확률 페트리 네트를 이용한 객체지향 기반의 칩마운터 시뮬레이터 구현

Object-Oriented Programming Based Chip-Mounter Simulator Using Stochastic Petri Nets.

박 기 범, 박 태 형
(Gi-Beom Park and Tae-Hyoung Park)

Abstract : An implementation method for chip-mounter simulator is proposed to improve the productivity and utility of electronic assembly lines. The simulator emulates the assembly sequence graphically to verify the chip mounter program in offline. It also presents functions of time estimation and productivity analysis considering the error probability. To increase the flexibility of simulator, stochastic petri nets are applied to modelling of the assembly sequence. The sequence model is then implemented as extendable classes by an object oriented language. The simulator is applied to a commercial chip mounter to verify the usefulness of the method proposed.

Keywords : electronic assembly system, chip mounter, simulator, stochastic petri nets

I. 서론

최근 휴대폰, 인터넷 등 정보통신제품의 수요가 급증함에 따라 이를 제조하는 전자조립라인의 중요성이 증대되고 있다. SMT(Surface Mount Technology) 인라인 시스템은 표면실장형의 전자부품을 인쇄회로기판에 조립하는 전자조립시스템으로서, 제품의 경박 단소화 추세 및 자동화의 수월성에 입각하여 정보통신제품 제조라인의 대부분을 점유하게 되었다[1]. 특히 소비자들의 새로운 제품에 대한 요구와 제품 라이프사이클의 단축에 따라 다품종 소량생산을 위한 유연 생산라인의 구축을 위한 노력이 진행되고 있다.

SMT 인라인 시스템을 구성하는 장비 중 칩마운터는 인쇄회로기판의 부품실장 위치에 수십~수백개의 부품을 실장하는 전용로봇이며, 라인의 생산성 향상을 위하여 가장 중요하게 고려되어야 하는 장비이다. 칩마운터의 조립작업을 위해서는 장착점 들의 위치, 부품공급 피더의 배열 및 장착 시퀀스를 포함하는 조립프로그램이 작성되어야 한다. 일련의 조립프로그램 작성과정은 매우 복잡하며 많은 시간이 소요될 수 있다. 따라서 장비의 가동을 계속 유지하면서 최적화된 조립프로그램을 작성할 수 있는 오프라인의 자동 프로그래밍 기능이 필요하며, 작성된 프로그램을 오프라인에서 사전 검증할 수 있는 시뮬레이터가 수반되는 것이 바람직하다[2]-[4].

시뮬레이터는 칩마운터의 조립진행과정을 그래픽으로 보여주어, 작성된 조립프로그램의 적절성을 사전에 검증할 수 있도록 한다. 또한 총 조립시간을 예측하여 라인의 균형도 및 단위 시간당 생산량을 사전에 판단할 수 있게 하며, 생산성 분석을 통하여 라인의 생산성을 높이기 위한 근거를 제시한다. 다품종 소량생산 중심으로 전환되고 있는 인쇄회로기판 제조라인의 생산성을 높이기 위하여, 오프라인에서 조립프로그램을 검증할 수 있는 시뮬레이터의 중요성이 증대되고 있다. 본 논문은 칩마운터 시뮬레이터의 체계적인 구현 방법을 제시한다.

칩마운터의 시뮬레이터와 모델링을 통한 생산성 분석에 관한 기존 연구들을 살펴보면, Grotzinger 등[4]은 로터리형 칩마운터에 시간 페트리 네트를 적용하여 흡착과 장착 사이클에 따라 모델링하고 페트리 네트 응용 프로그램을 이용하여 생산성을 분석하였다. Zhou 등[5][6]은 듀얼겐트리-싱글헤드 형 칩마운터를 일반 페트리 네트와 시간 페트리 네트로 나누어 적용하고, 장착시퀀스를 분류하여 그에 따른 페트리 네트 모델을 세우고 수학적 분석을 하였다. Sciomachen 등[7]은 싱글겐트리-싱글헤드형 칩마운터를 시간 페트리 네트를 이용하여 모델링 하고 페트리네트 응용 프로그램을 사용하여 작업시간을 추정하였다. 또한 칩마운터 시뮬레이터의 구현에 관한 연구는 칩마운터의 설정 및 조립프로그램에 대한 최적화 연구에 수반되어 제시되었다[8]-[10].

칩마운터는 부품공급 피더에서 부품을 흡착하여 인쇄회로기판에 부품을 장착하는 기본적인 메커니즘을 갖는다. 그러나 기구적 구조에 따라 로터리 형, 싱글겐트리 형 및 듀얼겐트리 형으로 분류될 수 있으며, 부품의 흡·장착을 수행하는 조립헤드의 수에 따라 싱글헤드 형 및 멀티헤드 형으로 분류될 수 있다[11]. 칩마운터의 조립시퀀스는 기구적 구조에 따라 많은 차이가 있으며, 또한 각 제조사의 모델에 따라서도 차이가 발생될 수 있다. 특히 듀얼겐트리-멀티헤드형 칩마운터는 최근 그 수요가 증가하고 있으나, 기구적 구조 및 시퀀스가 복잡하여 시간추정 및 생산성 분석이 상대적으로 어렵다. 또한 상호 겐트리의 충돌방지를 위한 대기 시퀀스와 여러 개의 헤드가 연계되어 동작하는 동시흡착 시퀀스 등이 추가로 고려되어야 한다.

본 논문은 새로이 듀얼겐트리-멀티헤드 형 칩마운터를

접수일자 : 2000. 11. 7., 수정완료 : 2001. 2. 20.
박기범 : (주) 삼성중공업(13omi@samsung.co.kr)
박태형 : 충북대학교 전기전자공학부(taehpark@chungbuk.ac.kr)

대상으로 확률 페트리 네트에 의하여 모델링 하고 그 생산성을 분석한다. 그리고 다른 형식의 칩마운터에 쉽게 확장될 수 있도록 하기 위하여, 확장성과 이식성을 갖춘 모델링 및 프로그래밍 방법을 적용한다. 또한 시뮬레이터 통합환경의 구현을 위한 방법을 제시한다. 페트리 네트로 모델링된 칩마운터를 페트리 네트 전용 프로그램을 가지고 시뮬레이션 하여 생산성 분석과 추정을 할 때, 조립프로그램의 특성만큼 페트리 네트 모델을 세워야 하는 단점이 있다. 따라서 객체지향형 언어를 사용하여 이식성을 갖춘 클래스로 페트리 네트를 구현하고 그래픽 환경을 도입하여, 이를 통합해 조립프로그램에 대응하는 시뮬레이터를 구현한다. 개발된 시뮬레이터는 다른 칩마운터 형식에 용이하게 확장될 수 있다. 시뮬레이션 결과 데이터는 칩마운터의 조립프로그램작성에 라인의 균형과 단위 시간당 생산량을 높이기 위한 자료로 활용되어 더 적절한 조립프로그램을 작성하는 바탕이 된다.

본 논문의 구성은 다음과 같다. II장에서는 적용 칩마운터의 소개, III장에서는 페트리 네트 모델링, IV장에서는 객체지향언어에 의한 페트리 네트 구현과 조립프로그램에 대한 오프라인 검증 및 성능평가와 분석을 한다. 마지막 V장에서는 본 논문에서 제안한 방법 및 결과 설명과 추후과제 등을 살펴본다.

II. 적용 시스템

표면 실장형 부품을 인쇄회로 기판 위에 실장 하는 칩마운터는 부품실장 메커니즘에 따라 로터리형(rotary type), 싱글겐트리형(single-gantry type), 듀얼겐트리형(dual-gantry type)으로 나눌 수 있다. 본 논문에서 적용한 칩마운터는 듀얼겐트리-멀티헤드형으로, 대응부품, 실장속도, 가격면에서 로터리형과 겐트리형의 장점을 취하고 있다. 대응 부품은 칩, IC, QFP부품 등이며, 실장속도는 고속이다. 최근 생산라인에서 매우 각광 받는 장비로 부각되고 있으나, 이 장비는 두 개의 프레임에 많은 조립헤드가 부착된 관계로 사용자의 조작 및 프로그래밍이 어려우며, 사용자가 작성한 프로그램에 따라 조립시간의 편차가 크게 발생될 수 있다[4].

