ISSN 2383-630X(Print) / ISSN 2383-6296(Online)
Journal of KIISE, Vol. 43, No. 5, pp. 569-578, 2016. 5
http://dx.doi.org/10.5626/JOK.2016.43.5.569

FBDtoVHDL: FPGA 개발을 위한 FBD에서 VHDL로의 자동 변환

(FBDtoVHDL: An Automatic Translation from FBD into VHDL for FPGA Development)

김 재 엽 † 김 의 섭 †† 유 준 범 ††† 이 영 준 †††† 최 종 균 †††††
(Jaeyeob Kim) (Eui-Sub Kim) (Junbeom Yoo) (Young Jun Lee) (Jong-Gyun Choi)

요 약 PLC (Programmable Logic Controller)는 원자력 발전소의 디지털 제어시스템의 개발을 위해 널리 사용되어왔지만 복잡성의 증가와 유지보수 비용 등의 문제로 인해 FPGA (Field Programmable Gate Array) 기반 제어시스템이 대안으로 떠오르고 있다. 하지만 PLC 개발자가 FPGA 기반 제어시스템을 개발하기 위해서는 FPGA 개발을 위한 언어를 사용해야 하고 기존의 PLC 개발에서 획득한 노하우 및 지식의 재사용을 어렵게 만든다는 등의 문제가 발생한다. 본 논문에서는 이와 같은 문제를 해결하기 위해서 PLC 소프트웨어 개발을 위한 언어 중 하나인 FBD (Function Block Diagram)를 FPGA 개발을 위한 하드웨어 기술 언어 중 하나인 VHDL로의 자동 변환을 위한 방법과 이를 기반으로 개발한 자동 변환 도구인 FBDtoVHDL을 소개한다. 본 연구에서 소개하는 FBDtoVHDL 도구를 사용하여 FBD를 VHDL로 자동 변환함으로써 PLC 개발자는 하드웨어 기술 언어에 대한 지식이 없이도 FPGA 개발하는 것이 가능하다.

키워드: PLC, FPGA, FBD, VHDL, 자동변환기

Abstract The PLC (Programmable Logic Controller) has been widely used for the development of digital control system of nuclear power plant. The PLC has high maintenance costs and increasing complexity, hence, FPGA (Field Programmable Gate Array) based digital control system has been considered as an alternative. However, the development of FPGA based digital control system is a challenge for PLC engineers because they are required to learn about new language to develop FPGA and knowledge and know-how acquired in the development of PLC is not transferable. In this study, we proposed and implemented an automatic translation tool for translation of FBD (Function Block Diagram), a programming language of PLC software, into VHDL (VHSIC Hardware Description Language). Automatically translating the FBD to VHDL using this tool allows PLC engineers to develop FPGA without any knowledge of the hardware description language.

Keywords: PLC, FPGA, FBD, VHDL, automatic translation tool

· 본 연구는 한국원자력연구원의 "FPGA-기반 제어기 통합개발환경을 위한 핵심 소프트웨어 기술 개발" 사업과 "원자력 계측제어 계통 안전 적합성 평가체계" 사업의 지원으로 연구한 결과 입니다.

† 비 회 원 : 건국대학교 컴퓨터 정보통신공학
radic2510@gmail.com
†† 학생회원 : 건국대학교 컴퓨터 정보통신공학
atang34@konkuk.ac.kr
††† 정 회 원 : 건국대학교 컴퓨터공학부 교수(Konkuk Univ.)
jbyoo@konkuk.ac.kr
(Corresponding author임)
†††† 정 회 원 : 한국원자력연구원 MMIS Lab.
yjlee426@kaeri.re.kr
††††† 비 회 원 : 한국원자력연구원 MMIS Lab.
choijg@kaeri.re.kr

논문접수 : 2015년 11월 12일
(Received 12 November 2015)
논문수정 : 2016년 3월 7일
(Revised 7 March 2016)
심사완료 : 2016년 3월 11일
(Accepted 11 March 2016)

정보과학회논문지 제43권 제5호(2016. 5)

1. 서 론

원자력 발전소의 안전계통이 아날로그 기반의 I&C (Instrumentation and Control) 시스템에서 디지털 기반의 I&C로 전환되면서 소프트웨어와 네트워크가 시스템의 일부가 되었다. 예를 들면 안전계통 시스템 중 하나인 RPS (Reactor Protection System)의 경우 FBD (Function Block Diagram), LD (Ladder Diagram)와 같은 PLC 프로그래밍 언어[1]로 작성된 소프트웨어 사용하며 네트워크를 통해 다른 장비들과 통신한다. 디지털 시스템으로의 전환을 통해 신뢰도와 더 나은 성능을 제공하게 되었지만[2] 소프트웨어 등의 디지털 기술이 사용되면서 공통원인고장의 발생 가능성과 유지보수 비용, 복잡성의 증가가 문제로 떠올랐다.

하드웨어 기반의 FPGA (Field Programmable Gate Array)를 이용한 다양성 확보는 공통원인고장으로부터 RPS를 보호하기 위한 방법 중 하나이다[3,4]. FPGA는 하드웨어 기술 언어를 사용해 시스템을 개발해야 하지만 PLC 기반 언어와는 다르고 새로운 개발 프로세스를 따라 개발해야 한다. 따라서 PLC 기반의 시스템을 개발하던 엔지니어가 FPGA 기반 시스템을 개발하는 것은 새로운 언어를 습득해야 하는 어려움과 PLC 기반의 경험과 지식 등을 이용하지 못한다는 문제점이 있다.

본 논문에서는 PLC 소프트웨어 개발을 위한 언어 중 하나인 FBD로부터 하드웨어 기술 언어 중 하나인 VHDL로의 변환을 소개한다. FBD를 VHDL로 변환함으로써 PLC 개발자는 하드웨어 기술 언어에 대한 지식이 없이도 FPGA를 개발하는 것이 가능할 것이다. 또한 PLC 소프트웨어에 적용되던 모든 V&V와 안전성 분석이 FPGA 개발에서도 유효하게 될 것이다. 우리는 변환 규칙을 기반으로 구현한 자동 변환 도구인 FBDtoVHDL을 소개하고 이를 통해 변환한 VHDL이 FBD와 같은 기능을 수행하는지 확인하기 위해 KNICS RPS BP로직을 이용한 실험을 수행하였다.

