함수 블록 다이어그램으로 구현된 PLC 프로그램에 대한 정형 검증 기법

(A Formal Verification Technique for PLC Programs Implemented with Function Block Diagrams)

지 은 경 † 전 승 재 ††
(Eunkyoung Jee) (Seungjae Jeon)

차 성 덕 †††
(Sungdeok Cha)

요 약 프로그래머블 로직 콘트롤러(PLC)가 원자력 계측제어 시스템과 같은 안전 필수 시스템 구현에 많이 사용됨에 따라, PLC 프로그램에 대한 정형검증의 필요가 높아지고 있다. 본 연구에서는 함수 블록 다이어그램(FBD)으로 구현된 PLC 프로그램에 대한 자동화된 정형검증 기법을 제안한다. FBD 프로그램을 검증하기 위해서 먼저 FBD 프로그램을 검증언어인 Verilog로 변환하고, 변환된 Verilog 모델에 대해 SMV 모델체커를 호출해 모델체킹을 수행한다. 자동화를 위해 FBDVerifier 도구를 개발하였다. FBD Verifier는 FBD 프로그램으로부터 Verilog 모델로의 자동 변환 기능뿐 아니라 모델체킹 결과 생성된 반례를 직관적이고 효과적으로 분석할 수 있는 기능 또한 제공한다. 제안된 기법과 도구를 사용해 원전계측제어시스템 개발사업단의 원자로 보호시스템에 대한 방대한 양의 FBD 프로그램을 성공적으로 검증하였다.

키워드 : 프로그래머블 로직 콘트롤러, 함수 블록 다이어그램, 정형검증, 모델체킹, 반례분석

Abstract As Programmable Logic Controllers (PLCs) are increasingly used to implement safety critical systems such as nuclear instrumentation & control system, formal verification for PLC based programs is becoming essential. This paper proposes a formal verification technique for PLC program implemented with function block diagram (FBD). In order to verify an FBD program, we translate an FBD program into a Verilog model and perform model checking using SMV model checker. We developed a tool, FBDVerifier, which translates FBD programs into Verilog models automatically and supports efficient and intuitive visual analysis of a counterexample. With the proposed approach and the tool, we verified large FBD programs implementing reactor protection system of Korea Nuclear Instrumentation and Control System R&D Center (KNICS) successfully.

Key words : Programmable Logic Controller, Function Block Diagram, Formal Verification, Model Checking, Counterexample Analysis

1. 서 론

프로그래머블 로직 컨트롤러(PLC: Programmable Logic Controller)[1]는 산업용 컴퓨터로서 제어 시스템 구현에 널리 사용되고 있다. PLC가 원자력 제어계측 시스템과 같은 안전 필수 시스템 구현에 많이 사용되면서 PLC 소프트웨어에 대한 확인 및 검증의 필요성이 높아지고 있다. 본 연구는 IEC61131-3[2] 표준에 정의된 PLC 프로그래밍 언어들 중, 가장 널리 사용되는 것 중 하나인 함수 블록 다이어그램(FBD: Function Block Diagram)으로 작성된 프로그램을 대상으로 한다. 안전 필수 소프트웨어를 구현한 FBD 프로그램에 대한 철저한 검증은 필수적이라 할 수 있지만, FBD 프로그램에 실제로 적용 가능한 정형검증 기법에 대한 연구는 미비한 실정이다. 이에 본 논문에서는 FBD로 구현된 PLC 소프트웨어에 대한 정형검증 기법을 제안하고자 한다.

FBD 프로그램을 정형검증 하기 위해 FBD 프로그램을 Verilog[3] 모델로 자동 변환하며, 변환된 Verilog 모델에 대해 Cadence SMV 모델체커(model checker)를 이용해 정형검증을 수행한다. 이러한 일련의 검증과정을 자동화하기 위해 FBDVerifier 도구를 개발하였다.