그림 1은 듀얼겐트리-멀티헤드형 칩마운터의 구조를 보여준다. 듀얼겐트리-멀티헤드형 칩마운터의 부품 실장 메커니즘은 두개의 XY 운동을 하는 겐트리형 프레임에 다수 개(약 2~10개)의 조립헤드가 있는 헤드블록(Head Block)이 부착되어, 피더에서 부품을 흡착하고 PCB위에 부품을 장착하는 흡착과 장착(pick and place)의 사이클 운동에 의해 부품을 실장 하도록 구성되어 있다. 두개의 헤드 블록이 장착 위치인 PCB를 공유하는 관계로 블록간의 충돌을 회피하기 위하여 한 헤드 블록이 PCB상에 부품을 장착하는 동안 다른 헤드 블록은 그 외의 작업을 수행하거나 대기하는 상태가 된다. 또한 그림1에서 보이는 것과 같이 구조상 동작 영역에 대한 제약조건이 발생하게 되므로 전면(후면)의 헤드 블록은 전면(후면)의 피더에서 부품을 흡착하고 카메라와 노즐교환기도 전면(후면)에 설치된 것만 사용하게 된다.

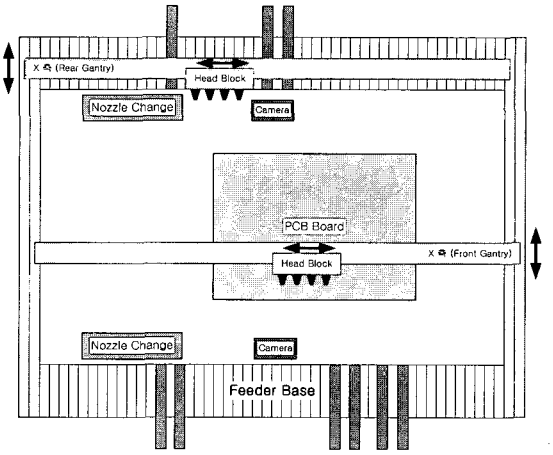


그림 1. 듀얼 겐트리형 칩마운터 구조.

Fig. 1. Structure of chip mounter for dual gantry type.

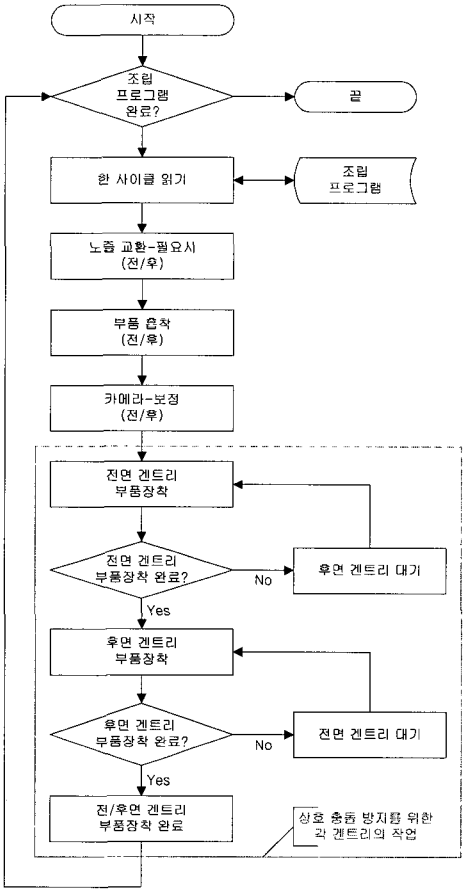


그림 2. 칩마운터의 작업 순서도.

Fig. 2. Flowchart of working sequence for chip-mounter.

칩마운터의 작업순서를 요약하면 그림 2와 같다. 한 사이클(cycle)은 두 개의 겐트리가 동시에 여러 헤드를 이용하여 각 부품에 맞는 노즐(nozzle)을 장착하고, 피더에서 부품을 흡착한다. 그리고 카메라로 영상을 취득하여 장착위치 및 각도 보정(centering)을 한 후 전면 겐트리가

PCB상의 장착위치로 이동하여 장착을 하는 동안 후면 겐트리는 대기 위치에서 전면 겐트리의 장착작업이 끝날 때까지 대기하고, 전면 겐트리의 장착 작업이 끝나면 후면 겐트리가 장착 작업을 하고 전면 겐트리는 대기하는 것으로 끝나게 된다. 칩마운터는 조립프로그램내의 데이터를 한 사이클씩 읽어 동작하게 되며, 조립프로그램 내의 모든 사이클이 수행될 때까지 반복하여 PCB조립을 끝낸다. 대부분의 칩마운터는 위와 같은 작업 순서를 가지고 있으며, 노즐 교환 방식이나 부품의 흡착, 흡착된 부품의 센터링 방식에서 각 제조사의 모델별로 차이가 있다. 일반적으로 칩마운터의 작업 순서에 따른 겐트리의 작업사이클별 자세한 동작 순서는 다음과 같다.

1) 노즐교환

Case 1 : 노즐 교환위치로 이동 →노즐 내려놓음 →설치할 노즐이 놓인 위치로 이동 →선택된 노즐을 설치 → 교환할 노즐의 수만큼 반복.

Case 2 : 노즐 교환 위치로 이동 →노즐 내려놓음 → 교환할 노즐의 수만큼 반복 →설치 노즐이 놓인 위치로 이동 →노즐 설치 →교환 노즐의 수만큼 반복.

2) 부품흡착

피더로 이동 →높이조절(Z축) →흡착 →높이조절 → 흡착할 수만큼 반복.

3) 부품 센터링

카메라 위치로 이동 →이미지 획득 →보정값 계산.

4) 부품 장착

부품 장착위치로 이동 →높이조절(Z축) →장착 →높이조절 →장착 부품 수만큼 반복.

Ⅲ. 페트리 네트 모델링

페트리 네트는 생산 시스템의 모델링, 분석, 시뮬레이션 그리고 컨트롤에 있어서 매우 유용하다는 것은 이미 입증되었다. 특히 페트리 네트의 다양한 확대는 자원의 활용, 실패율, 생산률 등의 효과에 대해 양적으로나 질적으로 분석이 가능하다. 확률 페트리 네트(Stochastic Petri Nets; SPN)는 시간과 트랜지션의 확률적인 동작으로 수량적인 관점에서 시스템을 모델하고 분석할 수 있는 도구로 TTPN(Timed Transition PN)이다[6][14].

확률 페트리 네트는 SPN, GSPN(Generalized Stochastic Petri Nets), ESPN(Extended Stochastic Petri Nets), DSPN(Deterministic Stochastic Petri Net)으로 분류할 수 있고, 이들은 트랜지션의 발화지연시간과 확률분포의 함수에 따라 분류된 것이다[9][10][12][13].

페트리 네트로 칩마운터의 조립시간의 계산과 실패율, 복구율에 따른 생산성을 평가하기 위해서 컨베이어의 운동시간, 겐트리의 운동시간, 헤드의 운동시간 등 조립프로그램의 사이클 데이터에 따라 다르게 계산되어 상수값을 가지는 시간지연 트랜지션과 컨베이어 실패/복구, 피더의 실패/복구, 흡착의 실패, 장착의 실패/복구에 따른 확률시간 지연 트랜지션을 사용 할 수 있는 DSPN으로 모델링 한다.

확률 페트리 네트에 의한 칩마운터의 페트리 네트 모델은 다음과 같이 표현된다.

$$SPN = (P, T, I, O, M^o, \Lambda) \quad (1)$$

단, P : 플래이스의 집합

T : 트랜지션의 집합

$I \subset P \times T$: 입력아크의 집합

$O \subset T \times P$: 입력아크의 집합

M^o : 초기 토큰의 집합.

