent C programs for PLC, while '*FBDtoVerilog 2.0/2.1*' [24], [25] transforms FBDs into Verilog programs for FPGA. We are working on the transformation from FBDs into VHDL programs.

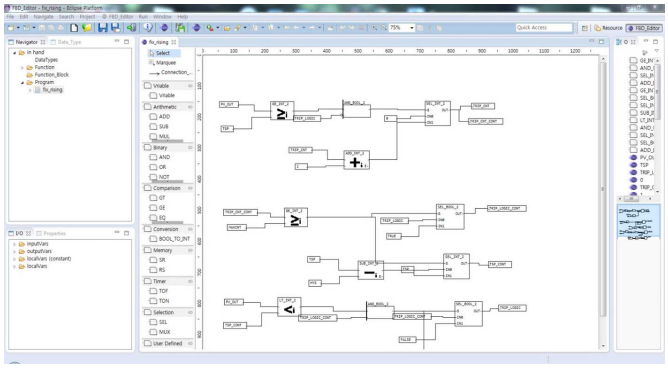

Fig. 5. An example FBD program in '*FBD Editor*'

C. The PLC Implementation Phase

The C programs transformed by the '*FBDtoC*' can be compiled into executable codes for a specific target PLC. Most commercial software engineering tools, however, translate FBDs into equivalent C and executable codes subsequently, and also download them into specific target PLCs. Most PLC vendors typically use COTS (Commercial Off-the-Shelf) software such as '*TMS320C55x*' of Texas Instruments for the C compilers. The COTS compilers were well verified and certified enough to be used without additional verification effort. However, the vendor-provided automatic translators from FBD to C should demonstrate its functional safety and correctness rigorously, as we proposed in [26].

D. The FPGA Implementation Phase

The Verilog program translated by *FBDtoVerilog 2.0/2.1* is the starting point of the fully-automated FPGA synthesis procedure provided by commercial tools. On the other hand, nuclear regulation authorities require more considerate demonstration of the correctness and even safety of the mechanical synthesis processes of FPGA synthesis tools, even if the FPGA industry have acknowledged them empirically as correct and safe processes and tools. While the synthesis process can be formally verified with the compiler verification techniques [27], [28], it is hard to apply them to the works of 3rd-party developers. It must be the most important obstacle for FPGAs to be used as a new platform of nuclear I&C systems. We are trying to overcome the obstacle through the safety and correctness demonstration technique proposed in [25].

E. Auxiliary Support for the Compiler Verification

The formal verification of compiler, translator and synthesizer is an important issue, and should be fully

demonstrated whenever new PLC compilers or FPGA synthesis tools are proposed to use to develop new safety-critical digital systems in nuclear power plants. These are typically developed by 3rd-parties, and we have no information to perform the in-depth analysis on them with typical compiler verification techniques. We have proposed an indirect demonstration technique [25], which uses the VIS equivalence checking and (HW/SW) co-simulation [29]. It is our current on-going research issue.

IV. Conclusion and Future Work

This paper introduces the new version of *NuDE* framework, briefly. It extends to corporate the FPGA-based software development as well as the PLC-based software, seamlessly and simultaneously. While various formal verification and safety analysis are also supported, we are now trying to systematically demonstrate the safety and correctness of various translators of our own and commercial synthesis tools of FPGA vendors. We are also preparing a full-scale case study, starting from a formal requirements specification and an FBD program in order to demonstrate the usefulness of the framework.

Acknowledgment

This research was supported, in part, by a grant from the Korea Ministry of Science, ICT and Future Planning, under the development of the integrated framework of I&C dependability assessment, monitoring, and response for nuclear facilities. It was also supported, in part, by a grant from the Korea Atomic Energy Research Institute, under the development of the core software technologies of the integrated development environment for FPGA-based controllers.

References

[1] J. She, "Investigation on the benefits of safety margin improvement in candu nuclear power plant using an fpga-based shutdown system," Ph.D. dissertation, The University of Western Ontario, 2012.

[2] J.-G. Choi, "Survey of the CPLD/FPGA Technology for Application to NPP Digital I&C System," Korea Atomic Energy Research Institute, Tech. Rep., 2009.

[3] 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.

[4] International Electrotechnical Commission, "International standard for programmable controllers: Programming languages," 1993, 61131-3.

[5] J. Yoo, E. Jee, and S. S. Cha, "Formal Modeling and Verification of Safety-Critical Software," *IEEE Software*, vol. 26, no. 3, pp. 42–49, May/June 2009.

[6] J.-H. Lee and J. Yoo, "NuDE: Development Environment for Safety-Critical Software of Nuclear Power Plant," in *Transactions of the Korean Nuclear Society Spring Meeting 2012*, 2012, pp. 1154–1155.

[7] J. Yoo, T. Kim, S. Cha, J.-S. Lee, and H. S. Son, "A Formal Software Requirements Specification Method for Digital Nuclear Plants Protection Systems," *Journal of Systems and Software*, vol. 74, no. 1, pp. 73–83, 2005.

[8] SIEMENS, "SPACE, engineering system of Teleperm XS PLC," Germany, Tech. Rep. KWU NLL1-1026-76-V1.0/11.96, 1996.