FBDVerifier는 FBD 프로그램을 Verilog 모델로 자동 변환하여 SMV 모델체킹이 가능하게 해 줄 뿐만 아니라, Cadence SMV 모델체커를 통해 얻어진 반례(counterexample)의 사이즈가 클 때 반례 분석에 시간과 노력이 너무 많이 든다는 문제점을 해결하기 위해

· 이 논문은 2008 한국컴퓨터종합학술대회에서 '함수 블록 다이어그램으로 구현된 PLC 프로그램에 대한 정형검증'의 제목으로 발표된 논문을 확장한 것임
· 본 연구는 지식경제부 및 정보통신연구진흥원의 대학 IT연구센터 지원사업과 (IITA-2009-(C1090-0902-0032)) 고려대학교 특별연구비에 의하여 수행되었음

† 학생회원 : KAIST 전자전산학과
ekjee@dependable.kaist.ac.kr
†† 정 회 원 : 삼성전자 영상디스플레이사업부 사원
seungjae.jeon@samsung.com
††† 종신회원 : 고려대학교 컴퓨터통신공학부 교수
scha@korea.ac.kr

논문접수 : 2008년 9월 4일
심사완료 : 2008년 11월 16일

반례 분석 지원 기능 또한 제공한다. 텍스트 형태의 반례 정보를 직관적인 타이밍 그래프로 표현하여 분석을 용이하게 하며, 사용자가 관심 있는 수식을 선별해 변화 추이를 관찰할 수 있게 함으로써 오류의 원인을 짧은 시간 안에 효과적으로 찾을 수 있게 한다.

제안된 기법과 도구를 사용하여 원전계측제어시스템 개발사업단(KNICS)[4]의 원자로 보호 시스템(RPS: Reactor Protection System) 정형검증을 성공적으로 수행하였으며, 이를 통해 제안된 기법과 도구의 효과를 확인할 수 있었다.

본 논문은 다음과 같이 구성되어 있다. 2장에서 FBD와 Verilog 모델체킹에 대해 간략히 설명하며, 3장에서는 FBD로부터 Verilog로의 변환 방법을 설명한다. 4장에서는 FBDVerifier 도구와 실제 시스템에 대한 사례 연구를 소개하고 5장에서는 관련연구를 소개한다. 끝으로 6장에서 결론을 맺는다.

2. 배 경

2.1 함수 블록 다이어그램

PLC[1]는 화학 공정 발전소, 원자력 발전소, 교통 제어 시스템 등 제어 시스템에 광범위하게 사용되고 있는 산업용 컴퓨터이다. FBD는 IEC 61131-3[2] 표준에 정의된 다섯 가지 PLC 프로그래밍 언어들 중 가장 널리 사용되는 것 중 하나로서, 표기 방법이 이해하기 쉽고 제어 블록들간의 데이타 흐름을 잘 표현하는 언어이다.

FBD 프로그램은 시스템 행위를 함수 블록들간 신호의 흐름으로 표현한다. 입력 변수와 출력 변수 사이의 연결은 전기 회로와 같은 형태로 연결된 함수 블록들의 집합으로 표현된다.