Λ : 트랜지션 t_i 의 발화율의 집합.

1) 플래이스 $p_i \in P$ 와 트렌지션 $t_j \in T$ 는 다음 그림 3, 4와 같이 정의한다.

2) 정의된 플래이스와 트렌지션에 대한 입력아크 $x_{ij} \in I$ 는 0 또는 1 의 값을 갖고, 플래이스에서 트랜지션으로 입력이 있는 경우 1로 설정한다. 그 외 모두 0으로 설정한다.

3) 정의된 플레이스와 트렌지션에 대한 입력아크 $y_{ij} \in O$ 는 0 또는 1 의 값을 갖고, 트랜지션에서 플래이스로 입력이 있는 경우 1로 설정한다. 그 외 모두 0으로 설정한다.

4) 정의된 플레이스 p_i 에 대한 초기 토큰 $m_i^o \subset M^o$ 은 플래이스 P1과 MNTP3을 1로 하고 플래이스 P2, NZPi2, NZPij1, PKPi3, MNTPi3 (i, j = 1,2)은 각각 조립프로그램의 총 사이클 수, 첫 번째 사이클의 노즐교환 헤드 수, 첫 번째 사이클의 흡착 헤드 수, 첫 번째 사이클의 장착 헤드 수로 설정하고 그 외는 모두 0으로 설정하여 초기 토큰을 구성한다.

5) 정의된 트렌지션 t_j 에 대한 발화율 $\lambda_j \in \Lambda$ 는 트랜지션 LDFT, LDRT, PKFTi, FDFTi, FDRTi, MNTFTi, MNTRTi(i = 1,2)에 변수로 설정한다. 발화율은 시뮬레이터에서 매개변수 값으로 입력한다.

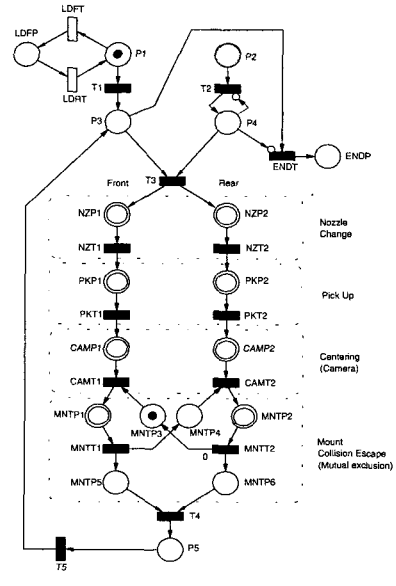


그림 3. 칩마운터의 페트리 네트 모델.

Fig. 3. The model of petri net for chip mounter.

표 1. 그림 3의 플래이스, 트랜지션 정의.

Table 1. The definition of place and transition for Fig. 3.

Place	
P1	PCB Load
LDFP	PCB Load Failure
P2	Total Cycle Count Load
P3	1 Cycle Start and End
P4	1 Cycle Ready
NZP1	Front Nozzle Station State
NZP2	Rear Nozzle Station State
PKP1	Front P ick Up State
PKP2	Rear Pick Up State
CAMP1	Front Camera (Vision) State
CAMP2	Rear Camera (Vision) State
MNTP1	Front Mount State
MNTP2	Rear Mount State
MNTP3	Front Mount Ready (Mutual Exclusion)
MNPT4	Rear Mount Ready (Mutual Exclusion)
MNTP5	Front Mount Done
MNTP6	Rear Mount Done
P5	One Cycle End
ENDP	1 PCB Assemble Done
Transition	
LDFT	Average time to fail for PCB Load
LDRT	Average repair time for PCB Load
T1	PCB Load Time
T2	Supply One Cycle Count
T3	Front/Rear Gantry Work Start
NZT1	Front Nozzle Change Done
NZT2	Rear Nozzle Change Done
PKT1	Front Pick Up Done
PKT2	Rear Pick Up Done
CAMT1	Front Camera (Vision) Done
CAMT2	Rear Camera (Vision) Done
MNTT1	Front Mount Done
MNTT2	Rear Mount Done
T4	Front/Rear Gantry Work Done
T5	One Cycle Done
ENDT	1 PCB Assemble Done

표 2. 그림 4의 플래이스, 트랜지션 정의.

Table 2. The definition of place and transition for Fig. 4.

Place		Transition	
NZPi1	Select Change Nozzle Type	NZTi1	Enter The Nozzle Station
NZPi2	Get Total Change Nozzle	NZTi2	Supply one Head
NZPi3	One Head	NZTi3	Supply one Head
NZPi4	Total Change Nozzle	NZTi11	Ready to Nozzle Change (One to One)
NZPi5	One Head	NZTi12	Move Time : Max(X, Y, Z)
NZPi11	Get Change Type (One to One)	NZTi13	Move Time : Max(R,Z) + ReleaseNozzleTime
NZPi12	Wait Position	NZTi14	Move Time : Max(X, Y, Z)
NZPi13	Nozzle Station Change Position	NZTi15	Move Time : Max(R,Z) + PickNozzleTime
NZPi14	Release Nozzle	NZTi16	Move Time : Max(X, Y, Z)
NZPi15	Nozzle Station Change Position	NZTi17	Nozzle Change Done
NZPi16	Pick Nozzle (Change)	NZTi21	Ready to Nozzle Change (All)
NZPi21	Get Change Type (All)	NZTi22	Move Time : Max(X, Y, Z)
NZPi22	Wait Position	NZTi23	Move Time : Max(R,Z) + ReleaseNozzleTime
NZPi23	Nozzle Station Change Position	NZTi24	Move Time : Max(X, Y, Z)
NZPi24	Release Nozzle	NZTi25	Nozzle Release Done/Ready to Pick Nozzle
NZPi25	Nozzle Station Change Position	NZTi26	Move Time : Max(R,Z) + PickNozzleTime
NZPi26	Pick Nozzle (Change)	NZTi27	Move Time : Max(X, Y, Z)
NZPi6	Nozzle Change Out	NZTi28	Nozzle Change Done
PKPi1	From ANC Position	PKTi1	Ready to Pick
PKPi2	Feeder Position	PKTi2	Moving Time : Max(X,Y,Z)
PKPi3	Get Total Pickup Components	PKTi3	Supply One Component
PKPi4	One Component	PKTi4	Moving Time : Z + PickDelayTime
PKPi5	Pick Up	PKTi5	Moving Time : Max(X,Y,Z,R)
PKPi6	Pick Up Complete	PKTi6	Pick Up Complete
PKFPi	Fail to Pick Up Component	PKFTi	Average Time to Fail for Pick Up
FDFPi	Change Feeder	PKRTi	Repair time for Pick Up
		FDFTi	Average time to Change for Feeder
		FDRTi	Average Change time for Feeder
CAMPi1	Pick Up Position	CAMTi1	Ready to Vision Work
CAMPi2	Image Grabbing	CAMTi2	Moving Time : Max(X,Y,Z)
CAMPi3	Vision Complete	CAMTi3	Image Grabbing Time
MNTPi1	From Camera Position	MNTTi1	Ready to Mount
MNTPi2	Mount Position	MNTTi2	Moving Time : Max(X,Y,Z)
MNTPi3	Get Total Pickup Components	MNTTi3	Supply One Component
MNTPi4	One Component	MNTTi4	Moving Time : Z + PlaceDelayTime
MNTPi5	Mount	MNTTi5	Moving Time : Max(X,Y,Z,R)
MNTPi6	Mount Complete	MNTTi6	Mount Complete
MNTFPi	Fail to Mount Component	MNTFTi	Average Time to Fail for Mount
		MNTRTi	Average Repair time for Mount

(i = 1,2)

그림 3은 정의된 플래이스, 트랜지션 및 아크에 의하여 구성된 칩마운터 전체 페트리네트 모델이다. 전체 페트리 네트 모델에서 노즐교환, 부품흡착, 센터링, 부품장착의 동작은 하나의 플래이스로 표현하고 이는 곧 매크로 플래이스에 해당한다. 각각의 매크로 플래이스에 대한 모델은 그림 4에 나타나 있다.