본 논문의 구성은 다음과 같다. 2장에서 FBD와 VHDL에 대한 소개를 간략하게 하고 3장에서 FBD를 VHDL로 변환하기 위한 규칙과 자동 변환 도구인 FBDtoVHDL을 소개한다. 4장에서는 사례 연구를 다루고 마지막 5장에서 결론을 논한다.

2. 배경 지식

2.1 FBD (Function Block Diagram)

FBD는 IEC 61131-3 표준[1]에 정의된 5가지 PLC 프로그래밍 언어 중 하나이다. FBD는 그래픽 기반의 표현이 가능한 언어로 각각의 기능을 수행하는 FB (Function Block)들과 이들의 연결로 표현이 된다. FB는 사각형으로 묘사되고 입력/출력 값과 연결되며 각 FB는 산술연산, 논리연산, 비교연산, 선택연산, 시간연산 등의 연산을 수행한다. 각 FB는 실행 순서가 부여되고 선으로 연결되어 절차적인 흐름을 표현하게 된다.

그림 1은 FBD로 구현된 KNICS RPS BP의 FIX_FALLING[5]의 로직 중 일부이다. 블록 위쪽 괄호가 있는 것이 FB이며 나머지 부분은 값에 해당한다. 괄호 안의 숫자는 블록의 실행 순서로 낮은 숫자부터 높은 숫자 순으로 실행된다. 예를 들어 31번째 실행 순서를 가지는 AND_BOOL_2 블록은 29번째와 30번째 실행 순서를 가지는 LT_INT_2가 실행되면 그 출력을 입력으로 하여 연산을 수행한다. 32번째 실행 순서를 가지는 블록의 IN1의 ○는 NOT에 해당하는 기호이다. 그림 1은 이전 실행순서의 FB에서 연산된 값인 cond_final_(P)TRIP_LOGIC의 값과 에러 값, 상태 값을 고려해서 출력 값인 (P)TRIP을 결정하는 기능을 수행한다.

2.2 VHDL(VHSIC Hardware Description Language)

VHDL은 IC(Integrated Circuit) 개발자들에게 널리 사용되는 하드웨어 기술 언어 중 하나로 IEEE 표준 언어이다[6]. VHDL을 이용한 설계는 시스템 수준부터 게이트 수준까지 표현하는 것이 가능하기 때문에 넓은 범위의 설계가 가능하다는 장점이 있다. 또한 VHDL을 이용한 하향식 설계가 가능하기 때문에 설계에서 발생할 수 있는 에러를 초기에 발견하는 것이 가능하고 이를 이용하여 설계에 필요한 비용을 최소화할 수 있다. 다수의 EDA(Electronic Design Automation) 도구들이 VHDL을 이용하고 합성 과정을 지원하기 때문에 FPGA 장비 또는 반도체 칩 제조 과정 등과는 무관하게 회로 설계에만 집중할 수 있다는 장점이 있다.

3. FBDtoVHDL

FBDtoVHDL은 기존의 PLC 기반 제어기를 개발하기 위해 사용된 언어 중 하나인 FBD를 FPGA 기반 제어기 개발을 위한 언어 중 하나인 VHDL로 자동 변환하는 도구이다. 우리는 이를 위해서 FBD를 VHDL로 변환하는 규칙을 정의하고 변환 규칙을 VHDL 형태로 만드는 변환 결과 프레임을 만들었으며, 이를 토대로 자동 변환 도구를 구현하였다.

3.1 변환 규칙

IEC 61131-3 표준에서 Function과 Function Block (FB), Program 모두 POU(Programmable Organization Unit)로 정의되어 있으며 Function은 Function을, FB는 FB와 Function을, Program은 Program과 FB, Function을 포함할 수 있다고 정의되어 있다. 우리는 그 중 Program을 POU의 단위로 하여 하나의 FBD는 POU의 계층구조로 이루어져 있다고 보고 변환 규칙을

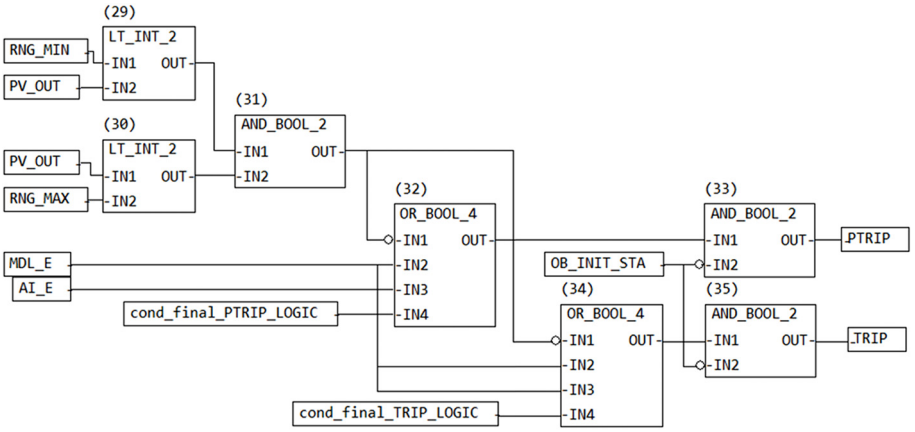

그림 1 FBD로 구현된 FIX_FALLING 로직의 일부
Fig. 1 A part of the FIX FALLING logic implemented in FBD

<Interface>
1. for each input/output variable in FBD – *INPUT*| *OUTPUT*
   = [variable name] : in/out/buffer [data type]
   * buffer is used for feedback variable
2. for each local variable in FBD – *CONSTANT*
   = signal [variable name] : [data type] := [initial value]
3. for each connection/continuation variable in FBD – *CON*
   = signal [continuation name] : [data type]

<Body>
4. for executing all FB
   = [FB name]_[local id] : entity work. [entity] port map
   ([port name* => *INPOUT*| *OUTPUT*| *CONSTANT*| *CON*|)+
   * : Port name is the other entity's *INPUT* or *OUTPUT*
   + : This symbol means that elements can be repeated
5. for all assignment output variables
   = [*COM*| *OUTPUT*] <= [*CONSTANT*| *COM*]
6. for initialization output variables
   = [*OUTPUT(buffer)*] <= [initial value]
7. for all feedback assignment
   = [*OUTPUT(buffer)*] <= [*CONSTANTS*| *COM*]

그림 2 FBD를 VHDL로의 변환 규칙
Fig. 2 A translation rule of FBD into VHDL

정의하였다. VHDL 언어의 특성을 고려하였을 때 하나의 VHDL 파일에 하나의 Program이 변환될 수 있기 때문이다. 하나의 POU는 입력과 출력 등의 정보를 가지는 인터페이스 부분과 기능에 해당하는 바디 부분으로 구성되어 있다.