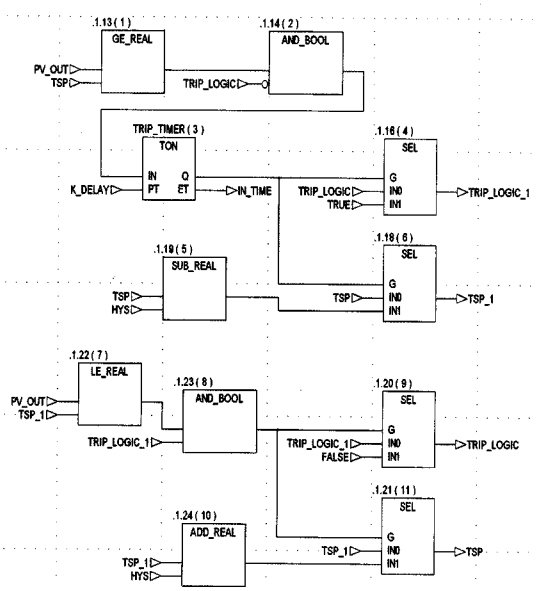


그림 1 원자로 보호 시스템의 고정설정치상승(FIX_RISING) 모듈

그림 1은 FBD 프로그램의 예를 보여준다. FBD 프로그램에서 출력 변수들은 함수 블록들의 순차적 조합에 의해 계산된다. 그림 1은 KNICS 원자로 보호 시스템 논리의 일부로서 고정설정치 상승 트립 여부를 계산하는 부분이다. 출력 변수인 TRIP_LOGIC은 공정값(PV_OUT)이 지정된 트립설정치(TSP) 이상인 상태로 K_DELAY 시간 이상 지속되면 트립을 나타내는 값인 1로 설정된다. 트립이 발생하면 이후 논리에서 원자로를 정지시키는 작업을 수행하게 된다. 앞으로 이 FBD 프로그램 예제를 통해 제안하는 기법을 설명한다.

2.2 Verilog 모델체킹

Verilog[3]는 광범위하게 사용되는 하드웨어 설계 언어(HDL:Hardware Description Language)중 하나이다. FBD의 시맨틱(semantics)과 Verilog의 시맨틱이 비슷하여 FBD가 Verilog로 효과적으로 변환될 수 있기 때문에, FBD 프로그램 검증언어로 Verilog를 선택하였다.

Verilog로 변환된 FBD를 정형 검증하기 위하여 모델체킹 기법을 사용하였다. 모델체킹은 시스템이 주어진 속성을 만족하는가를 조사하는 기법이며, 시스템이 속성을 만족하면 참(true)이라는 결과를 내주고, 만족하지 않으면 반례를 제공한다. 이 반례는 시스템 오류를 분석하는데 중요한 자료가 된다.

본 연구에서는 FBD로부터 생성된 Verilog 모델을 검증하기 위한 모델체커로서 Cadence SMV를 사용하였다. 다른 Verilog 모델체커들도 사용될 수 있다.

3. FBD로부터 Verilog 모델로의 변환

함수 블록 다이어그램에 대한 정형적 정의가 [5]에 정리되어 있다. [5]의 정형적 정의를 바탕으로, FBD 프로그램으로부터 Verilog 모델로 변환하는 규칙이 그림 2에 정리되어 있다.

그림 2의 템플릿을 따라 FBD 프로그램으로부터 Verilog 모델이 생성된다. 규칙 1에서는 모듈 이름과 입출력 포트를 선언하고, 규칙 2에서는 변수의 저장 타입, 비트 벡터의 크기가 선언된다. 규칙 3은 레지스터 타입(reg) 변수의 초기값을 지정한다. 함수 블록들의 조합으로 표현된 FBD의 핵심 논리는 규칙 4에서 변환된다. 규칙 5는 현재 주기의 출력값을 reg 변수에 저장하는 과정으로, @ (**posedge** clk)는 클럭의 상승 에지(positive edge), 즉 매 주기의 시작을 의미한다. 갱신된 reg 변수의 값은 다음 주기에서 사용된다. 마지막으로 규칙 6에서 사용자가 검증 속성을 삽입하면 Verilog 모델 생성이 완료된다. 그림 3은 그림 1의 FBD로부터 생성된 Verilog 모델이다.

```
// 규칙 1. 모듈 선언
module main (clk, [input ports], [output ports]);

// 규칙 2. 변수 선언-타입 및 비트(bit) 크기 결정
input | reg | wire | output [size(v) : 0] v;

// 규칙 3. reg 변수(v_reg)들에 대한 초기화
initial begin
v_reg = [initial_value_of_v_reg];
end

// 규칙 4. wire 변수 및 출력 변수들에 대한 계산식 할당
assign v = f(v);

// 규칙 5. reg 변수(v_reg)들에 대해 저장할 값들 할당
always @ (posedge clk) begin
v_reg = [stored value];
end

// 규칙 6. 검증 속성 삽입
always begin
{if [condition]} assert [label]: [assertion];
end
endmodule
```

