한국시뮬레이션학회 논문지
Vol. 17, No. 4, pp. 1-9 (2008. 12)

PLC 시뮬레이션에서 Plant model 자동 생성을 위한 PLC Symbol 규칙

박형태[1] · 왕지남[2] · 박상철[3†]

PLC symbol naming rule for auto generation of Plant model in PLC simulation

Hyeongtae Park · Ginam Wang · Sangchul Park

ABSTRACT

Proposed in the paper is an automated procedure to construct a plant model for PLC simulation. Since PLC programs only contain the control logic without the information on the plant model, it is necessary to build the corresponding plant model to perform simulation. Conventionally, a plant model for PLC simulation has been constructed manually, and it requires much efforts as well as the in-depth knowledge of simulation. As a remedy for the problem, we propose an automated procedure to generate a plant model from the symbol table of a PLC program. To do so, we propose a naming rule for PLC symbols so that the symbol names include enough information on the plant model. By analyzing such symbol names, we extract a plant model automatically. The proposed methodology has been implemented, and test runs were made.

Key words : Programmable Logic Controller(PLC), Plant model, Simulation, DEVS, Virtual Factory

요 약

본 논문은 Programmable Logic Controller(PLC) 시뮬레이션을 하기 위한 공장 모델(Plant Model)을 자동으로 생성하는 절차에 대해 기술한다. PLC 프로그램은 공정을 제어하는 로직에 관한 정보이며 그 자체로 공장 모델에 대한 어떤 정보도 포함하고 있지 않기 때문에 시뮬레이션을 위해서는 PLC 프로그램에 대응하는 공장 모델이 반드시 필요하다. 지금까지 PLC 시뮬레이션을 위한 공장 모델은 사용자가 직접 구축하는 방식으로 모델링 되었으나 이는 많은 노력과 공정로직의 완전한 이해 및 시뮬레이션 지식이 요구된다. 이런 어려움을 극복하기 위해 논문은 PLC 프로그램의 심볼테이블(Symbol table)로부터 공장모델을 자동으로 생성하는 과정을 제안한다. 이를 위해 PLC 심볼이 공장 모델의 생성을 위한 정보를 포함시키는 PLC 심볼의 작명 규칙을 제안한다. 입력된 심볼 리스트를 분석함으로써 공장 모델을 자동으로 추출할 수 있으며 간단한 예제 공정을 대상으로 구현해 본다.

주요어 : Programmable Logic Controller(PLC), 시뮬레이션, 공장 모델, I/O model

1. 서 론

현대 제조환경 시대에 놓인 제조업체는 기업의 생존과 번영을 위해 끊임없이 제품 개발과 생산 기술의 향상에 노력을 기울인다. 현대의 제조 라인은 다 기능 로봇, 물류 이동 시스템, 자동 저장 시스템, 그리고 전체 시스템의 작동을 제어하는 컴퓨터 제어 시스템 등의 자동화 작업장으로 구성된 고차원의 통합 시스템이다. 이런 고차원의 자동화 생산 라인의 구축은 큰 비용이 투자되기 때문에 구축하고자 하는 라인이 예상되는 목적을 성공적으로 달성 할 수 있는지에 대한 사전 구동 검증은 반드시 이뤄져야 한다.

시뮬레이션 기술은 수리적으로 기술되기 어려운 복잡한 시스템의 디자인과 분석을 위한 필수적인 도구로 여겨진다[1,2,3]. 시뮬레이션은 설비의 활용도를 통계적으로 계

* 본 연구의 일부는 방위사업청과 국방과학연구소의 지원으로 수행되었습니다.(UD080042AD)
2008년 5월 20일 접수, 2008년 12월 13일 채택
1) 아주대학교 산업공학과 대학원
2) 아주대학교 산업정보시스템 공학부 정교수
3) 아주대학교 산업정보시스템 공학부 부교수
주 저 자 : 박형태
교신저자 : 박상철
E-mail; scpark@ajou.ac.kr

산하거나 병목지점 찾기, 스케줄링 에러 감지 및 제조 스케줄을 생성하는데 유용하다. 전통적으로 ARENA, AutoMod 등을 포함한 다양한 시뮬레이션 언어가 제조 시스템의 시뮬레이션을 위해 사용된다[4]. 이런 시뮬레이션 언어는 산업분야와 학계에 널리 받아들여졌다. 그러나 생산 시스템의 세부 설계나 구축의 목적으로 사용될 만큼 충분히 실제적이지 않기 때문에 생산 라인의 디자인 단계에서의 분석 도구로 남아있다. 예를 들어 실제 생산 시스템은 보통 Programmable Logic Controller(PLC) 프로그램에 의해 제어되고 있지만 전통적인 시뮬레이션 언어들은 프로세스 사이의 독립적인 개체의 흐름으로 제어 로직을 대략적으로 묘사한다.

산업 제어 기술에서 가장 적합하며 널리 채택 중인 PLC에 의해 제어되는 생산 시스템은 동시적으로 작동하는 많은 기계들로 구성된다. PLC는 전기적인 Ladder Diagram을 모방한다. 즉각적으로 반응하는 평행 회로의 작동을 모방하기 위해 PLC는 순차적인 기계이기 때문에 입, 출력 신호와 순환적 스캔 싸이클을 이용한다. 하나의 프로그램이 PLC에서 작동할 때, 끊임없이 스캐닝 싸이클을 동작하게 된다. 프로그램 스캔은 입력으로 들어온 갱신된 신호에 따라 내부 와 출력 릴레이 테이블과 관련된 Boolean 로직을 해석하게 된다. 더욱이 내부 및 출력 릴레이 테이블 정보는 프로그램 스캔 동안 새롭게 수정된다. PLC에서 이 Boolean 로직은 일반적으로 Ladder diagram이라고 불리는 그래픽 언어로 표현된다[5].