우리는 FBD를 7부분으로 나눠 FBD를 VHDL로 변환하는 규칙을 정의하였으며 정의한 규칙은 그림 2와 같다. 1~3번 규칙은 POU의 인터페이스 부분을 4~7번 규칙은 POU의 바디 부분을 VHDL에 적합하게 변환하는 규칙이다.

규칙 1은 변환 중인 FBD의 외부 입출력 값을 VHDL 형태로 선언하는 규칙이다. 외부 입출력을 입력, 출력, 피드백인 경우로 구분하여 in, out, buffer를 선택하고 FBD의 값의 이름과 데이터 타입을 이용하여 규칙에 맞게 VHDL 형태로 변환한다. VHDL의 데이터 타입은 크기를 자유롭게 선언하는 것이 가능하기 때문에 FBD에서 사용한 데이터타입과 가장 유사한 데이터 타입으로 선언한다. 예를 들면 1bit인 데이터 타입은 VHDL에서 std_logic을 사용하여 선언하는 것이 가능하고 integer 등을 이용하여 다양한 데이터 타입을 선언한다. 피드백은 출력이 다음 실행 시 입력으로 사용되는 값을 의미하며 이는 buffer로 선언하여 입력과 출력으로 사용하는 것이 가능하다.

규칙 2는 FBD의 상수 값을 변환하는 규칙이다. 상수의 이름과 데이터 타입, 초기 값을 이용하여 signal로 선언한다.

규칙 3은 FBD에서 CON (connection/continuation)을 변환하는 규칙으로 POU에서 생성되는 출력 값을 signal에 저장하여 다른 POU의 입력이나 출력 값에 할당하는 용도로 사용할 수 있도록 하는 부분이다. Signal의 이름은 continuation에 해당하는 이름을 사용하고 유사한 데이터 타입을 사용하여 선언한다.

규칙 4는 FB에 해당하는 부분을 변환하는 규칙이다. VHDL에서는 하나의 POU를 하나의 entity로 변환하기 때문에 특정 FB를 호출하는 부분 또한 entity를 호출하는 형식으로 변환이 수행된다. 하나의 FB가 중복적으로 호출될 수 있기 때문에 호출하는 FB의 이름과 해당 FB의 localid의 조합하여 entity 호출의 선언에 사용하고 호출하는 entity의 port에 입력이나 출력, 상수, CON을 연결한다. port는 entity의 입력과 출력을 의미하며 entity를 호출하기 위해서는 모든 포트에 연결된 값이 있어야 한다. +기호는 반복될 수 있음을 의미하는 기호로 모든 port에 값이 연결되어야 함을 의미한다.

규칙 5는 값을 할당하는 부분을 변환하는 규칙으로 FBD의 출력으로 값을 할당하는 것과 CON에 값을 할당하는 경우가 이에 속한다. 모든 POU의 출력이 CON이 가지고 있기 때문에 CON 또는 상수 값이 CON이나 출력에 할당된다.

규칙 6과 **규칙 7**은 FBD의 피드백과 관련된 부분을 VHDL에서 동일하게 수행하도록 할 수 있게 변환하는 부분이다. FBD 에서 피드백에 해당하는 부분은 초기화를 수행하고 전체 수행이 한번 끝난 후 입력으로 사용된다. 이를 VHDL에서 이와 동일한 동작을 수행하도록 변환하기 위해서 **규칙 6**에서 피드백을 포함한 모든 출력에 초기 값을 할당하고 **규칙 7**에서 피드백 부분에 값을 할당하도록 한다.

3.2 변환 결과 프레임

FBD를 VHDL로 변환하여 개발을 수행하기 위해서는 변환규칙을 이용한 변환 결과 외에 VHDL의 기본적인 구조가 필요하다. 따라서 우리는 변환 규칙을 포함하는 VHDL 변환 결과 프레임을 만들었으며 이는 그림 3과 같다.

```
1: entity [POUName] is (
2:  generic ();
3:  port (
4:      clk : in std_logic;
5:      rst : in std_logic;
6:      pulse : in std_logic;
7:
8:      [[input Name] : in [Data Type;]] *              -- INPUT
9:      [[output Name] : out [Data Type;]] *            -- OUTPUT
10:     [[output Name] : buffer [Data Type;]] *         -- FEEDBACK
11:  );
12: end [POUName];
13:
14: architecture Behavioral of [POUName] is
15:  [signal [constant Name] : [Data Type] := [initial Value;]] *
16:                                                     -- CONSTANT
17:  [signal [CON Name] : [Data Type;]]*
18:                                          -- Connect/Continuation (CON)
19:
20:  [[FBname_[local id]] : entity work.[entity] port map
21:   ([,[portName] => clk|rst|INPUT|CONSTANTS|CON])*;]]*
22:                                                     -- Entity Call
23:
24:  [[CON|OUTPUT] <= [CONSTANTS|CON;]]* -- Assignment
25:
26:  process(clk, rst) begin
27:    if(rst = '1') then
28:      [[OUTPUT|FEEDBACK] <= [initialValue;]]*
29:                                                     -- Output Initialization
30:    elseif(pulse = '1') then
31:      [[FFEDBACK] <= [CONSTANTS|CON;]]*
32:                                                     -- Feedback Assignment
33:    end if;
34:  end process;
35: end Behavioral;
```