그림 2 FBD 프로그램으로부터 Verilog 모델을 생성하는 템플릿

4. 사례 연구 및 FBDVerifier 도구

4.1 사례 연구

제안된 FBD 검증 기법을 KNICS 사업단의 원자로 보호 시스템(RPS)[6]에 적용하였다. RPS는 비교논리프로세서(BP: Bistable Processor), 동시논리프로세서(CP: Coincidence Processor), 자동시험 및 연계프로세서(ATIP: Automatic Test and Interface Processor), 캐비닛운전원모듈(COM: Cabinet Operator Module)의 네 부분으로 구성되어 있는데, 이 중 안전 필수 등급 시스템인 BP와 CP에 대해서 정형검증을 수행하였다. 표 1은 정형검증 대상이 된 RPS 시스템에 대한 정보이다.

BP와 CP 시스템에 대한 설계명세는 합하여 350여 페이지가 된다. 이들 시스템에 대한 FBD 프로그램은 3천여 개의 함수 블록과 2천여 개의 변수로 구성되어 있다. 본 연구에서 제안한 방법과 도구를 사용하여 변환된

표 1 정형검증 대상 RPS 시스템 정보

대상 시스템	설계명세 페이지수 (페이지)	함수블록 개수 (개)	변수 개수 (개)	Verilog모델 라인수 (라인)
BP	190	1,335	1,038	7,862
CP	163	1,623	820	3,085

```
module main (clk, HYS, K_DELAY, PV_OUT, TRIP_TIMER_ET,
TRIP_TIMER_Q);

input clk;
input [1:0] HYS;
input [4:0] K_DELAY;
input [7:0] PV_OUT;
reg TRIP_LOGIC;
input TRIP_TIMER_ET;
input TRIP_TIMER_Q;
reg [7:0] TSP;
wire TRIP TIMER_IN;
wire [4:0] TRIP_TIMER_PT;
wire IN_TIME;
wire TRIP_LOGIC_1;
wire [7:0] TSP_1;
wire TRIP_LOGIC_out;
wire [8:0] TSP_out;

//constants
assign HYS = 1;
assign K_DELAY = 10;

initial begin
  TRIP_LOGIC = 0;
  TSP = 90;
end

assign TRIP_TIMER IN = ((PV_OUT >= TSP) && ! TRIP_LOGIC);
assign TRIP_TIMER_PT = K_DELAY;
assign IN_TIME = TRIP_TIMER_ET;
assign TRIP_LOGIC_1 = (TRIP_TIMER_Q ? 1 : TRIP_LOGIC);
assign TSP_1 = (TRIP_TIMER_Q ? (TSP - HYS) : TSP);
assign TRIP_LOGIC_out = (((PV_OUT =< TSP_1) &&
TRIP_LOGIC_1) ? 0 : TRIP_LOGIC_1);
assign TSP_out = (((PV_OUT =< TSP_1) && TRIP_LOGIC_1) ?
(TSP_1 + HYS) : TSP_1);

TON TRIP_TIMER(.clk(clk), .IN(TRIP_TIMER_IN),
PT(TRIP_TIMER_PT), .Q(TRIP_TIMER_Q), .ET(TRIP_TIMER_ET)
);

always @ (posedge clk) begin
TRIP_LOGIC = TRIP_LOGIC_out;
TSP = TSP_out;
end

always begin
if (PV_OUT >= TSP) && IN_TIME >= K_DELAY) assert P1:
TRIP_LOGIC_out == 1;
if (TRIP_LOGIC && PV_OUT >= TSP) assert P2: TRIP_LOGIC_out
== 1;
end
endmodule

module TON (clk, IN, PT, Q, ET);
input clk, IN;
input [4:0] PT;
output Q;
output [4:0] ET;
reg [4:0] t;
initial t = 0;
assign ET = t;
assign Q = IN && (ET >= PT);
always @ (posedge clk)
t <= IN ? ((t < PT) ? t+1 : PT) : 0;
endmodule
```

그림 3 그림 1의 FBD 프로그램으로부터 변환된 Verilog 모델

Verilog 모델은 전체 1만 라인 정도가 되었다.