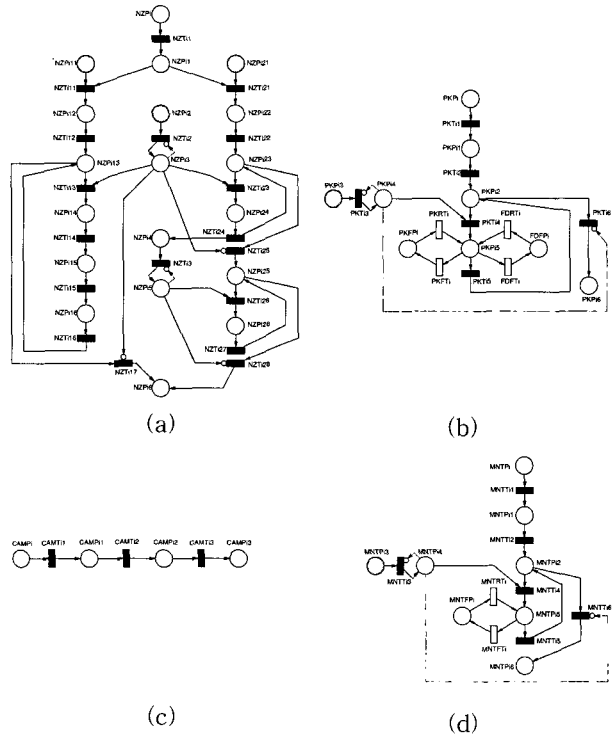


그림 4. (a) 노즐교환, (b) 부품흡착, (c) 센터링, (d) 부품장착.

Fig. 4. (a) Nozzle change, (b) Pick up, (c) Centering, (d) Place.

듀얼 겐트리-멀티 헤드형 칩마운터의 모델링에서 특히 고려해야 할 사항은 두 가지가 있다. 첫째로 두 개의 겐트리가 한 개의 PCB를 공유하면서 작업하므로 겐트리간 충돌방지를 위한 대기상황과, 둘째로 다수의 조립헤드에 대해 흡착할 부품의 피더 위치가 헤드의 간격과 동일하여 동시 흡착이 가능할 경우에 대한 동시흡착 상황을 고려해야 한다.

듀얼 겐트리형 칩마운터는 두개의 겐트리가 각각 별도로 작업을 진행하여 빠른 흡착 및 장착작업을 하지만 부품의 장착 작업에서 PCB를 공유하기 때문에 전면(후면) 겐트리가 장착작업을 하는 동안 후면(전면) 겐트리가 같은 PCB영역으로 진입을 하면 두 겐트리가 충돌하게 된다. 따라서 후면(전면) 겐트리는 대기 또는 다른 작업을 한다. 이런 겐트리간 충돌문제는 페트리 네트 모델에서 상호배제구조(mutual exclusion structures)를 이용한다. 전체 페트리 네트 모델에서 플래이스 MNTP3 과 MNTP4는 두 겐트리의 충돌방지를 조정하는 플래이스라 할 수 있다. 또한 다수의 조립 헤드로 피더에서 부품을 흡착할 때 흡착 시간의 단축을 위해 최소한의 흡착동작으로 많은 부품을 흡착하려는 동시흡착은 헤드의 간격과 흡착할 부품이 있는 피더의 배치가 동시흡착을 고려하여 프로그램 되어 있을 때 발생한다. 동시흡착 문제는 작업 사이클별로 동시흡착을 고려한 사용 가능한 헤드의 수로 토큰을 설정하여 모델링 함으로 동시흡착에 대한

페트리 네트 모델을 간단히 할 수 있다. 작업 사이클별로 동시흡착을 고려한 사용 가능한 헤드의 수는 조립프로그램의 분석을 통하여 사이클별로 계산하여 설정한다.

페트리 네트로 모델링된 칩마운터는 조립프로그램과 밀접한 관계를 가진다. 즉 칩마운터의 조립프로그램에 따라 초기토큰과 트랜지션의 지연시간이 바뀌게 되고 모델의 특성이 바뀌게 된다. 칩마운터 페트리 네트 모델의 동적 특성은 일반적으로 조립프로그램에 대해 초기 마킹의 개수가 결정적(bounded)이고, 생존(live)하며, 조립프로그램의 사이클이 끝날 때 교착상태(deadlock)가 된다. 충돌(conflict)은 금지 아크에 의해서 대부분 발생하지 않지만 플래이스 P1, PKPi5, MNTPi5(i = 1,2) 에서 발생되며, 이는 미리 정해진 규칙에 따라 임의의 트랜지션을 선택한다. 충돌상황은 확률 페트리 네트에서 로딩실패/성공, 흡착실패/성공, 피더교체/진행, 장착실패/성공 등의 사건을 정의하기 위하여 사용된다.

확정된 트랜지션 시간은 조립프로그램의 겐트리 속도 데이터와 겐트리의 위치, 헤드의 Z축 속도 데이터와 Z축 위치데이터 등에 따라 사이클 별로 계산하여 설정한다. 확률 트랜지션은 그림 3, 4와 같이 LDRT, PKRT, FDRT, MNTRT의 트랜지션에 사용한다. 이것은 각각 PCB로딩의 실패 시 PCB를 다시 로딩하는데 걸리는 시간과, 흡착 실패 시 재흡착 하는데 걸리는 시간, 피더의 교체에 걸리는 시간, 장착 실패 시 재장착 까지 걸리는 시간이다.

본 논문에서 제시한 칩마운터의 페트리 네트 모델은 두 겐트리 사이의 충돌문제와 멀티 헤드의 동시흡착문제를 고려한 확률 페트리 네트 모델이다.

IV. 객체지향 언어에 의한 페트리 네트 구현

1. 객체지향 언어에 의한 페트리 네트 구현

객체 지향 언어는 모든 프로그래밍 방법을 객체로 설정하여 이것을 운용하는 방식으로 설계하는 언어이다. 객체 지향 언어는 추상화, 캡슐화, 상속성, 다형성, 메시지 등의 특성을 가지고 있으며, 클래스는 공통의 특성을 가진 객체의 집합이다. 객체는 그의 외관상의 모습과 기능 또는 속성들이 함께 존재하는 형태를 가진다. 따라서 객체지향 언어에서는 데이터 영역(member)과 연산자 영역(method)으로 구분되어 다뤄진다. 데이터 영역은 객체가 가지는 정보를 보관하는 장소로 활용되며 실제적으로는 변수의 형태를 가진다. 객체의 또 하나의 구성 요소인 연산자 영역은 이 객체의 데이터 영역을 조작하여 저장된 값을 바꾸거나 정보를 밖으로 빼내는 작업등의 데이터를 다루는 작업을 한다[15]-[17].

플레이스, 트랜지션, 아크등 객체로 설정 되어진 페트리 네트 요소들은 페트리 네트 클래스에 연결되어 동작한다. 페트리 네트 클래스는 페트리 네트 객체들의 데이터를 총괄적으로 관리하고 사용한다. 이렇게 클래스화된 페트리 네트는 칩마운터의 시뮬레이터 뿐만 아니라 다른 시스템의 페트리 네트 모델에도 쉽게 이식되어 사용 가능하다.

본 논문의 페트리 네트 클래스(CPetriNet)는 객체지향 언어에 기반을 두고 제작되었다. 그림 5는 CPetriNet의 클래스 정의부 이며, 클래스를 구성하는 멤버함수와 멤버변수들을 묘사한다. 데이터 은닉(Data Hiding)은 객체지향형 프로그래밍의 특성 중 하나인 캡슐화의 한 예로 사용자가 데이터에 직접 접근하지 못하게 할 뿐만 아니라 데이터가 표현되는 방식에 신경 쓰지 않도록 해준다. 공용 선언(public)되어 있는 멤버들은 외부에서 사용자가 사용할 수 있는 함수나 데이터이고, 개별 선언(private)되어 있는 멤버들은 외부에서 사용자가 사용할 수 없는 함수나 데이터이다. 표 3은 공용 선언된 멤버함수의 설명이다.