그림 3 VHDL 변환 결과 프레임

Fig. 3 A translation result frame of VHDL

VHDL은 크게 entity와 architecture로 선언된다. entity는 외부에서 볼 수 있는 입력과 출력에 해당하는 포트를 선언하는 부분이며 architecture는 내부에서 사용되는 값을 선언하는 부분과 호출과 할당 등의 기능이 선언된다. 그림 3에서 entity와 architecture는 POU의 이름으로 선언된다. entity 내의 generic은 특정 값을 정의하는 부분으로 사용된다. port에는 기본적으로 사용되는 3가지 입력(clk, rst, pulse)가 선언되고 규칙 1~2의 변환 결과가 포함된다. Architecture 부분에는 규칙 3~7의 변환 결과가 포함되며, 규칙 6과 규칙 7의 경우 VHDL의 process를 사용하게 된다. 초기화 신호인 rst를 확인하여 값이 1인 경우 규칙 6에 해당하는 초기화를 수행하도록 하며, 피드백을 할당하기 위한 값인 pulse를 확인하여 값이 1인 경우 규칙 7에 해당하는 상수나 CON을 피드백에 할당하도록 한다. 규칙을 통해 변환되는 부분은 +기호를 통해 반복될 수 있음을 의미한다.

그림 4는 그림 1의 FBD를 대상으로 변환 규칙으로 변환을 수행하고 VHDL 변환 결과 프레임에 적용한 결과 중 29번째 실행순서를 가지는 LT_INT_2와 관련된 부분과 특징적인 부분만을 변환 결과 프레임에 적용한

```
1: entity FIX_FALLING is (
2:  generic(
3:    INT_HI : integer := 8388607;
4:    INT_LO : integer := -8388608
5:  );
6:  port(
7:    clk : in std_logic;
8:    rst : in std_logic;
9:    pulse : in std_logic;
10:
11:    PV_OUT : in integer range INT_LO to INT_HI;
12:
13: end FIX_FALLING;
14:
15: architecture Behavioral of FIX_FALLING is
16:    signal RNG_MIN: integer range INT_LO to INT_HI := 600;
17:
18:    signal LT_INT_2_wire_29_OUT : std_logic;
19:
20:    LT_INT_2_29 : entity work.LT_INT_2 port map
21:     (clk => clk, rst => rst, A_i => RNG_MIN,
22:      B_i => PV_OUT, R_o=>LT_INT_2_wire_29_OUT);
23:
24:    TRIP <= AND_BOOL_2_wire_35_OUT;
25:
26:  process(clk, rst) begin
27:   if(rst = '1') then
28:      TRIP_LOGIC <= '0';
29:   elseif(pulse = '1') then
30:      TRIP_LOGIC <= SEL_BOOL_2_wire_27;
31:   end if;
32:  end process;
33: end Behavioral;
```

그림 4 FIX_FALLING의 일부를 변환한 결과

Fig. 4 A translation result of a partial of FIX_FALLING

결과이다. 변환 결과 프레임은 굵게 표현된 부분으로 VHDL의 기본 구조를 표현하기 위한 부분이다. 11번째 줄은 입력으로 사용되는 PV_OUT을 선언하는 부분이고 16번째 줄은 입력 값과 비교를 위해 사용되는 지역 변수인 RNG_MIN을 선언하는 부분이다. 18번째 줄은 LT_INT_2_29의 출력을 다음 FB에 연결하기 위해 사용할 시그널인 LT_INT_2_wire_29_OUT을 선언하는 부분이다. 20~22번째 줄은 LT_INT_2_29를 선언하고 연산을 수행하는 부분이다. 사전에 만들어 둔 LT_INT_2의 입력 포트에 RNG_MIN과 PV_OUT을 연결하고 출력 포트에 LT_INT_2_wire_29_OUT을 연결하여 29번째 실행 순서를 가지는 FB의 변환이 완료된다. FBD의 출력 중 하나인 TRIP에 값을 할당하는 부분은 24번째 줄로 35번째 실행 순서를 가지는 AND_BOOL_2의 출력 값을 TRIP에 할당하게 된다. 26~32번째 줄은 프로세스로 구현한 부분으로 rst의 값을 확인하여 출력 값을 초기화하고 pulse 값을 확인하여 피드백에 할당하는 동작을 수행하게 된다.

3.4 자동 변환 도구 구현 (FBDtoVHDL)

우리는 3.1장에서 설명한 변환 규칙을 이용하여 FBD를 FPGA 개발을 위한 VHDL로의 자동 변환을 수행하는 도구인 FBDtoVHDL을 구현하였다. 도구는 그림 5와 같으며 FBD를 읽고 변환이 가능한 형태인지 검증하는 기능과 검증을 통과한 FBD를 VHDL로의 변환하는 기능을 가지고 있다. 변환 결과는 그림 3의 변환 결과 프레임에 맞도록 생성한다. 또한 FBD를 읽을 경우 xml 형태의 FBD를 좌측 하단에 출력하고 변환이 완료되었을 경우 변환 결과를 우측 하단에 변환 결과를 출력한다.

FBD가 변환이 가능한 형태인지 검증하는 기능은 특정 스키마를 만족하는지 확인하는 것이다. FBDtoVHDL은 PLCopen TC6 XML 버전 2.01 스키마[7]를 만족하는 FBD를 대상으로 변환을 수행한다. 스키마를 만족하는 FBD가 입력일 경우 변환 규칙에 따른 변환을 수행하며 "*.vhd"의 확장자를 가지는 VHDL 파일로 지정한 폴더에 저장한다.

4. 사례 연구

우리는 KNICS RPS BP의 5개 주요 로직과 APR-1400 원자로 보호 시스템의 프로토타입 명세를 NuSCR로 기술하고 이를 NuSCRtoFBD[8]를 통해 FBD로 변환한 예제(g_BP)를 대상으로 FBDtoVHDL을 통한 변환을 수행하였으며 변환 전후의 일치성을 확인하기 위하여 다음과 같은 두 가지 분석을 수행하였다. 우선 FBD와 VHDL의 파일을 분석하여 변환 전후의 구조의 일치성을 확인하였으며, 각각의 시뮬레이션 통하여 출력한 결과를 분석하여 같은 기능을 수행하는지 확인하였다.