도메인 전문가들이 추출한 필수 검증 속성들을 if… assert 형태의 논리식으로 만들어 Verilog 모델에 삽입하였고, SMV 모델체커로 모델체킹을 수행하였다. 대상 프로그램의 변수 개수가 많고 각 변수값의 범위가 커서

표 2 BP, CP에 대한 정형검증 결과

		BP	CP
검증한 속성 개수(개)		216	83
오류 분류	논리 오류	14	6
	명세 누락	0	2
	명세 모호	4	0
	FBD 프로그램 내 오류	13	5
	자연어설계명세 내 오류	16	0
전체 오류 (건)		47	13
주요 오류 (건)		10	3

상태폭발 문제가 자주 일어났기 때문에, 검증 과정에서 수동으로 추상화를 적용해 검증하였다. 검증 결과는 표 2와 같다.

BP, CP 시스템에 대해서 발견된 오류들의 구체적인 예는, _1_TRIP_LOGIC 변수가 할당되어야 하는 자리에 TRIP_LOGIC 변수가 할당된 변수명 잘못 사용 오류, > 연산자 대신 >= 가 사용된 연산자 사용 오류, 변수 범위 검사 논리 누락, 초기값이 설정되지 않아 의도치 않은 음수 연산 오류, 자연어 설계 명세와 FBD 프로그램간 명세가 다른 명세 일관성 오류, 임시 테스트 논리가 삭제되지 않고 남아 있어 발생한 오류 등이다. 모델 체킹을 통해 발견된 오류는 개발팀에 전달되어 BP, CP 소프트웨어 내 오류를 수정하고 안전성을 높이는 자료로 사용되었다.

4.2 FBDVerifier

제안된 FBD 검증 체계를 자동화하기 위해 FBD Verifier 도구를 개발하였다. 그림 4는 FBDVerifier 도구의 모습이다. FBDVerifier는 LDA 파일을 입력으로 받아 Verilog 모델로 변환한다. LDA는 FBD 저장 포맷으로, FBD 개발 도구인 pSET 에서 사용하는 포맷이다. 그림 4는 FBD프로그램을 Verilog로 변환하는 과정에서 사용자가 입력 변수의 비트(bit) 벡터 크기와 초기값을 결정해 주는 화면이다. FBD 프로그램으로부터 생성된 Verilog 모델에 사용자가 검증 속성을 수동으로 삽입하고 실행시키면 FBDVerifier는 Cadence SMV를 호출하여 모델체킹을 수행한다.

그림 5는 사례 연구 중 Cadence SMV에서 생성된 반례의 예를 보여준다. 변수 수가 많고 스텝수도 커서 이러한 반례 분석에는 상당한 시간과 노력이 든다. 반례 분석을 효과적으로 지원하기 위해 FBDVerifier는 반례의 내용을 하드웨어 엔지니어들에게 친숙한 타이밍 그