전통적인 시뮬레이터와 PLC 프로그램의 추상화 수준이 상당히 다르기 때문에, 전통적인 시뮬레이터의 제어 로직은 PLC 프로그램을 위해 재사용 될 수 없다. 통상적으로 전기 기술자는 그림 1에서 보이는 것처럼 전통적 시뮬레이터의 대략적인 제어 로직을 참고하여 PLC 프로그램을 작성하게 된다. PLC 프로그래밍 작업은 매우 지루하고 에러를 포함할 경향이 큰 작업이므로 생산 시스템의 안정화 시간을 줄이기 위해 PLC 프로그램을 오프라인으로 검증하는 것은 필수적이다.

PLC 프로그램에 관련된 이전의 접근은 크게 두 가지 그룹으로 분류된다: (1) 주어진 PLC 프로그램의 검증[6-10]과 (2) 신뢰할수 있는 PLC 프로그램의 생성[11-13]. 첫 번째 그룹에서는 주로 Statement List 언어로 기술된 PLC 프로그램을 UPPAAL2k, KRONOS, Supremica 그리고 HyTech같은 Timed Automata의 사용을 통한 PLC 기반의 시스템 검증을 위한 다양한 소프트웨어가 개발 되었다[14]. 이러한 소프트웨어는 PLC 프로그램을 어느 정도까지는 검증 할 수 있다. 그러나 이는 제한적이다. 이것들은

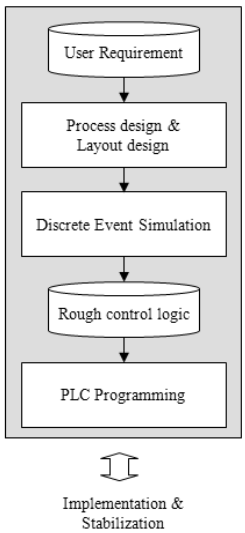


Fig. 1. Conventional procedure to construct a production system

주로 이론적인 속성(안전, 시스템생존여부)에 관점을 두고 있어서 사용자는 PLC 프로그램이 의도하는 제어 목적을 실제적으로 달성하는지의 여부를 결정하기가 쉽지 않다. 두 번째 그룹에서 많은 연구자들은 state diagram, Petri nets, 그리고 IDEF0를 포함하는 다양한 formalism으로부터 PLC 프로그램의 자동 생성에 관점을 두었다. 이러한 formalism은 제어 로직의 디자인 과정에서 도움이 될 수 있으나 제어 프로그램을 검증하는 가장 어려운 부분인 숨겨진 에러를 찾는 것은 여전히 어려운 일이다.

앞서 언급한 문제를 극복하기위해 PLC 프로그램 검증을 위한 시뮬레이션 기술의 사용이 필요하다. PLC 프로그램을 시뮬레이션 함으로써 다양한 면에서 제어 로직의 분석과 숨겨진 에러를 좀더 직관적으로 찾는 것이 가능하게 될 것이다. 비록 PLC 시뮬레이션이 생산 시스템의 섬세한 검증을 위한 강력한 도구임에도 불구하고 이에 수반하는 공장모델(제어프로그램의 대응모델)의 구축은 중요한 장애물이다. PLC 프로그램은 device 모델이 없는 오직 제어 정보만을 포함하고 있기 때문에 Fig. 2에 보이는 것처럼 시뮬레이션을 수행하기위해 이에 대응하는 공장모델의 구축이 필요하다. 그러나 공장모델의 구축은 과도한 시간과 노력이 요구된다. 때때로 공장모델의 구축은 PLC 프로그램작성보다 훨씬 더 많은 시간이 소요된다. 이점이 PLC 프로그램으로부터 공장 모델을 생성하기 위한 자동 절차를 찾는 가능성을 모색하게 되는 본 논문의 동기이다.