4.1 구조적 분석

실험 대상인 5개의 주요 로직은 각각 단일 POU로 구성되어 있고 g_BP는 POU의 계층구조로 구성되어 있다. 계층 구조인 g_BP는 상위, 중간, 하위 수준으로 구

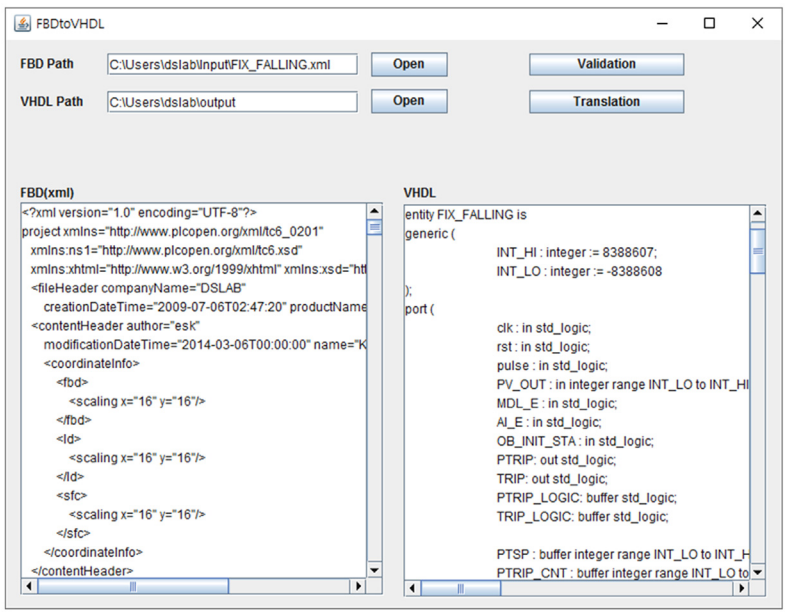

그림 5 FBDtoVHDL 자동 변환 도구

Fig. 5 An automatic translation tool FBDtoVHDL

성되어 있으며 상위 수준 POU인 g_BP는 6개의 중간 수준의 POU를 포함하고 있으며 중간 수준 POU는 다수의 하위 수준 POU를 포함하고 있다. 하위 수준의 POU는 다수의 입출력 값과 FB을 포함하고 있으며 이를 통한 연산을 수행하고 출력을 생산한다.

FBD와 VHDL가 구조적 일치성을 확인하기 위해서 우리는 변환규칙 1, 2, 4에 해당하는 입력과 출력, 상수, POU/FB 호출을 요소로 하여 확인과정을 수행하였다. 변환 규칙 3에 해당하는 connect/continuation은 FBD의 구조를 크게 변화시키지만 기능에는 영향을 주지 않는 부분이기 때문에 예외로 하여 고려하지 않았다. 표 1은 5개의 주요 로직을 분석한 결과이며 표 2는 g_BP에서 상위 수준과 중위 수준 POU와 중위 수준 POU 중 g_VAR_OVER_PWR의 하위 수준 POU를 분석한 결과이다. 두 표의 VHDL에서 입력의 개수는 변환 시 추가되는 기본적인 입력(clk, rst, pulse)을 제외한 개수이며 FBD의 POU와 FB는 VHDL에서 Entity Call에 대응한다.

표 1의 5개의 주요 로직은 입력과 출력, 상수의 개수가 변환 전후의 결과를 분석하여 차이가 없음을 확인하였으며 FB을 Entity Call로 변환한 결과 또한 차이가 없음을 확인하였다. 표 2에서 상위 수준과 중위 수준이 포함하는 POU가 변환 후의 Entity Call 대응한다는 점을 고려하면 변환 전후의 결과에서 구조적으로 차이가 없다는 것을 확인하였다. 또한 분석 결과를 통해 1:1 수준의 변환이 이루어졌다는 것을 확인할 수 있었다.

4.2 시뮬레이션을 통한 분석

기능적으로 변환이 옳바르게 이뤄졌는지 확인하기 위해 변환 전후의 5개의 주요 로직과 g_BP를 대상으로 시뮬레이션 수행하고 결과를 분석하였다. 기능적 일치성 확인을 위하여 IST-FPGA[9]의 Scenario Generator를 이용해 동일한 입력을 가지는 시나리오와 테스트벤치를 생성하였으며 FBD Simulaotr와 ModelSim으로 시뮬레이션을 수행하였다. 시뮬레이션 결과 비교는 자동화를 위해서 IST-FPGA 내의 Co-Simulator의 결과 비교 기능을 사용하였다.

시나리오와 테스트벤치는 POU의 주요 기능을 확인할 수 있는 입력을 가지도록 생성하였다. 표 3은 5개의 주요 로직 중 FIX_FALLING을 대상으로 시나리오 생성을

표 1 KNICS RPS BP 5개의 주요 로직을 대상으로 한 구조적 분석 결과
Table 1 Results of structural analysis for 5 main logics of KNICS RPS BP

	FED				VHDL			
POU Name	Input	Output	Constant	FB	Input	Output	Constant	Call
FIX_RISING	4	8	7	33	4	8	7	33
FIX_FALLING	4	8	7	33	4	8	7	33
MANUAL_RATE_FALLING	7	9	10	52	7	9	10	52
VARIABLE_RISING	4	10	10	49	4	10	10	49
VARIABLE_FALLING	4	10	10	49	4	10	10	49

표 2 APR-1400 원자로 보호 시스템의 프로토타입 예제를 대상으로 한 구조적 분석 결과
Table 2 Results of structural analysis for example of a prototype of APR-1400 NPP protection system