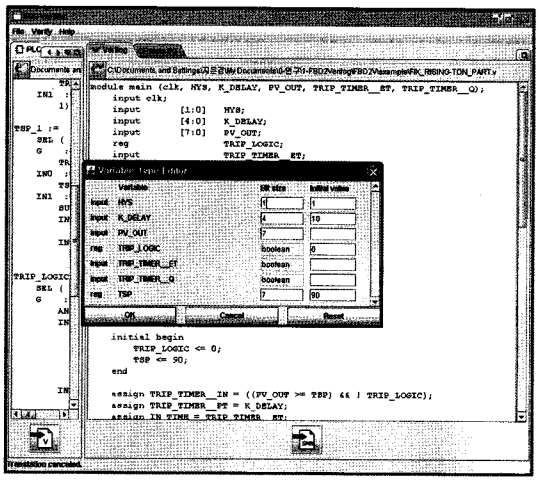


그림 4 FBDVerifier 스크린 덤프

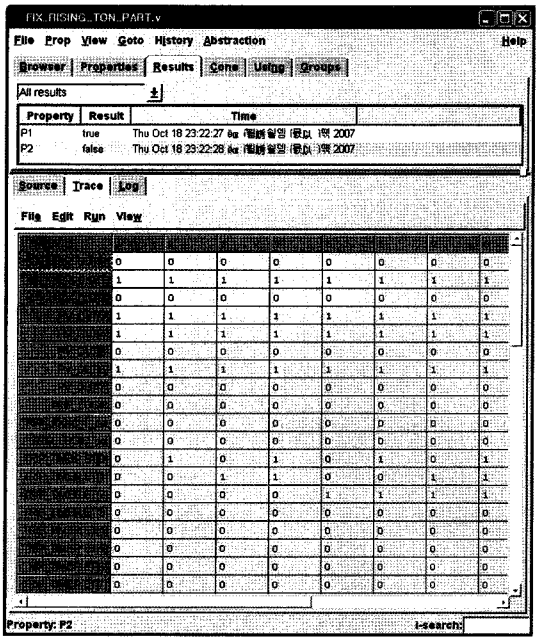
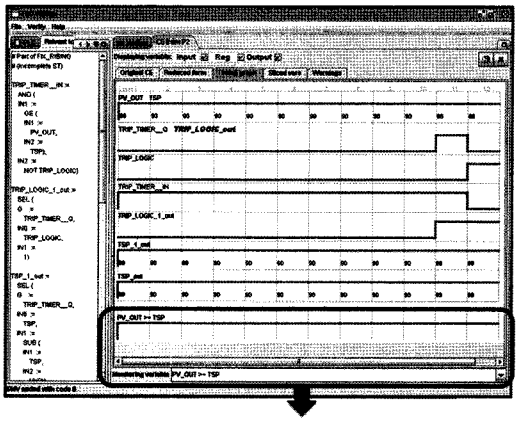


그림 5 Cadence SMV에서 생성된 반례



사용자가 추가한 모니터링 수식 및 그에 대한 타이밍 그래프

그림 6 FBDVerifier의 반례분석 지원 기능

래프 형태로 시각화하였다. 그림 5와 같은 반례 정보를 그림 6에서와 같이 타이밍 그래프로 시각화 하였으며, 사용자가 보기 원하는 수식을 입력하면 그 수식이 매 주기마다 어떻게 값이 변하는지 추적하여 보여줄 수 있는 모니터링 기능도 지원한다. 모니터링 수식은 프로그램에 쓰인 변수, 상수, 괄호나 사칙연산과 같은 산술 기호들을 써서 정의한다. 그림 6에서는 모니터링 수식에 "PV_OUT >= TSP"(공정치가 트립설정치 이상)와 같은 수식을 사용자가 입력하였고, 타이밍 그래프 중 맨 하단에 있는 그래프가 이 수식의 값 변화를 보여준다. 반례 정보에는 많은 변수들이 포함될 수 있는데 사용자가 원하는 부분만을 볼 수 있도록, 지정하는 변수들에 대한 정보만 보여주거나 숨기는 기능도 지원한다.

5. 관련 연구

PLC 프로그램에 대한 확인 및 검증, 시뮬레이션 또는 테스팅을 하기 위한 목적으로 PLC 프로그램을 정형화하려던 연구들이 [7]에 잘 정리되어 있다. FBD 프로그램에 대해서는 [8]에서 PLCTOOLS 도구가 소개되었다. [8]에서 FBD 프로그램은 하이 레벨 타임드 페트리넷(HLTPN: High Level Time Petri Nets)으로 모델링된다. PLCTOOLS 도구는 HLTPN으로 모델링된 FBD 프로그램을 확인(validation)하고 그로부터 코드 생성하는 것은 가능하지만, FBDVerifier처럼 정형검증을 지원하지는 않는다.