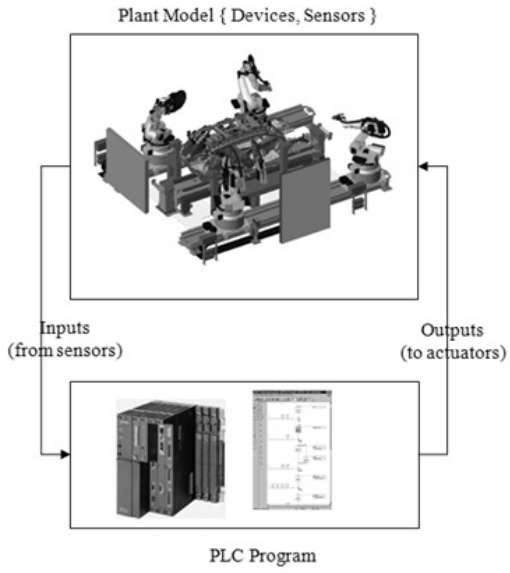


Fig. 2. The concept of PLC simulation

	A	B	C	D	E	F
1	Class	Identifier	MIT-Addr.	IEC-Addr.	Type	Initial
2	VAR_GLOBAL	EXLINE_MB_AGV_I_P1	X0001	%IX0001	BOOL	FALSE
3	VAR_GLOBAL	EXLINE_MB_AGV_I_P2	X0002	%IX0002	BOOL	FALSE
4	VAR_GLOBAL	EXLINE_MB_AGV_O_P2	Y0001	%QX0001	BOOL	FALSE
5	VAR_GLOBAL	EXLINE_MB_AGV_O_P1	Y0002	%QX0002	BOOL	FALSE
6	VAR_GLOBAL	EXLINE_MB_MC_I_IDLE1	X0003	%IX0003	BOOL	FALSE
7	VAR_GLOBAL	EXLINE_MB_MC_I_RUN2	X0004	%IX0004	BOOL	FALSE
8	VAR_GLOBAL	EXLINE_MB_MC_O_IDLE1	Y0003	%QX0003	BOOL	FALSE
9	VAR_GLOBAL	EXLINE_MB_MC_O_RUN2	Y0004	%QX0004	BOOL	FALSE

Fig. 3. Symbol table of a PLC program

PLC 프로그램의 목적이 비록 공장모델(디바이스모델)을 기술하기 위한 것은 아니지만, PLC 프로그램의 심볼 테이블은 공장모델의 일견을 제공할 수 있다. Fig. 3에 보이는 것처럼 PLC 프로그램에서 심볼은 보통 공장에 관련된 정보를 포함하고 있다. 예를 들어, 'EXLINE_MB_AGV_I_P1'은 'EXLINE' 라인의 'MB' 공정에 속하는 AGV (Auto Guided Vehicle) 설비의 제어에 관련된 신호를 의미한다.

본 논문은 다음의 두 가지 목적을 둔다: (1) PLC 심볼의 적절한 명명규칙의 제안과 (2) 심볼 이름을 분석함으로써 공장모델을 생성하기 위한 절차의 고안이다. 제안하는 방법론의 적용 분야는 자동차 생산라인, FMSs(flexible manufacturing systems) 그리고 ASRSs(automatic storage and retrieval systems) 같은 PLC에 의해 제어되는 모든 종류의 자동화 제조 시스템을 포함한다. 논문의 구성은 다음과 같다. 2장에서는 PLC 시뮬레이션을 위한 공장모델을 상세하게 기술한다. 3장은 공장모델의 자동생성을 가능하게 하는 PLC 심볼의 작명규칙을 기술하며 마지막으로 논문의 결론이 4장에 나타난다.

2. PLC 시뮬레이션을 위한 공장모델 (Plant Model)

PLC 시뮬레이션을 가능하게 하는 공장모델의 설명에 앞서 우리는 PLC 시뮬레이션의 중요성을 살펴보자. Chuang *et al*.[15]은 다음과 같은 아홉 단계로 구성된 산업 자동화 생산 시스템의 개발 절차를 제안하였다: (1) 제어할 공정의 정의; (2) 공정 운전의 스케치; (3) 공정 순서 만들기; (4) 스케치에서 공정 순서를 수행하기 위해 필요한 센서 삽입; (5) 프로세스 셋업 혹은 공정 체크를 위해 필요한 매뉴얼 컨트롤 삽입; (6) 작업 근로자의 안전 고려 후 필요시 추가물 혹은 조정장치 만들기; (7) 안전 비상 정지를 위한 마스터 스톱 스위치 추가; (8) PLC 프로그램의 기초로 사용 될 Ladder 로직 다이어그램 만들기; (9) 공정 순서가 정상에서 벗어날 가능한 지점의 고려. 여기서 제어 로직 엔지니어에게 가장 시간 소모적인 과정은 여덟 번째 과정이며 여기서는 제어 목적이 달성되기 까지 코드 작성, 테스트, 디버깅의 반복적인 과정이 수행된다[14]. 이 때문에 전통적인 PLC 프로그램 작성이 보통 비효율적이며 사람의 에러가 내포될 가능성이 높은 이유이다. 생산 라인의 구성과 그것의 제어 프로그램이 점점 복잡해짐 에 따라 좀 더 효율적인 PLC 시뮬레이션 환경이 절실히 요구되게 된다. 이런 관점에서 본 논문이 긍정적인 행보를 할 수 있기를 희망한다.

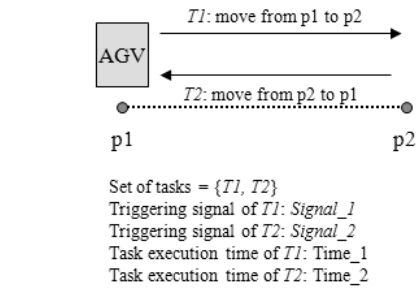
PLC는 입력, 출력 신호를 가진 전용의 컴퓨터 시스템으로 여겨진다. PLC를 실행하기 위해 PLC의 입력과 출력 신호에 대응하는 공장모델(대응시스템)이 요구된다. 공장 모델의 행위는 PLC 검증을 달성하기 위해 실제 시스템의 행위와 같아야만 한다. 생산 시스템은 로봇, 운반기, 지그, 솔레노이드밸브, 근접센서, 광센서[16] 등을 포함한 다양한 디바이스로 구성되어 있기 때문에 우리는 공장모델을 디바이스모델의 세트로 생각할 수 있다. 이러한 디바이스모델을 만들기 위해 본 논문은 이산 사건 모델을 구조적, 모듈적 방법 도시를 지원하는 Zeigler의 DEVS (Discrete Event Systems Specification) 형식론을 채택하였다[17,18]. 이 DEVS 형식의 의미론은 시뮬레이션 모델을 위한 객체 지향 개념과 높은 양립성을 갖는다. 우리는

디바이스모델의 행위를 나타나기 위해 DEVS 형식론의 원자모델(atomic model)을 사용할 수 있다. 형식적으로 원자모델 M은 7가지의 요소에 의해 상세화된다.