표 3. CPetriNet클래스의 공용 멤버함수 설명.

Table 3. Explanation of public member function for the CPetriNet.

멤버함수	내용
CPetriNet()	CPetriNet Class 생성자(멤버변수를 초기화 한다.)
~CPetriNet()	소멸자(멤버변수를 초기화 한다.)
ReadPN()	페트리 네트 데이터 파일 읽는다.
InitAllPetriNet()	멤버클래스의 모든 데이터를 초기화 한다.
InitPN()	페트리 네트를 초기화 한다.(초기 토큰 설정)
AddFireList()	현재 발화 가능한 모든 트랜지션을 발화목록에 넣는다.
Firing()	발화목록을 바탕으로 발화 시키며, 토큰의 이동이 있다.
AddCycleTime()	멤버클래스ClassPNTIME에 시간 데이터를 추가한다.
AddTotalTime()	멤버변수 m_fTotalTime에 데이터를 추가한다.
SetCycleTime()	멤버변수 m_fCycleTime에 데이터를 설정한다.
SetStopPN()	멤버변수 m_bStopPN에 데이터를 설정한다.
SetToken()	멤버클래스ClassPNPLACE에 토큰 데이터를 설정한다.
SetTime()	멤버클래스ClassPNTRANS에 시간 데이터를 설정한다.
GetTime()	멤버클래스ClassPNTRANS에서 시간 데이터를 가져온다.
GetCycleTime()	멤버변수 m_fCycleTime의 I 사이클의 시간값을 가져온다.
GetCycleTime(index)	멤버클래스ClassPNTIME에서 참조값의 데이터를 가져온다.
GetTotalTime()	멤버변수 m_fTotalTime의 총 시간데이터를 가져온다.
GetTotalFiring()	멤버변수 m_wTotalFiring에서 총 발화횟수를 가져온다.
GetPlaceID()	멤버클래스ClassPNPLACE에서 플레이스의 참조값을 가져온다.
GetTransitionID()	멤버클래스ClassPNTRANS에서 트랜지션의 참조값을 가져온다.
GetToken()	멤버클래스ClassPNPLACE에서 플레이스의 토큰개수를 가져온다.
IsEnable()	멤버클래스ClassPNPLACE에서 플레이스가 활성화 인지를 판단한다.

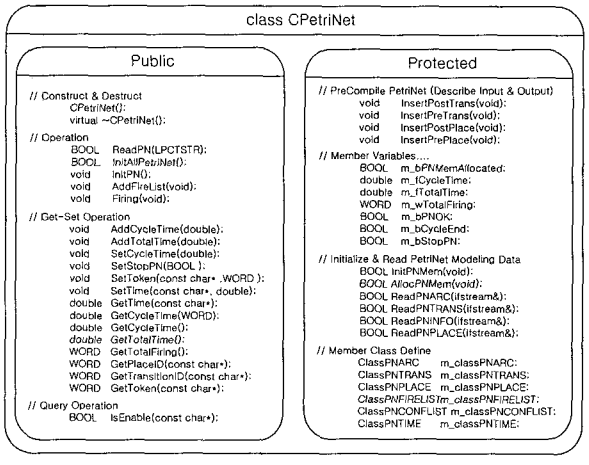


그림 5. 페트리 네트 클래스(CPetriNet)의 헤더파일 구조.

Fig. 5. The header file structure of CPetriNet.

그림 6은 응용 프로그램 내에서 CPetriNet이 동작하는 일반적인 모습을 나타낸다. 칩마운터의 페트리 네트 모델은 CPetriNet클래스를 사용하기 위하여 텍스트 형식의

데이터 파일로 작성한다. 이 텍스트 형식의 데이터 파일은 페트리 네트의 플래이스와 트랜지션 그리고 아크 데이터로 구성된다. 플래이스 데이터는 플래이스 번호, 플래이스 이름, 토큰의 수(초기마킹)로 나열하고 트랜지션 데이터는 트랜지션 번호, 트랜지션 이름, 발화지연시간(초기값 = 0)으로 나열하며, 아크 데이터는 플래이스에서 트랜지션방향을 순방향으로 하여 아크번호, 방향, 아크 속성(금지아크) 등을 나열하여 모델링 파일을 작성한다.

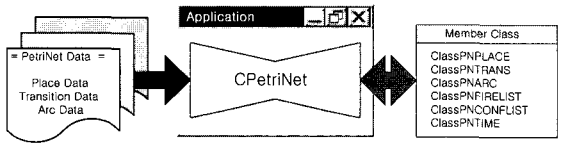


그림 6. 응용 프로그램에서 CPetriNet 클래스의 동작.
Fig. 6. Behavior of CPetriNet in application program.

이렇게 작성된 페트리 네트 모델링 파일은 CPetriNet에서 읽으면서 전처리 과정을 통해 CPetriNet의 멤버클래스인 ClassPN*에 맞는 데이터로 저장된다. CPetriNet는 멤버클래스인 ClassPN*과 데이터를 주고받으며 동작하게 된다. 그림 7은 CPetriNet의 사용 예 이다.

```
CPetriNet m_classPN    ; // Member클래스로 선언.
...
void CSimMPSDoc::OnTimeSimul()
{
// Get PetriNet Modeling File Path.....
        CString strPNPath = m_strPrjPath;
        strPNPath.TrimRight(m_strPrjName);
        strPNPath += "model.ptn";
        if(! m_classPN.ReadPN(strPNPath)    ) return FALSE;

// The Init Special Place
        m_classPN.SetToken("P2",m_classPLACE.SetTotalCycleNo()+1)          ;

// Time Simulation
  ...
        while(! m_classPN.IsEnable("ENDP")    )
        {
                m_classPN.AddFireList()    ;
                m_classPN.Firing()    ;
                ...
        }
  ...
}
```

그림 7. CPetriNet의 사용 예.
Fig. 7. Example of using CPetriNet.

객체지향 프로그래밍으로 제작된 CPetriNet클래스는 정의된 기본 기능 이외에 사용자가 원하는 기능을 추가하여 사용할 수 있고, 다른 시뮬레이터나 응용프로그램을 CPetriNet클래스를 멤버클래스로 추가하거나 상속하는 것으로 쉽고 빠르게 제작할 수 있다.

2. 통합 환경 구성

본 논문에서 개발한 칩마운터 시뮬레이터는 크게 두 가지 기능을 가지고 있다. 첫째 조립프로그램에 의한 칩마운터의 동작을 묘사하고, 둘째 페트리 네트 모델에 의해 생산성을 평가한다. 칩마운터의 동작 묘사는 조립프로그램에 따른 피더의 배치 및 장착위치 표시 그리고 겐트리가 이동을 하면서 실장 하는 동작을 묘사한다. 사용자 화면은 그림 8과 같이 세 개의 창(window)으로 나뉘며 조립프로그램의 사이클 데이터를 표시하는 창, PCB 확대 창, 전체 칩마운터를 표시하는 창이다. 사이클 데이터를 표시하는 창은 노즐, 피더, 장착점의 정보를 표시한다.

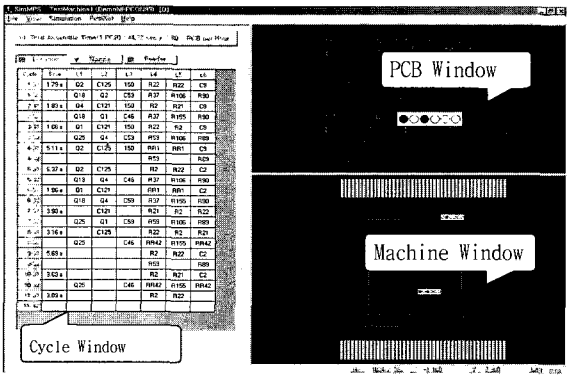


그림 8. 시뮬레이터의 주화면.
Fig. 8. Main window of the simulator.