	FBD					VHDL			
POU Name	Input	Output	Constant	POU	FB	Input	Output	Constant	Entity Call
g_BP	60	43	-	6	-	60	43	-	6
1. g_VAR_OVER_PWR	8	8	-	9	-	8	8	-	9
1) f_VAR_OVER_PWR_Val_Out	3	1	2	-	3	3	1	2	3
2) h_VAR_OVER_PWR_Int_SP	1	1	14	-	158	1	1	14	158
3) f_VAR_OVER_PWR_Trip_SP	2	1	3	-	4	2	1	3	4
4) f_VAR_OVER_PWR_Ptrp_SP	2	1	4	-	5	2	1	4	5
5) th_VAR_OVER_PWR_Trip_Logic	2	1	9	-	21	2	1	9	21
6) th_VAR_OVER_PWR_Ptrp_Logic	2	1	9	-	21	2	1	9	21
7) f_VAR_OVER_PWR_PV_Err	1	1	4	-	4	1	1	4	4
8) f_VAR_OVER_PWR_Trip_Out	5	1	2	-	13	5	1	2	13
9) f_VAR_OVER_PWR_Ptrp_Out	5	1	2	-	13	5	1	2	13
2. g_LO_SG1_LEVEL	10	6	-	6	-	10	6	-	6
3. g_HI_LOG_POWER	12	8	-	8	-	12	8	-	8
4. g_LO_PZR_PRESS	11	9	-	12	-	11	9	-	12
5. g_SG1_LO_FLOW	8	8	-	9	-	8	8	-	9
6. g_HI_LOCAL_POWER	16	4	-	4	-	16	4	-	4

표 3 FIX_FALLING을 대상으로 한 실험 설정 및 결과
Table 3 An experimental setup and result of FIX_FALLING

	FIX_FALLING	Meaning of value
Initial Value	14,000	PTSP = 13920
Rate of Change	10-100 (Stepwise: 10)	To confirm the various cases
Error Value	0	To observe the production of (P)TRIP by (P)TSP
Number of Input	100	To confirm the number of results
Number of Scenario	1000	
Simulation Results	Correct	

위해 사용한 초기 값과 변화율, 에러 값, 입력 횟수, 시나리오 수의 값과 각 값의 의미를 정리한 표이다. 초기 값은 시나리오에서 값이 시작되는 부분을 설정하기 위한 값이며 변화율은 범위 값으로 범위 내에서 무작위로 값을 설정하고 현재 값과 연산하여 다음 입력 값을 결정한다. 에러 값은 에러로 인한 출력 생성을 관찰하기 위해서 설정하는 값으로 주요 기능을 관찰하기 위해서는 에러가 발생하지 않게 설정하였다. 입력 횟수와 시나리오의 수는 시뮬레이션 대상을 충분히 실행시키고 결과를 확인하기 위해 100개의 입력을 가지는 1000개의 시나리오를 생성하였다. FIX_FALLING의 주요 기능은 (P)TRIP 값의 생성과 (P)TSP 값의 변화이다. 입력 값인 PV_OUT이 (P)TSP 보다 낮을 값을 일정 주기 이상 지속될 경우 (P)TRIP의 값이 1이 되는 것이고 (P)TSP 값은 현재 값에서 Hysterisis 값을 더한 값이 된다. 그 후 입력 값이 현재의 (P)TSP 보다 위로 올라갈 경우 (P)TRIP이 다시 0이 되고 (P)TSP의 값을 원래대로 되돌리게 된다.

생성한 시나리오 중 FIX_FALLING의 주요 기능을 확인할 수 있는 시나리오를 이용하여 FBD Simulator를 통해 시뮬레이션을 수행한 결과가 그림 6이며 ModelSim을 통해 시뮬레이션을 수행한 결과가 그림 7이다. 그림 6에서 입력 값(노란색)은 V자 형태로 하강하다가 상승하는 경향을 보여주는데 PTSP(분홍색 선)와 TSP(파란색) 아래로 내려가고 일정 시간이 지난 후 (P)TSP의 값이 상승하는 것을, 위로 올라갈 경우 (P)TSP의 값이 원래대로 돌아오는 것을 관찰할 수 있다. ModelSim을 통한 시뮬레이션 결과는 Waveform의 형태로 출력되며 결과적으로 FBD Simulator와 같은 결과를 보여준다. 결과 분석은 pulse가 1이 되는 순간을 기준으로 수행하게 된다. 그림 7에서 가장 좌측 원 부분이 (P)TRIP이 1이 되는 부분을 의미하며 그 다음 원 부분은 (P)TRIP이 다시 0이 되는 부분을 의미한다.

FBD 시뮬레이션 결과와 VHDL 시뮬레이션의 결과를 비교하였을 때 동일한 입력을 주었을 때 동일한 출력이 발생하는 것을 확인할 수 있었으며 동일한 기능을 수행하는 것을 확인할 수 있었다. FIX_FALLING 외에 다

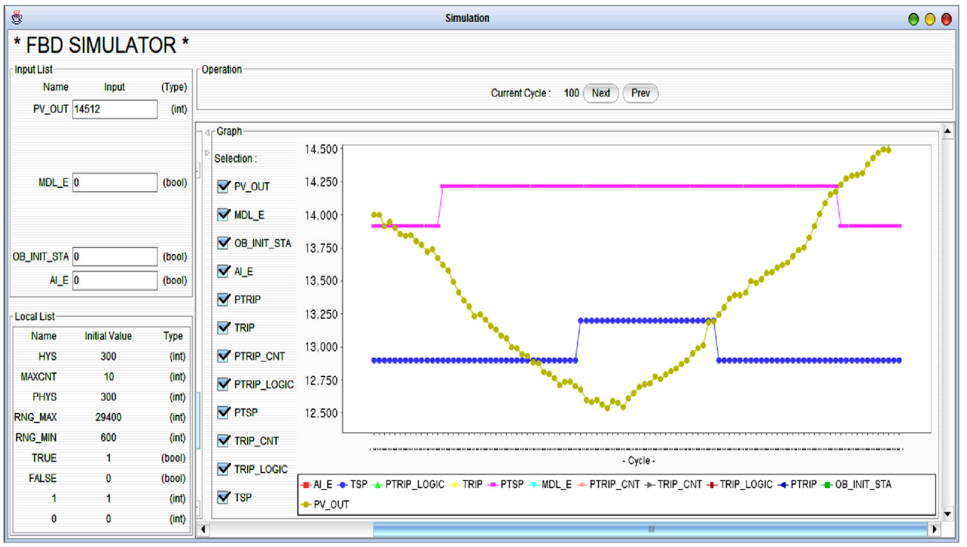

그림 6 FBD Simulator를 통한 FIX_FALLING(FBD)의 시뮬레이션 결과
Fig. 6 A simulation result of FIX_FALLING (FBD) using FBD Simulator