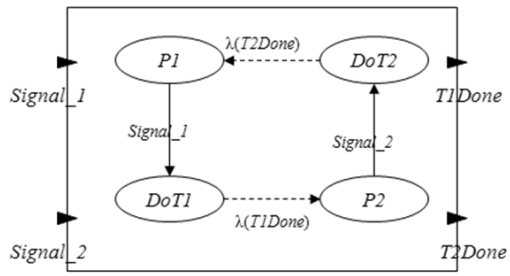
$M = < X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a >$
X : input events set;
S : sequencial states set;
Y : output events set;
δ_{int} : $S \rightarrow S$: internal transition function;
δ_{ext} : $Q * X \rightarrow S$: external transition function
$Q = \{(s,e)| s \in S, 0 \leq e \leq ta(s)\}$: total state of M;
λ : $S \rightarrow Y$:output function;
t_a : $S \rightarrow Real$: time advance function

7가지 요소 중 네 가지, 즉, δ_{int}, δ_{ext}, λ 그리고 t_a는 원자모델의 특징적인 함수이다. DEVS 형식론의 원자모델은 timed-FSA(finite state automata)로 여겨지며 이것은 디바이스 모델의 행위를 기술하기에 적합하다. 일단 디바이스 모델(공장모델)이 얻어지면 PLC 시뮬레이션의 수행이 가능하게 된다. 현재는 디바이스 모델이 직접 손으로 작성되어야 하며 이는 많은 시간과 노력을 소모한다. 이 문제점을 극복하기 위해 본 논문의 목적은 디바이스모델의 자동 생성 절차를 제안한다. 공장모델의 자동 생성 절차를 설명하기 전에 디바이스모델을 생성하는 수동의 절차를 살펴보자. 첫 번째로 디바이스 모델을 생성하기 위해 디바이스에 할당된 업무(task)의 정의가 필요하다. 일단 임의의 디바이스에 대한 업무가 파악되면 상태(state) 변화 다이어그램을 추출하는 것이 가능하며 이것은 DEVS 형식론의 원자모델을 정의한다. Fig. 4-(a)는 T1(p1에서 p2로의 이동)과 T2(p2에서 p1으로 이동), 두 가지 업무를 가진 AGV의 간단한 예를 나타낸다. 두 업무는 외부 이벤트에 의해 유발됨으로 AGV의 외형(shell)부분은 Fig. 4-(b)에 보이는 것처럼 *Signal_1*과 *Signal_2*라고 명시된 두 입력 포트를 가져야 한다. 업무의 셋으로부터 상태 변화 다이어그램을 예시하는 것이 가능하다. 이 예제에서는 *P1*, *DoT1*, *P2* 그리고 *DoT2*의 네 가지 상태가 존재한다. *P1*과 *P2*가 상태 변화를 위해 입력포트(*Signal_1*, *Signal_2*)로부터 외부 이벤트를 취하는 반면, *DoT1*과 *DoT2*는 두 업무(*T1*, *T2*)의 끝을 알리는 이벤트인 내부 이벤트를 취한다. AGV에 상응하는 가상 디바이스의 DEVS 원자모델은 다음과 같다.

Shell of a virtual device : $M = < X, S, Y, \delta_{int}, \delta_{ext}, \lambda, t_a >$



(a) Task identification of a device



(b) Atomic (device) model of the AGV

Fig. 4. Device model of an AGV

X : {*Signal_1*, *Signal_2*}
S : {*P1*, *DoT1*, *P2*, *DoT2*}
Y : {*T1Done*, *T2Done*}
δ_{int} (*DoT1*) = *P2*
δ_{int} (*DoT2*) = *P1*
δ_{ext} (*P1*, *Signal_1*) = *DoT1*
δ_{ext} (*P2*, *Signal_2*) = *DoT2*
λ (*DoT1*) = *T1Done*
λ (*DoT2*) = *T2Done*
t_a (*DoT1*) = *Time_1*
t_a (*DoT2*) = *Time_2*

일단 공장모델이 구축되면 PLC 프로그램의 직관적 검증을 가능하게 하는 PLC 시뮬레이션의 수행이 가능하다. Fig. 5는 PLC 프로그램과 공장모델 사이의 연결을 보여준다. 공장모델은 생산시스템의 모든 디바이스 모델을 포함하며, PLC 프로그램은 공장모델을 위한 제어로직을 포함한다. 공장모델과 PLC프로그램을 통합하기 위해 공장모델과 PLC프로그램 사이의 매핑관계의 정의가 필요하며 이를 I/O mapping이라고 한다. PLC 프로그램의 시각적인 검증을 가능하게 하기 위해서는 3D 그래픽 모델을 불러와야 하며 이 모델은 논리적 디바이스 모델(상태 변