그림 9는 시뮬레이터의 메뉴 구성을 나타낸 것이다. 메뉴는 [File | View | Simulation]으로 구성되며 각 메뉴의 기능은 다음과 같다.

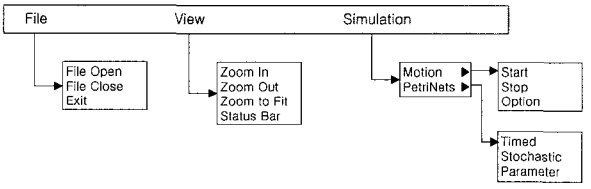


그림 9. 사용자 메뉴.
Fig. 9. User menu.

1) File은 시뮬레이터에서 조립프로그램을 열거나 닫고, 프로그램을 종료하는 기능을 제공한다.

2) View는 상태바를 보여주거나 감추며, PCB창과 머신창의 확대, 축소, 맞춤 등의 보기에 관한 기능을 제공한다. 또한 이 기능은 마우스의 오른쪽 버튼을 눌러 나오는 팝업 메뉴(Popup menu)에도 제공한다.

3) Simulation은 [Motion]과 [PetriNet]의 하위메뉴로 구성되며 조립프로그램에 의한 칩마운터의 동작묘사와 페트리 네트로 작업시간 추정, 생산성 평가를 하는 기능을 제공한다. [Motion]은 [Start | Stop | Option...] 의 하위메뉴로 나뉘며 동작묘사의 시작, 종료, 동작묘사 속도조절의 기능을 제공한다. [PetriNet]은 [Time | Stochastic | Parameter]의 하위메뉴로 나뉘며 1개의 PCB를 조립하는데 걸리는 시간과 1시간에 조립하는 PCB의 수를 추정하고, 확률변수에 의한 생산성을 평가하며 확률시뮬레이션에 필요한 변수를 설정한다.

그림 10은 시뮬레이터 동작의 예시화면이다. 동작 시뮬레이션은 조립프로그램의 부품 배치, 피더 배치, 실장 시퀀스 등을 가시적으로 판단할 수 있다. 그림 11은 확률

페트리 네트로 생산성 분석을 할 때 필요한 시뮬레이션 시간, 실패율, 복구율 등의 확률 파라미터 값을 설정하는 대화 상자이다.

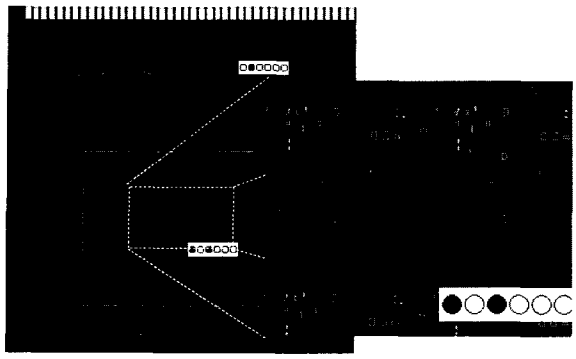


그림 10. 시뮬레이터 동작 예.
Fig. 10. Example of simulator execution.

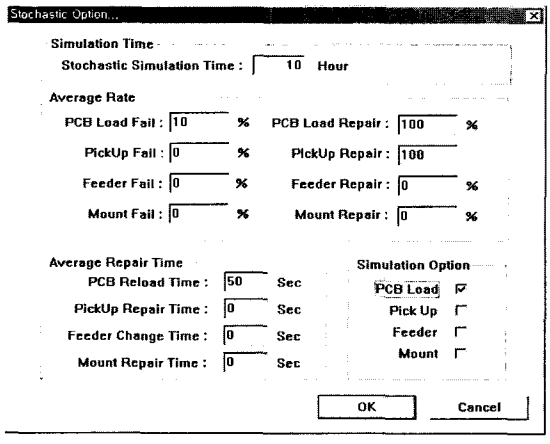


그림 11. 확률 파라미터 대화상자.
Fig. 11. Stochastic parameter dialog.

시뮬레이터는 오프라인에서 조립프로그램의 적절성을 체계적으로 검증할 수 있다. 시뮬레이터는 칩마운터의 피더 배치, 실장 시퀀스 등을 보여주고, 생산성 추정과 분석을 하여 조립프로그램을 시각적, 분석적으로 검증할 수 있기 때문에 조립프로그램 및 작업조건 수정에 소요되는 시간과 비용을 감소시키는 효과가 있다.

3. 페트리 네트 시뮬레이션 결과

페트리 네트에 의한 시뮬레이션은 본 논문에서 제작한 페트리 네트 클래스(CPetriNet Class)를 사용하여 시뮬레이션 하였고, 조립프로그램에 의한 작업시간 시뮬레이션과 생산성 시뮬레이션으로 나뉜다.

그림 12는 CPetriNet 클래스를 사용 한 작업시간 시뮬레이션의 순서도 이다. 시뮬레이션 순서를 살펴보면 먼저 CPetriNet을 이용하여 페트리 네트 모델을 읽는다. 읽어온 모델 파일은 CPetriNet에서 요구하는 파일 형식에 따라 텍스트로 작성되었다. 조립프로그램의 작업 데이터를 사이클 별로 읽어 해당 사이클에 사용하는 헤드의 수는 초기 토큰으로 설정하고 컨베이어와 겐트리 그리고 헤드의 움직임에 의한 시간은 트랜지션의 시간 데이터로 설정하여 시뮬레이션 한다. 한 사이클이 끝날 때 작업시간이 합산되고 이렇게 각 사이클 별로 시뮬레이션한 시간은 전체 조립프로그램이 끝날 때 총 작업시간으로 합산이 된다. 시간 시뮬레이션은 칩마운터의 오류가 없다고 가정하여 한 개의 PCB를 조립하는데 걸리는 시간과 단위 시간당 생산량을 시뮬레이션 한다.

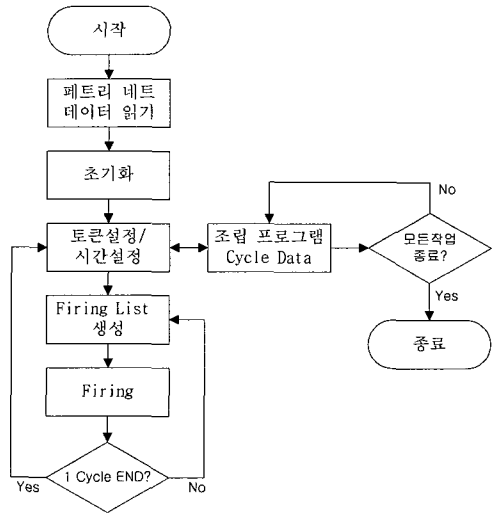


그림 12. 시간 시뮬레이션 순서도.
Fig. 12. The flowchart of time simulation.

생산성 시뮬레이션은 작업시간 시뮬레이션을 하고 그 데이터를 바탕으로 확률 페트리 네트를 적용한다. PCB 로드의 실패율과 복구율, 부품흡착의 실패율, 부품장착의 실패율과 복구율, 피더 실패율 등의 데이터는 사용자 입력데이터이고, 실패율과 복구율은 1시간당 백분율로 입력한다. 각각의 작업 실패시점은 비율에 따라 무작위로 발생되며, 평균 복구시간은 실패율에 의한 실패시간과 복구율에 따라 지수분포로 나타난다. 그림 13은 생산성 평가 결과의 예시 화면이다.