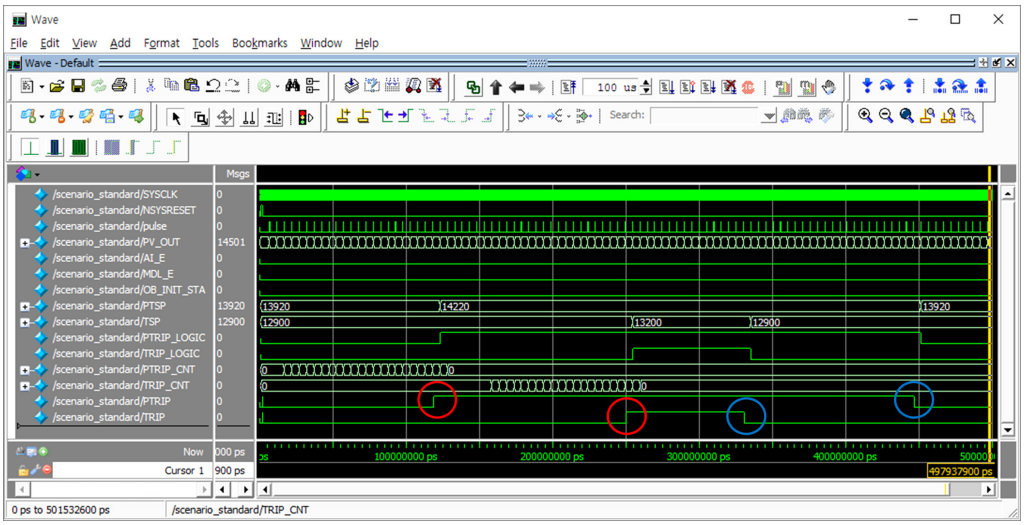

그림 7 ModelSim를 통한 FIX_FALLING(VHDL)의 시뮬레이션 결과
Fig. 7 A simulation result of FIX_FALLING (VHDL) using ModelSim

른 4개의 주요 로직과 g_BP 전체를 대상으로 동일한 실험을 수행하였으며 시뮬레이션 결과를 분석하였을 때 VHDL의 기능이 FBD와 같은 것을 확인하였다. 따라서 시뮬레이션을 통한 분석을 통해 FBDtoVHDL을 통해 FBD와 동일한 기능을 수행하는 VHDL을 생성하였다고 볼 수 있다.

5. 관련연구

PLC 개발을 위한 언어를 FPGA의 개발을 위한 하드웨어 기술 언어로 변환하는 연구는 다방면으로 연구되어 왔으며 대표적인 연구는 다음과 같다. PLC 개발을 위해 가장 널리 사용되어온 언어인 LD (Ladder Diagram)를 VHDL로 변환한 연구로는 대표적으로 [10,11]이 있다. LD의 이용하여 실행 순서와 의존 관계를 그래프를 통한 모델링을 수행하고 다양한 그래프 분석 기법을 적용하여 변환 알고리즘을 제안한다. LD와 FBD는 표현 방식의 다르기 때문에 본 논문에서 제안하는 변환 규칙과는 차이가 있다.

[12,13]는 PLC 개발을 위한 모든 언어를 대상으로 VHDL로 변환하는 연구이다. PLC 개발을 위한 모든 언어를 Statement List (STL)로 변환하고 중간언어인 C로 변환한 후 VHDL로 변환을 수행한다. PLC 언어를 STL로 변환하기 위해서 SIMATIC STEP 7 환경 하에서 변환을 수행한다. 중간언어인 C로 변환을 수행하기 위한 변환 규칙과 VHDL로 변환을 수행하는 전반적인 변환 프로세스는 본 논문에서 제안하는 변환과는 차이가 있다.

[14]는 FBD를 Verilog 변환하는 연구로 본 논문의 모티브가 된 논문이며 변환 룰을 정의하여 자동변환을 수행한다. FBD를 Verilog로 변환하기 위해서 그림 2와 유사한 변환 규칙을 제안하고 이를 기반으로 자동변환 도구인 FBDtoVerilog를 소개하였다. [14]에서 제안하는 변환 규칙은 POU를 기준으로 하였다는 점에서 본 논문에서 정의한 규칙과 유사하지만 본 논문에서는 VHDL을 작성하기 위한 변환 결과 프레임을 정의하고 제안하였다는 점에서 차이가 있다.

[15]는 [14]와 마찬가지로 FBD를 Verilog로 변환하는 변환 규칙을 정의하고 정형검증을 수행하였다. 본 논문을 포함하여 다른 논문들은 FPGA를 개발하기 위한 하드웨어 기술언어를 PLC 기반 언어를 이용해 변환하는 것이 목적이지만 [15]은 FBD를 정형검증 하는 것이 목적으로 한다. 따라서 FBD를 Verilog 변환하여 정형검증 도구인 VIS[16]의 입력 언어로 사용하였다. 검증을 위한 목적으로 수행한 연구이기 때문에 시간 속성을 고려하지 않았다는 점에서 본 논문과 차이가 있다.

6. 결론 및 향후 연구

본 논문에서는 FBD를 VHDL로 자동 변환 규칙과 이를 기반으로 구현한 도구인 FBDtoVHDL을 소개하였다. KNICS RPS BP 5개의 주요 로직과 APR-1400 원자로 보호 시스템의 프로토타입 예제를 대상으로 구조적 일치성을 확인하고 시뮬레이션을 통한 분석을 수행하였다. 구조적 분석을 통하여 1:1 수준의 변환이 수행되는 것을 확인하였으며 시뮬레이션 결과 분석을 통해 FBD로부터 동일한 기능을 수행하는 VHDL이 생성되는 것을 확인하

였다. 본 논문에서 제안하는 변환 규칙과 도구를 통해 PLC 개발자가 하드웨어 기술 언어에 대한 지식 없이 FBD를 이용하여 FPGA 개발을 위한 VHDL 설계를 생성하는 것이 가능하다. 또한 기존 PLC의 V&V 및 안전성 분석 등의 기술과 노하우를 FPGA 기반 시스템을 개발할 때 재사용할 수 있게 될 것이라 기대한다.