Cadence SMV 이외에 CBMC[9], VCEGAR[10]와 같이 Verilog 모델을 검증할 수 있는 다른 모델체커들도 있다. Cadence SMV 대신 이러한 모델체커들을 사용해 보는 것도 의미 있을 것이다.

반례 시각화에 대한 연구도 활발히 이루어지고 있다. smv2vcd[11]은 SMV에서 생성된 반례를 VCD (Variable Change Dump) 포맷으로 변환한다. VCD 포맷은 하드웨어 엔지니어들이 입출력값 분석을 위해 주로 사용하는 표준 포맷으로, 상용이나 공개로 개발되어 있는 많은 VCD 뷰어를 이용하여 볼 수 있다.

6. 결 론

본 논문에서는 PLC 프로그램 언어 중 널리 사용되는 FBD로 작성된 프로그램에 대한 정형검증 기법을 제안하였다. FBD 프로그램으로부터 Verilog 모델을 자동 생성하고, 생성된 Verilog 모델을 Cadence SMV 모델체커를 이용하여 검증하였다. 전체적인 검증 체계를 자동화 하기 위해 FBDVerifier 도구를 개발하였으며, 이 도구는 Verilog 모델 자동 생성과 효과적인 반례 분석을 지원한다.

제안된 방법을 KNICS RPS 설계명세 정형검증에 적용하였고, FBDVerifier의 변환 자동화 및 반례 분석 지원 기능을 통해 큰 규모의 FBD 프로그램을 비교적 짧은 시간 안에 효과적으로 검증할 수 있었다. 발견된 오류들은 시스템의 안전성을 높이는데 유용하게 사용되었고, 실제 산업에서의 사례연구를 통해 기법의 효용성을 확인할 수 있었다.

향후 연구로는 현재의 FBDVerifier가 pSET 도구의 저장 포맷만 지원한다는 단점을 개선하여 다양한 제조사의 FBD 저장포맷을 지원할 수 있도록 도구를 확장하고자 한다.

참 고 문 헌

[1] Mader, A., "A Classification of PLC Models and Applications," *Proc. WODES 2000: 5th Workshop on Discrete Event Systems*, Gent, Belgium, Aug. 21-23, 2000.

[2] International Standard for Programmable Controllers: Programming Languages (Part 3), IEC, 1993.

[3] Standard Hardware Description Language Based on the Verilog Hardware Description Language (Std 1364-2001), IEEE, 2003.

[4] 원전계측제어시스템 개발사업단 (KNICS: Korea Nuclear Instrumentation and Control System Research and Development Center), http://www.knics.re.kr

[5] 유준범, "NuSCR 정형명세로부터 Function Block Diagrams의 생성", 박사학위논문, KAIST, 2005.

[6] KNICS-RPS-SDS231 (Rev.02), 원자로보호계통 소프트웨어 설계 명세서, 두산중공업㈜, 2007.

[7] Bani Younis, M. and Frey, G., "Formalization of Existing PLC Programs: A Survey," *Proceedings of CESA 2003*, Lille, France, July 9-11, 2003.

[8] Baresi, L., Mauri, M., Monti, A. and Pezzè, M., "PLCTOOLS: Design, Formal Validation, and Code Generation for Programmable Controllers," *Formal methods in PLC programming Special Session at IEEE Conference on Systems, Man and Cybernetics (SMC'2000), Nashville*, USA, Oct. 8-11, 2000.

[9] CBMC, Bounded Model Checking for ANSI-C, http://www.cs.cmu.edu/~modelcheck/cbmc.

[10] Jain, H., Sharygina, N., Kroening, D. and Clarke, E., "Word Level Predicate Abstraction and Refinement for Verifying RTL Verilog," In *Proc. 42nd Design Automation Conference (DAC)*. Anaheim, USA, 2005.

[11] smv2vcd, http://www.cs.cmu.edu/~modelcheck/smv2vcd.html.