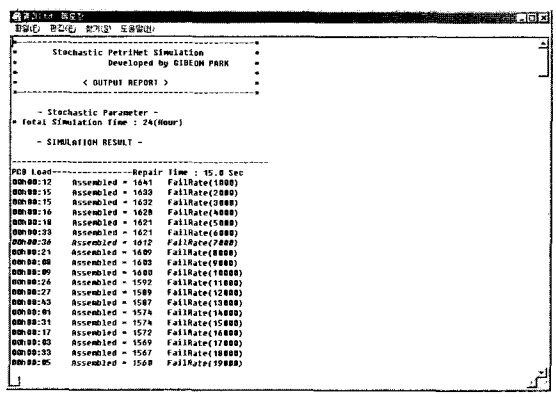


그림 13. 결과출력 화면.
Fig. 13. Screen of the output report.

시뮬레이터는 PC Windows 환경에서 동작하며, Microsoft® 의 Visual C++ 를 사용하여 제작하였다. 장착점 100개의 조립프로그램에 대해 10시간 작업의 생산성을 시뮬레이션 할 경우 시뮬레이션 시간은 10초 정도이다. 듀얼 겐트리형 칩마운터로 장착점 수가 100개인 PCB를 조립할 때 시간당 생산성 변화와 각 작업의 실패와 복구율에 따른 생산성 변화 그래프는 다음 그림 14~16에 나타나 있다.

그림 14는 작업시간 10시간 동안 PCB 진입실패, 흡착실패, 피더교환, 장착실패의 사건이 일어날 확률의 변화에 따른 생산성 변화 그래프로 각 실패에 따른 복구율은 100%로 설정하고 복구 시간은 입력받은 평균 복구시간의 실패율과 복구율에 따른 지수분포 내의 시간이다. 그래프를 살펴보면 피더의 교체가 가장 생산성에 큰 영향을 미치며 흡착실패는 자동으로 복구되므로 생산성에 큰 영향을 미치지 못하는 것으로 판명된다.

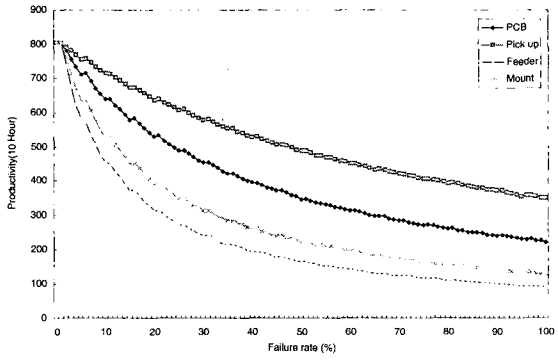


그림 14. 실패율에 따른 생산량.

Fig. 14. A mount of production versus failure rate.

그림 15는 작업시간 10시간 동안 PCB 진입실패, 흡착실패, 피더교환, 장착실패의 사건이 동일한 비율로 발생할 때 이들의 복구율의 변화에 따른 생산성 변화 그래프이다. 어떤 에러에 대해서도 복구율이 0%일 경우 1개의 PCB도 조립하지 못하는데 이는 그래프에 나타나 있다. 부품흡착은 칩마운터에서 자동 복구가 되므로 생산성에 영향이 거의 없고, 피더의 교체는 복구 시간이 많이 소요되는 에러이므로 생산성에 큰 영향을 미친다.

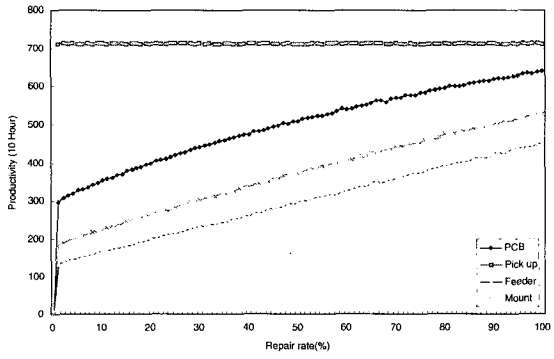
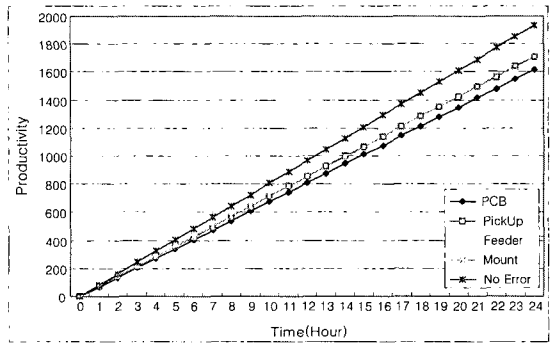


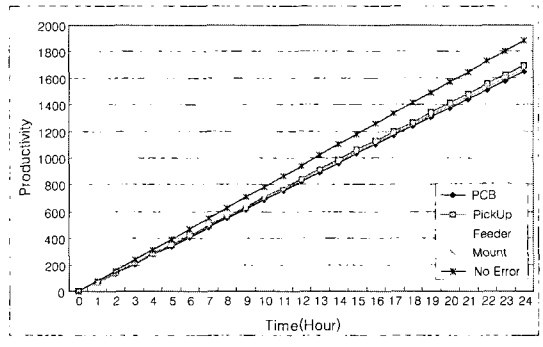
그림 15. 복구율에 따른 생산량.

Fig. 15. A mount of production versus repair rate.

그림 16은 실제와 흡사한 경우에 시간당 생산성 변화와, 조립프로그램의 동시흡착의 효과에 대한 결과 그래프이다. (a)는 동시흡착이 고려된 조립프로그램에 의한 결과이고, (b)는 동시흡착이 고려되지 않은 조립프로그램의 결과이다. 칩마운터에서 일반적으로 부품흡착, PCB 진입, 부품장착, 피더교체의 순으로 에러가 발생하고, 에러의 복구시간은 피더교체, PCB 진입, 부품장착, 부품흡착의 순으로 복구시간이 많이 소요된다. 결과 그래프는 위의 일반적인 에러 발생률과 복구시간을 입력으로 하여 나온 결과이다. 시간이 경과함에 따라 각각의 에러에 따른 효과가 증가하며 에러의 효과를 조금 더 명확히 하기 위하여 에러가 없는 상황을 비교 대상으로 하였다. 생산성에 피더의 교환은 낮은 에러발생에도 불구하고 복구시간이 많이 소요되므로 큰 영향을 미치며, 높은 에러율을 가진 부품흡착은 복구시간이 짧고 자동복구의 이점으로 생산성에 큰 영향은 없다.



(a) 동시흡착



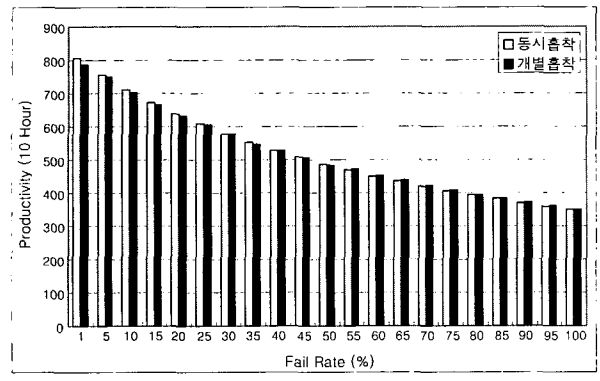
(b) 개별흡착

그림 16. 시간당 생산량.

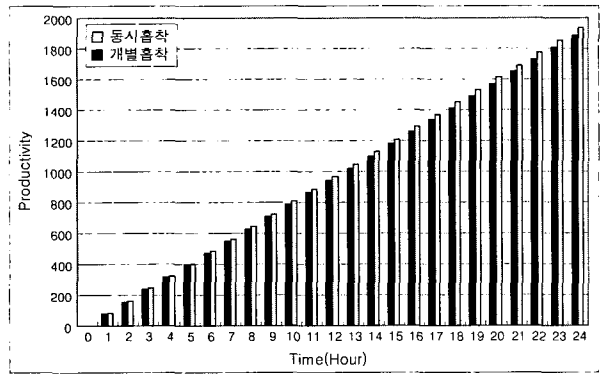
Fig. 16. A mount of production per hour.