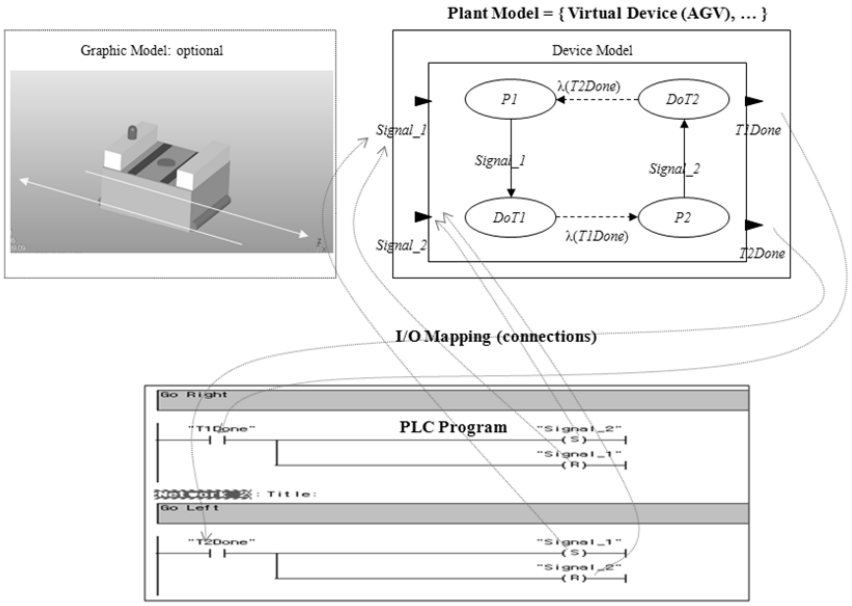


Fig. 5. Connections between a PLC program and a plant model

화 다이어그램)에 의해 제어된다. 3D 그래픽 모델은 항상 필요한 것이 아니기 때문에 PLC 시뮬레이션을 위해 선택적이다. 앞서 이미 언급했듯, 본 논문의 목적은 PLC 프로그램의 심볼 이름으로부터 디바이스모델을 추출하기 위한 것이다. 이를 위해 PLC 심볼에 대한 적절한 작명 규칙의 고안이 필요하다. 작명 규칙은 다음 장에서 기술된다.

3. 공장모델 생성을 위한 PLC 심볼 작명

비록 IEC 61131-3에서 PLC에 관한 다양한 표준 사항을 제공하고 있지만, PLC 심볼의 작명 규칙은 거의 초첨화 되지 못하였다. PLC 심볼 작명에 대한 어떤한 표준 규칙도 존재하고 있지 않기 때문에 이는 순전히 PLC 프로그램어 개개인의 취향에 의존한다.

PLC 심볼로부터 디바이스 모델을 생성하기 위해 PLC 심볼이 공장모델에 관한 충분한 정보를 포함하도록 하여야 한다. 이 목적을 위해 우리는 많은 PLC 프로그램어와 인터뷰를 수행했고 다양한 인습적인 규칙을 분석하였다. 결론적으로 우리는 다섯 개의 필드(field)를 가진 심볼 이름 구조를 고안하였다: (1) 라인 이름, (2) 공정 이름, (3) 디바이스 이름, (4) 입력 혹은 출력 신호, (5) 업무(상태) 이름. Fig. 6은 제안하는 PLC 심볼의 이름 구조를 보여준다.

PLC 심볼이 제안하는 작명 구조로 표기된다면 심볼

SYMBOL structure

Level 1	Level 2	Level 3	Level 4	Level 5

LINE__ProcessNum__Device__I/O__Task(State)

Level	Contents	example
Level 1	Line name	BB(body build)
Level 2	Process number	S301(station 301)
Level 3	Device name	RBT3(robot3)
Level 4	Input or Output	I or O
Level 5	Task	WELD(welding)

Fig. 6. Proposed structure for PLC symbol naming

이름을 단순히 분석함으로써 디바이스모델(DEVS 형식론의 원자모델)을 생성하는 것이 가능하게 된다. 심볼은 네 번째 필드에 의해 크게 입력과 출력의 두 종류가 존재한다. 출력 신호의 목적은 심볼의 다섯 번째 필드에 정의된 업무를 유발시키는 것이다. 따라서 출력 신호의 분석을 통해 특정 디바이스의 업무(상태)의 셋을 식별하는 것이 가능하게 된다. 앞서 언급했듯, 일단 디바이스의 업무의 셋이 정의되면 디바이스모델의 상태변화 다이어그램을 추출하는 것이 가능하며 이것이 DEVS 형식론의 원자모델을 정의한다. 출력 신호가 업무를 유발하기 위해

PLC로부터 오는 PLC 출력 신호인 반면, 입력 신호는 업무의 완료를 알리기 위해 PLC의 입력으로 가는 신호를 디바이스에 의해 발송되게 된다. 이점은 디바이스모델의 외부전이함수(external transition function(δ_{ext}))와 내부전이함수(internal transition function(δ_{int}))가 각각 출력과 입력 심볼로부터 자동으로 추출될 수 있음을 의미한다. Fig. 7에 보이는 예제를 사용하여 PLC 심볼로부터 공장모델의 생성절차를 설명해 보자.

