Transactions of the Korean Nuclear Society Spring Meeting
Chuncheon, Korea, May 25-26 2006

High Speed Simulation Framework for Reliable Logic Programs

Wan-Bok Lee,a Seog-Ju Kim,b
a Dep. of Computer Game, Joongbu Univ., Kumsan-gun, Chungnam, wblee@joongbu.ac.kr
b Korea Electrotechnology Research Institute, Changwon-city, Kyongnam, sjkim@keri.re.kr

1. Introduction

This paper shows a case study of designing a PLC logic simulator that was developed to simulate and verify PLC control programs for nuclear plant systems. The nuclear control system requires strict restrictions rather than normal process control system does, since it works with nuclear power plants requiring high reliability under severe environment. One restriction is the safeness of the control programs which can be assured by exploiting severe testing. Another restriction is the simulation speed of the control programs, that should be fast enough to control multi devices concurrently in real-time. To cope with these restrictions, we devised a logic compiler which generates C-code programs from given PLC logic programs. Once the logic program was translated into C-code, the program could be analyzed by conventional software analysis tools and could be used to construct a fast logic simulator after cross-compiling, in fact, that is a kind of compiled-code simulation.[1]

2. Simulation Framework in pSET Authoring Tool

Safety grade PLC software for nuclear power plants can be developed using an authoring tool, so-called pSET, which was developed by KERI and POSTECH. Currently, the pSET authoring tool supports a systematic design scheme of control program in a visual environment. However, it still does not provide simulation facility which can be a very helpful element for users to find logical errors and debug the program.

Simulation can be regarded as an analysis process of models using a computer and is frequently used for performance evaluation or debugging purpose. General commercial PLC software vendors tend to provide simulation toolkit which execute the PLC logic at Rung or scan time unit. However, the toolkit is weak and infeasible to verify or test the control program. The control program which will be loaded on safety grade PLC must be verified or tested severely before its real adaptation. That is, just simulation is not sufficient in case of safety grade PLC simulator.

To solve this problem, we designed a new PLC simulator which is suitable to develop safety grade PLC software. The basic idea in our approach is that the PLC control program is transformed to C-code first. The transformed C-code model can be used both for verification and simulation. To ensure correct transformation of the logic program we implemented a logic compiler, named as LD/FBD compiler, using the general authoring tools such as Lex and Yacc.

As PLC control program is developed on a graphic development environment, pSET, a new specification for control logic programs should be defined such that the LD/FBD compiler can understand. Thus, a PLC logic specification language, so-called LD/FBD GL, was defined in BNF format. A control program designed in pSET editor can be saved in a form of the LD/FBD GL language, which in turn is transformed to C-code specification by LD/FBD compiler.

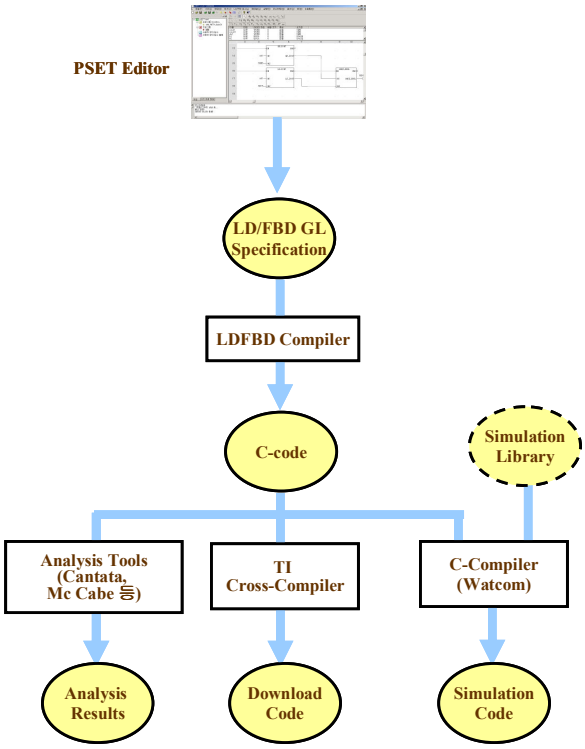

Figure 1. Simulation Framework in pSET

Simulation execution codes are generated following the process flow as shown in Fig. 1. The C-code specification generated in the process can be used in three phases; firstly, it can be used to evaluate safeness of the control program using commercial testing and analysis tools like Mc Cabe[2] or Cantata[3]. Secondly, the C-code can be transformed to the simulation execution code just by compiling and/or linking with simulation library. Several functions are provided in the simulation library such as execution run control, appointment of break points, and identification and modification of a specified IO port or memory content. Thirdly, the C-code can be transformed to a download code that can be executed on a target PLC after cross-

compiling. In contrast to commercial PLCs, the download code can be executed in a high speed since the code is consisted of the instructions of microprocessor embedded in the target PLC hardware. As a result, there is no run-time interpretation process at the PLC side and it is the key point which increases the execution speed.

By these approaches, the logic program simulator for nuclear power plants was designed. It is in fact a seamless development environment providing three features altogether such as the safeness evaluation, and performance estimation, and the generation of the final download code. Moreover, execution speed of the control program was significantly increased as PLC control program is pre-interpreted by LD/FBD compiler. That method can be regarded as a compiled code simulation method that increases simulation speed by several decades when applied to the currently-used Cad programs [1].

3. PLC Logic Program Compiler

Transforming the PLC logic program to C code, the logic should be represented in a text format, called LD/FBD GL (LD/FBD Graphic Language) that is necessary for LD/FBD compiler to find logical error and to understand the dynamics of the logic program. LD/FBD GL compiler detects various errors of PLC logic program. Those errors can be classified into three groups.

- Lexical Error
  - Detected by Lex
  - Eg: wrong identifier, wrong variables, etc.
- Syntax Error
  - Wrong connection of nodes
  - Detection of cycle in LD logic
- Logic Error
  - Wrong connection of nodes
  - Detection of cycle in LD logic
  - Incoincidence of data type
  - Maldefinition of data

4. Conclusions

C-code conversion of logic programs is advantageous in that it can be used in various part of development process. But, the long build time to get the final simulation code may bother the logic designer.

At present, the simulator developed supports LD and FBD out of 5 PLC programming languages defined in the IEC standard[4]. Hereafter, the simulator needs to be complemented to support SFC.

REFERENCES

[1] D. Lewis, " A Hierarchical Compiled Code Event-Driven Logic Simulator", *IEEE Trans. CADICS*, 10:726-737. 1991.

[2] T.J. Mc Cabe, "A complexity measure", *IEEE Trans. on Software Engineering*, 2(4):308-320, 1976.

[3] ---, Testing C with Cantata++, http://www.ipl.com.

[4] ---, IEC 61131-3 International Standard Part 3: Programming languages, International Electro-technical Commission, 1993.