향후 계획으로는 변환 전후의 일치성을 확인하기 위하여 [15]과 같이 정형검증 방법을 이용한 검증을 수행할 예정이다. 구조적 일치성과 시뮬레이션을 통한 확인에는 한계가 있기 때문에 정형검증을 통한 확인이 요구되기 때문이다. 또한 FPGA 개발 프로세스를 FBDtoVHDL을 이용하여 변환한 VHDL을 통해 수행하고 하드웨어에서의 기능이 FBD와 동일한지 확인할 예정이다. 추가적으로 FPGA 개발 프로세스에서 발견된 수정 사항을 FBD에서 적용하는 방안에 대해서도 연구할 예정이다.

References

[1] International Electronical Commission (IEC), "International standard for programmable controller-Part 3: Programming languages," IEC, 1993.

[2] International Atomic Energy Agency, 2008, "Instrumentation and control (I&C) systems in nuclear power plants: A time of transition," [Online]. Available: http://www.iaea.org (downloaded 2015, Nov. 11)

[3] International Atomic Energy Agency (IAEA), 2002, "Instrumentation and Control Systems Important to Safety in Nuclear Power Plants Safety Guide," [Online]. Available: http://www.iaea.org (downloaded 2015, Nov. 11)

[4] J-G Choi, et al., "Survey of the CPLD/FPGA technology for application to NPP digital I&C system," Korea Atomic Energy Research Institute (KAERI), Tech. Rep., 2009.

[5] Korea Atomic Energy Research Institute (KAERI), "KNICS-RPS-SRS101 Rev.00.," 2003.

[6] Institute of Electrical and Electronics Engineers (IEEE), IEEE Standard VHDL Language Reference Manual, IEEE-1076, 2009.

[7] PLCopen Technical Committee 6, 2009, XML Formats for IEC 61131-3, [Online]. Available: http://www.plcopen.org (downloaded 2015, Nov. 11)

[8] H. Back, J. Yoo, S. Cha, "A CASE Tool for Automatic Generation of FBD Program from NuSCR Formal Specification," *Journal of KIISE : Computing Practices and Letters*, Vol. 15, No. 4, pp. 265-269, Apr. 2009. (in Korean)

[9] J. Kim, E.-S Kim, J. Yoo, Y. J. Lee, J.-G Choi, "An Integrated Software Testing Framework for FPGA-based Controllers in Nuclear Power Plants," *Nuclear Engineering and Technology*, Vol. 48, No. 2, pp. 470-481, 2016.

[10] D. Du, Y. Liu, X. Guo, K. Yamazaki and M. Fujishima, "Study on LD-VHDL conversion for FPGA-based PLC implementation," *The International Journal of Advanced Manufacturing Technology*, Vol. 40, No. 11-12, pp. 1181-1190, 2009.

[11] D. Du, X. Xu, and K. Yamazaki, "A study on the generation of silicon-based hardware Plc by means of the direct conversion of the ladder diagram to circuit design language," *The International Journal of Advanced Manufacturing Technology*, Vol. 49, No. 5-8, pp. 615-626, 2010.

[12] C. Economakos, and G. Economakos, "FPGA implementation of PLC programs using automated high-level synthesis tools," *2008 IEEE International Symposium on Industrial Electronics*, pp. 1908-1913, 2008.

[13] C. Economakos and G. Economakos, "Optimized FPGA implementations of demanding PLC programs based on hardware high-level synthesis," *2008 IEEE International Conference on Emerging Technologies and Factory Automation*, pp. 1002-1009, 2008.

[14] E.-S. Kim, J. Yoo, J.-G. Choi, Y. J. Lee, and J.-S. Lee, "A Technique for Demonstrating Safety and Correctness of Program Translators: Strategy and Case Study," *2014 IEEE International Symposium on Software Reliability Engineering Workshops*, pp. 210-215, 2014.

[15] J. Yoo, J.-H. Lee, S. Jeong, and S. D. Cha, "FBDtoVerilog: A Vendor-Independent Translation from FBDs into Verilog Programs," *The Twenty-Third International Conference on Software Engineering and Knowledge Engineering (SEKE 2011)*, pp. 48-51, 2011.

[16] R. K. Brayton, et al., "VIS: A system for verification and synthesis," *Computer Aided Verification*, Springer, pp. 428-432, 1996.

김 재 엽

2015년 건국대학교 컴퓨터공학부 졸업(학사). 2015년~현재 건국대학교 컴퓨터·정보통신공학부 석사과정. 관심분야는 소프트웨어 공학, 안전성 분석

김 의 섭

2015년 건국대학교 컴퓨터 정보통신공학부 졸업(석사). 2015년~현재 건국대학교 컴퓨터·정보통신공학부 박사과정. 관심분야는 소프트웨어 공학, 정형기법

유 준 범

2005년 KAIST 전자전산학과 전산학전공 졸업(박사). 2008년 삼성전자주식회사 통신연구소 책임연구원. 2008년~현재 건국대학교 컴퓨터공학부 부교수. 관심분야는 소프트웨어 공학, 안전성 분석, 정형기법

이 영 준

2003년 충남대학교 컴퓨터공학과 졸업(석사). 2008년 충남대학교 컴퓨터공학과 수료(박사). 2003년~현재 한국원자력연구원 선임연구원. 관심분야는 원전 I&C 시험, 신뢰도, 사이버 보안

최 종 균

2001년 KAIST 원자력공학과 졸업(박사) 2001년~현재 한국원자력연구원 책임연구원. 관심분야는 디지털 계측제어계통 설계, 안전성 분석