예제 공정에서 원재료 파트는 작업자에 의해 AGV위에 놓여지게 된다고 가정한다. AGV가 파트의 존재를 감지하게 되면 이를 Machine으로 이송하기 위해 움직인다. 이송 후 Machine은 원재료파트의 가공을 시작한다. 예제의 경우 공장모델은 AGV와 Machine 두 개의 디바이스

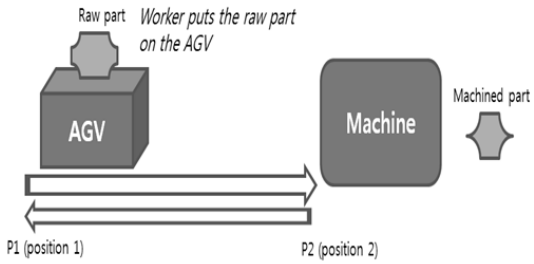


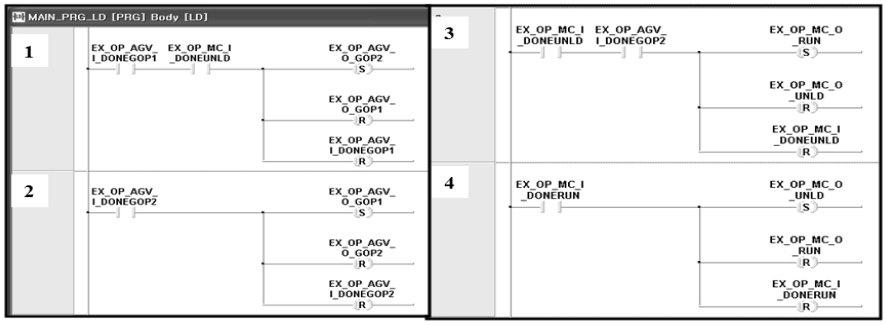
Fig. 7. A simple manufacturing cell consisting of two devices

모델을 갖는다. 예제 공정을 제어하기 위한 PLC 프로그램은 Fig. 8-(a)와 같으며 Fig. 8-(b)는 이것의 심볼테이블을 보여준다.

Fig. 8-(b)에 보이는 것처럼 AGV 모델은 두 개의 출력 신호와 두 개의 입력 심볼을 가지고 있다. 출력 심볼(EX_OP_AGV_O_GOP1, EX_OP_AGV_O_GOP2)로부터 우리는 AGV가 두 가지 업무, 즉 P2에서 P1으로의 이동과 P1에서 P2로의 이동, 을 가지고 있는 것을 직관적으로 인지할 수 있다. 출력 심볼을 사용하여 Fig. 9-(a)에 보이는 외부 변이 함수뿐만 아니라 상태 변화 다이어그램을 추출할 수 있다.

앞서 언급했듯 출력 심볼은 디바이스모델의 업무를 유발하게 되며 입력 심볼은 업무의 완료를 통지하기 위해 다바이스에 의해 생성된다. 업무의 이행은 디바이스에 의해 내부적으로 실행되기 때문에 디바이스 모델의 내부변이함수는 입력 심볼(EX_OP_AGV_I_DONEGOP1, EX_OP_AGV_I_DONEGOP2)로부터 쉽게 추출 된다. 이와 같은 방법으로 Machine의 디바이스 모델 또한 Fig. 9-(b)에 보이는 것처럼 관련된 신호(EX_OP_MC ..)로 부터 추출 될 수 있다. 디바이스모델의 구축 절차는 다음과 같다.

Step 1) 출력 심볼과 입력 심볼 사이에 서로 상응하는 모든 쌍을 찾는다. 출련 심볼이 업무를 유발하는 반



(a) PLC Program

Global Variable List

	Class	Identifier	MIT-Addr.	IEC-Addr.	Type	Initial
0	VAR_GLOBAL	EX_OP_AGV_I_DONEGOP1	X0	%IX0	BOOL	FALSE
1	VAR_GLOBAL	EX_OP_AGV_I_DONEGOP2	X1	%IX1	BOOL	FALSE
2	VAR_GLOBAL	EX_OP_MC_I_DONERUN	X2	%IX2	BOOL	FALSE
3	VAR_GLOBAL	EX_OP_MC_I_DONEUNLD	X3	%IX3	BOOL	FALSE
4	VAR_GLOBAL	EX_OP_AGV_O_GOP2	Y0	%QX0	BOOL	FALSE
5	VAR_GLOBAL	EX_OP_AGV_O_GOP1	Y1	%QX1	BOOL	FALSE
6	VAR_GLOBAL	EX_OP_MC_O_UNLD	Y2	%QX2	BOOL	FALSE
7	VAR_GLOBAL	EX_OP_MC_O_RUN	Y3	%QX3	BOOL	FALSE

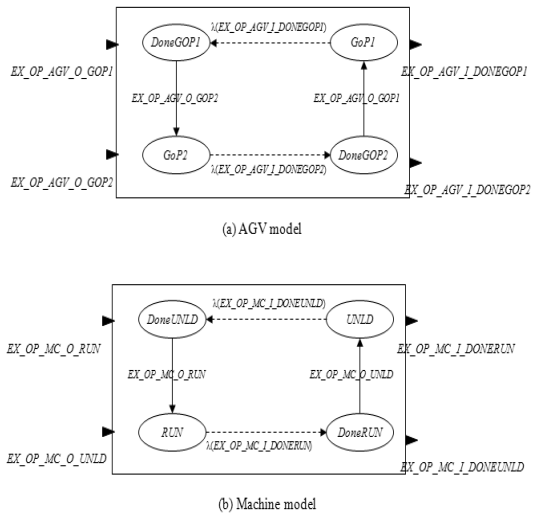
(b) Symbol table

Fig. 8. A PLC program to control the simple cell and its symbol table

면 대응하는 입력 심볼은 업무의 완료를 통지한다. 예를 들어 EX_OP_AGV_O_GOP1은 EX_OP_AGV_I_DONEP1에 상응한다.