[9] SIEMENS , "TELEPERM XS, brief description," Germany, Tech. Rep. KWU NLL1-1004-76-V2.2/04.98, 1998.

[10] invensys, "Safety software suite, TriStation 1131 (TS1131)," http://iom.invensys.com/.

[11] KNICS, "Korea nuclear instrumentation and control system r&d center," http://www.knics.re.kr/english/eindex.html.

[12] S. Cho, K. Koo, B. You, T.-W. Kim, T. Shim, , and J. Lee, "Development of the loader software for PLC programming," in *Conference of the Institute of Electronics Engineers of Korea*, vol. 30, 2007, pp. 959–960.

[13] S. Brown and J. Rose, "FPGA and CPLD architectures: A tutorial," *IEEE Software*, vol. 13, no. 2, pp. 42–57, 1996.

[14] E. Jee, S. Jeon, S. Cha, K. Koh, J. Yoo, G. Park, and P. Seong, "FBD Verifier: Interactive and Visual Analysis of Counterexample in Formal Verification of Function Block Diagram," *Journal of Research and Practice in Information Technology*, vol. 42, no. 3, pp. 255–272, August 2010.

[15] K. McMillan, "Cadence SMV," *http://www.kenmcmil.com*.

[16] S. Yun, D.-A. Lee, and J. Yoo, "NuFTA: A case tool for automatic software fault tree analysis," in *Transactions of the Korean Nuclear Society Spring Meeting 2010*, 2010.

[17] J. Yoo, S. Cha, C. H. Kim, and D. Y. Song, "Synthesis of FBD-based PLC Design from NuSCR Formal Specification," *Reliability Engineering and System Safety*, vol. 87, no. 2, pp. 287–294, 2005.

[18] D.-A. Lee, E.-S. Kim, Y.-J. Seo, and J. Yoo, "FBDEditor: An FBD Design Program for developing Nuclear Digital I&C Systems," in *Korea Conference on Software Engineering (KCSE 2014)*, 2014, pp. 315–318, in Korean.

[19] E. Jee, J. Yoo, S. Cha, , and D. Bae, "A Data Flow-based Structural Testing Technique for FBD Programs," *Information and Software Technology*, vol. 51, no. 7, pp. 1131–1139, 2009.

[20] R. K. Brayton, G. D. Hachtel, A. Sangiovanni-Vincentelli, F. Somenzi, A. Aziz, S.-T. Cheng, S. A. Edwards, S. P. Khatri, Y. Kukimoto, A. Pardo, S. Qadeer, R. K. Ranjan, S. Sarwary, T. R. Shiple, G. Swamy, and T. Villa, "VIS : A system for verification and synthesis," in *the Eighth International Conference on Computer Aided Verification, CAV '96*, 1996, pp. 428–432.

[21] J. Yoo, S. Cha, and E. Jee, "Verification of PLC Programs Written in FBD with VIS," *Nuclear Engineering and Technology*, vol. 41, no. 1, pp. 79–90, 2009.

[22] S. Jung, J. Yoo, and S. Cha, "VIS Analyzer : A Visual Assistant for VIS Verification and Analysis," in *The 13th IEEE Computer Society symposium dealing with the rapidly expanding field of object/component/service-oriented real-time distributed computing (ORC) technology, ISORC 2010 Symposium*, 2010.

[23] J. Yoo, E.-S. Kim, and J.-S. Lee, "A Behavior-Preserving Translation from FBD Design to C Implementation for Reactor Protection System Software," *Nuclear Engineering and Technology*, vol. 45, no. 4, pp. 489–504, 2013.

[24] D.-A. Lee, E.-S. Kim, and J. Yoo, "FBDtoVerilog 2.0: An automatic translation of FBD into Verilog to develop FPGA," in *5th International Conference on Information Science and Application (ICISA 2014)*, 2014, pp. 447–450.

[25] E.-S. Kim, , J. Yoo, J.-G. Choi, J.-Y. Kim, and J.-S. Lee, "A Correctness Verification Technique for Commercial FPGA Synthesis Tools," in *the 2nd International Workshop on Assurance Cases for Software-intensive Systems (ASSURE 2014)*, 2014, co-located with ISSRE 2014.

[26] D.-A. Lee, J. Yoo, and J.-S. Lee, "A Systematic Verification of Behavioral Consistency between FBD Design and ANSI-C Implementation Using HW-CBMC," *Reliability Engineering and System Safety*, vol. 120, no. 12, pp. 139–149, 2013.

[27] T. Hoare, "The verifying compiler: A grand challenge for computing research," *Journal of the ACM*, vol. 50, no. 1, pp. 63–69, 2003.

[28] X. Leroy, "Formal Verification of a Realistic Compiler," *Communication of the ACM*, vol. 52, no. 7, pp. 107–115, 2000.

[29] S. Sicklinger, V. Belsky, B. Engelmann, H. Elmqvist, H. Olsson, R. Wuchner, and K.-U. Bletzinger, "Interface jacobian-based co-simulation," *International Journal for Numerical Methods in Engineering*, vol. 98, no. 6, pp. 414–444, 2014.