Step 2) 입력과 출력 심볼의 마지막 이름 필드를 사용 하여 디바이스의 상태(state)를 정의 한다. AGV의 경우 우리는 네 가지 상태 즉, GoP1, DoneGoP1, GoP2, DoneGoP2를 정의 하였다.

Step 3) 출력(입력) 심볼을 사용하여 외부(내부) 변이 함수를 정의한다.



(a) AGV model

(b) Machine model

Fig. 9. Generated plant model from the symbol table

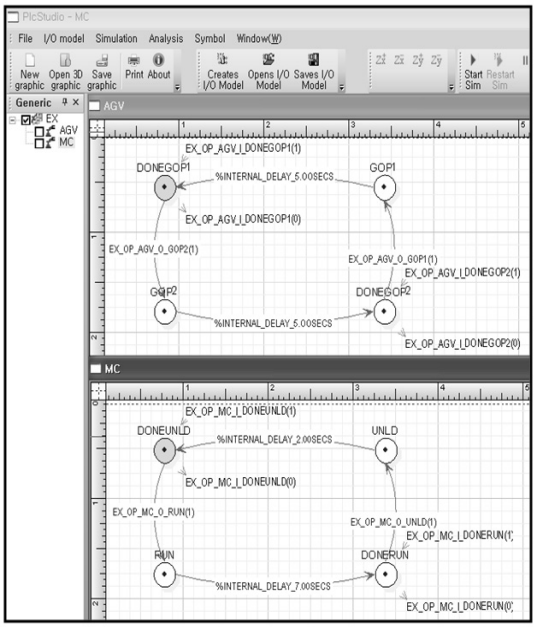


Fig. 10. The implementation of the proposed methodology

일단 공장모델이 얻어지면 PLC심볼과 공장모델사이의 I/O mapping을 정의함으로써 PLC 시뮬레이션이 가능하게 된다. PLC 시뮬레이션을 통하여 PLC 프로그램이 우리가 원하는 제어 목적을 달성하고 있는지 아닌지를 효율적으로 확인 할 수 있다.

논문이 제안하는 방법론은 C++ 프로그래밍 언어로 구현되었으며 시 구동은 Fig. 10에 보이는 것처럼 PC에서 실행되었다. Fig. 8-(a)의 PLC 프로그램은 Mitsubishi Electric Corporation이 제공하는 GX IEC developer 버전 7.0을 사용하여 프로그래밍 되었다. GX IEC developer는 PLC 프로그램에 사용된 심볼리스트를 엑셀(Excel) 포맷으로의 이출이 가능하다. 이출된 심볼 테이블은 디바이스 모델 생성을 위한 입력 요인이 된다. Fig. 10은 이출된 심볼 테이블을 분석하여 생성한 디바이스 모델을 보여 준다.

4. 토의 및 결론

PLC 시뮬레이션은 제어로직을 다양한 면에서 분석하는 것과 숨겨진 에러를 좀더 직관적으로 찾는 것을 가능하게 한다. 비록 PLC 시뮬레이션이 생산 시스템의 섬세한 검증을 위한 매우 강력한 도구임에도 불구하고 이에 수반되어야 하는 공장모델의 구축은 많은 시간과 노력이 요구된다. 이런 문제점을 극복하기 위해 본 논문은 PLC 프로그램의 심볼테이블로부터 공장모델을 생성하는 자동 절차를 제안한다. 또한 이를 위해, 심볼 이름이 공장모델에 대한 충분한 정보를 포함시키도록 PLC 심볼 작명 규칙을 제안한다. 심볼 이름을 분석함으로써 공장모델은 자동적으로 추출될 수 있다. 일반적으로 공장은 다양한 제조 설비들로 구성 되어있기 때문에 우리는 공장모델을 디바이스모델의 세트로 간주한다. 이런 디바이스 모델을 나타내기 위해 제안하는 방법은 Zeigler의 DEVS 형식론을 이용한다. DEVS 형식론의 원자모델은 디바이스모델의 논리적 행위를 나타내기 위해 사용된다. 다시 말해, 심볼 테이블로부터 DEVS 형식론의 원자모델 형태로 디바이스 모델을 추출하는 것이 반드시 필요하다.

비론 제안하는 방법론이 PLC 프로그램의 국부적인 검증을 다루고 있지만 공장의 기계적인 측면의 검증을 포함하도록 본 방법론의 확장이 가능하다[19,20]. Fig. 11에 보이는 것과 같이 우리는 제안하는 디바이스모델(DEVS 형

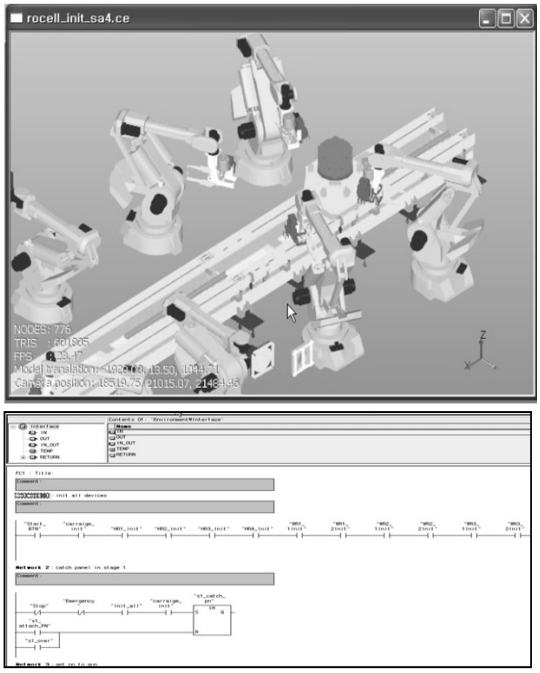


Fig. 11. Extended PLC simulation including mechanical verification.

식론의 원자모델)과 디바이스의 형상모델을 통합함으로써 물리적인 검증을 할 수 있다.

참 고 문 헌

1. H. Hibnio, T. Inukai and Y. Fukuda, “Efficient Manufacturing System Implementation Based on Combintion between real and virtual factory,” *International journal of Production Research*, Vol. 44(18), pp. 3897-3915, 2006.
2. A. M. A. Al-Ahmari, K. Ridgway, “An integrated modeling method to support manufacturing system analysis and design,” *Computers in Industry*, Vol. 38, pp. 225-238, 1999.
3. P. Klingstam, P. Gullander, “Overview of Simulation tools for Computer-aided production engineering,” *Computers in Industry*, Vol. 38, pp. 173-186, 1999.
4. Anglni A, Grieco A, Pacella M, Tolio T., “Object-oriented modeling and simulation of flexible manufacturing system: a rule-based procedure,” *Simulation Modeling Practice and Theory*, Vol. 10, pp. 209-234, 2002.
5. IEC (International Electro-technical Commission), *International Standard IEC 61131-3: Programmable Logic Controllers*, Second Edition, 2003.
6. D. Alexandre, B. Gerd, G. L. Kim, Y. Wang, A tool architecture for the next generation of UPPAAL, *http://www.uppaal.com/*; 2003.
7. G. L. Kim, P. Paul, Y. Wang, UPPAAL in a nutshell, *International Journal on Software Tools for Technology Transfer*, Vol. 1, No. 1/2, pp. 134-152, 1997.
8. M. Uzam, A. H. Jones, Discrete event Control system design using automation PetriNets and their ladder diagram implementation, *International Journal of Advanced Manufacturing System*, Vol. 14, No. 10, pp. 716-728, 1998.
9. E. Twiss and M. C. Zhou, Design of industrial automated system via relay ladder logic programming and Petri Nets, *IEEE Trans. On Systems, Man, and Cybenetics -Part C*, Vol. 28, No. 1, pp. 137-150, 1998.
10. R. David, H. Alla, Petri Nets for modeling of dynamic systems, *Automatica*, Vol. 30, No. 2, pp. 175-202, 1994.
11. G. Fray, Automatic implementation of Petri net based control algorithms on PLCs, *Proceeding of the American control conference ACC 2000*, Chicago; pp. 2819-23, 2000.
12. J. S. Lee, P. L. Hsu, A PLC-based design for the sequence controller in discrete event systems, *Proceedings of the 2000 IEEE International conference on Control Applications*, Ahchorage; pp. 929-34, 2000.
13. M. A. Jafari, T. O. Boucher, A rule-based system for generating a ladder logic control program from a high level system model, *Journal of Intelligent Manufacturing Systems*, Vol. 5, pp. 103-120, 1994.
14. S. Manesis, K. Akantziotis, Automated synthesis of ladder automation circuits based on state-diagrams, *Advances in Engineering Software*, Vol. 36, pp. 225-233, 2005.
15. C. P. Chuang, X. Lan, J. C. Chen, A systematic procedure for designing state combination circuits in PLCs, *Journal of Industrial Technology*, Vol. 15, No. 3, pp. 2-5, 1999.
16. M. P. Groover, Fundamentals of modern manufacturing, *Wiley*, 2006.
17. B. P Zeigler, Multifacetted modeling and discrete event *simulation, Academic Press*, Orland, 1984.
18. T. G. Kim, DEVSIM++ User's Manual, *Department of Electrical Engineering, KAIST, Korea*, 1994.
19. M. Onosato, K. Iwata, Development of a virtual manufacturing system by integrating product models and factory models, CIRP, Vol. 42, No. 1, pp. 475-478, 1993.
20. K. Iwata, M. Onosato, K. Teramoto, S. Osaki, A modeling and simuation architecture for virtual manufacturing systems, *CIRP*, Vol. 44, No. 1, pp. 399-402, 1995.

박 형 태 (taiji416@ajou.ac.kr)

2005 아주대학교 산업정보시스템공학부 학사
2007 아주대학교 산업공학과 대학원 석사
2007～현재 아주대학교 산업공학과 대학원 박사과정

관심분야 : Programmable Logic Controller(PLC), Simulation, CAD/CAM, FMS, Factory Automation

왕 지 남 (gnwang@ajou.ac.kr)

1983 아주대학교 산업공학과 학사
1985 한국과학기술원 산업공학과 석사
1992 미 Texas A&M 산업공학과 박사
1993～현재 아주대학교 산업정보시스템공학부 정교수

관심분야 : Intelligent Information & Manufacturing System, System Integration & Automation

박 상 철 (scpark@ajou.ac.kr)

1994 한국과학기술원 산업공학과 학사
1996 한국과학기술원 산업공학과 석사
2000 한국과학기술원 산업공학과 박사
2004～현재 아주대학교 산업정보시스템공학부 부교수

관심분야 : Digital Manufacturing System, CAD/CAM, CAPP, Manufacturing System Modeling & Simulation