그림 17은 칩마운터 조립프로그램의 동시흡착 효과에 대한 결과 그래프이다. (a)는 동시흡착과 개별흡착의 흡착실패에 따른 생산성 변화 그래프이고, (b)는 동시흡착과 개별흡착의 시간에 따른 생산성 변화 그래프이다. 그림 17의 (a)를 보면 실패율 50% 이상부터 동시흡착의 효과는 감소하며, 그 이하 일 때 동시흡착은 개별흡착보다 생산성이 높은 것을 알 수 있다. 그림 17의 (b)는 작업 시간이 증가할 수록 동시흡착의 효과가 증가함을 알 수 있

다. 결과적으로 동시흡착을 고려한 조립프로그램은 칩마운터의 생산성 향상에 영향을 준다. 이는 칩마운터 조립프로그램에 따른 생산성의 변화를 단적으로 보여 주는 것으로 이런 조립프로그램의 적절성은 시뮬레이터를 통하여 사전에 검증된다.



(a) 피터의 실패율에 따른 생산량



(b) 부품흡착에 의한 시간당 생산량

그림 17. 동시흡착의 효과.
Fig. 17. The effect of simultaneously pick up.

칩마운터의 작업에서 생산량에 가장 큰 영향을 주는 에러는 피더 교체이다. 피더의 교체는 다른 에러에 비하여 매우 낮은 비율로 발생 하지만 복구하는 시간이 다른 에러에 비하여 많이 걸린다. 부품의 흡착은 타 에러에 비하여 발생률이 높지만 칩마운터에서 자동 복구를 하므로 시뮬레이터 상에서는 생산성 변화에 큰 영향은 미치지 못하였다. 동일한 에러율과 동일한 복구시간을 가질 때 생산성에 큰 영향을 미치는 것은 부품장착 이였다. 이것은 한 사이클에서 가장 많은 작업을 하기 때문에 나타난 현상이다. 전체적으로 살펴보면 피더의 교체가 생산성에 가장 큰 영향을 미치고 PCB로드와 부품장착 그리고 부품 흡착 순으로 영향을 미친다.

시뮬레이터를 통하여 얻은 이러한 정보는 칩마운터의 조립프로그램작성에 라인의 균형과 단위 시간당 생산량을 높이기 위한 자료로 활용되어 더 적절한 조립프로그램을 작성하는데 바탕이 된다.

V. 결론

본 논문에서는 오프라인에서 칩마운터 조립프로그램의 검증으로 전자조립라인의 생산성 및 가동률 향상을 지원하는 칩마운터 시뮬레이터의 구현 방법을 제시하였다. 대상 시스템은 듀얼겐트리-멀티헤드형 칩마운터로 상호 겐트리간 충돌과 동시흡착문제를 고려하여 자원의 활용, 실패율, 생산률 등의 효과에 대에 양적, 질적으로 분석이 가능한 확률 페트리 네트를 이용하여 모델링 하였고, 객체지향 언어를 이용한 페트리 네트 클래스를 제작하여 시뮬레이터 구현에 사용함으로서 확장성과 이식성을 갖춘 모델링 및 프로그래밍 방법을 적용하였다.

칩마운터의 조립프로그램은 오프라인 프로그래밍의 사용증가와 유연 생산에 따른 작업교체주기 단축으로 작성의 빈도수가 일주일에 수십 개에 이를 정도로 증가하였다. 이러한 조립프로그램들은 다품종 소량생산 중심의 인쇄회로기판 제조라인에서 생산성과 생산 단가에 직결되는 자원과 시간 낭비의 방지를 위해 조립프로그램에 의한 칩마운터의 작업조건 설정 및 동작상태 확인과 생산성 추정 및 평가가 필수적이며, 이를 위한 시뮬레이터의 필요성이 증가 추세에 있다.

본 논문은 시뮬레이터 개발에 페트리 네트라는 체계적인 모델링 방법의 적용을 제시하였고, 다양한 형태의 칩마운터에 대응하기 위해 객체지향 언어를 이용한 프로그래밍으로 확장성 및 이식성을 고려한 페트리 네트 클래스를 구현하였다. 또한 칩마운터 시뮬레이터에 그래픽 통합환경으로 사용 편의성 증대를 위한 방법을 제시하고, 조립프로그램의 적절성 검증을 위한 분석적 방법도 제시하였다.

시뮬레이터는 독립적으로 운영되는 것 보다 오프라인 프로그래밍 도구와 연계되어 통합적으로 운영되어질 때 그 기능을 충분히 발휘한다고 볼 수 있다. 오프라인 프로그래밍 도구로 조립프로그램을 작성하고 동시에 시뮬레이션 하면, 조립프로그램의 작업조건설정, 피더배치 및 조립순서에 따른 동작과 생산성 등을 바로 파악할 수 있고, 조립프로그램 변경의 효과를 수시로 확인할 수 있으므로 최적화된 조립프로그램 작성에 이점이 있다. 이러한 통합프로그램의 모듈(module)로 시뮬레이터를 개발하는 것이 필요하다.

참고문헌

[1] T. L. Landers et. al., *Electronic Manufacturing Processes*, Prentice-Hall, 1994.

[2] K. Feldmann, J. Franke, and B. Zollner, " Optimization of SMT-systems by computer-aided planning, simulation and monitoring ", *Electronic Manufacturing Technology Symposium, IEMT Conf., 8th IEEE/CHMT International*, pp. 102~113, 1990.

[3] K. Feldmann, and K. Grampp, " Improvement of productivity by computer aided process planing, control and diagnosis " *Electronic Manufacturing*

Technology Symposium, 1992. IEMT 1992. 12th International pp. 360~367, 1992.
[4] S. Grotzinger and A. Sciomachen, "A petri net characterization of a high speed placement Machine", *Electronics Components Conference, Proceedings of the 38th*, pp. 64~68, 1988.
[5] M. C. Zhou and M. C. Leu, " PetriNet modeling of a flexible assembly station for printed circuit boards", *Proc. of the 1991 IEEE Int. Conf. on Robotics and Automation*, pp. 2530~2535, 4, 1991.
[6] M. C. Zhou and M. D. Jeng, " modeling, analysis, simulation, scheduling, and control of semiconductor manufacturing systems : A petri net approach", *IEEE Transactions on Semiconductor Manufacturing*, vol. 11, no. 3, 8, 1998.
[7] A. Sciomachen, S. Grotzinger, and F. Archetti, " Petri net-based emulation for a highly concurrent pick-and-place machine", *Robotics and Automation, IEEE Trans.* vol. 62, pp. 242~247, 4, 1990.
[8] T. M. Tripak, "Simulation software for surface mount assembly", *Winter Simulation Conf.*, 1993.
[9] M. A. Arslan and I. Fidan, "Modeling and performance analysis of an SMD assembly station using stochastic Petri nets and artificial neural networks ", *Systems, Man and Cybernetics, Intelligent Systems for the 21st Century., IEEE International Conf.* vol. 2, pp. 1768~1773, 1995.
[10] S. C. K. Shiu, E. C. C. Tsang, D. S. Yeung, and M. B. Lam, "Evaluation of printed circuit board assembly manufacturing systems using fuzzy colored petri nets", *Systems, Man, and Cybernetics, 1998 IEEE International Conf.* vol. 2, pp. 1506~1511, 1998.
[11] 박태형, "전자조립용 CAM 시스템의 개발 동향" 전자 공학회지 vol. 36, no. 3 pp. 272~280, 3, 1999.
[12] A. A. Desrochers and R. Y. Al-Jaar, "Application of Petri Nets in manufacturing systems," *IEEE Press*, USA, 1995.
[13] G. Lefranc, P. Vera, N. Gonzilez, and P. Valenzuela, "Software for stochastic petri nets oriented to flexible manufacturing systems ", *Proceedings of the IEEE International Symposium* vol. 1 , pp. 171~176, 1997.
[14] G. Ciardo, R. German, and C. Lindemann, "A characterization of the stochastic process underlying a stochastic petri net", *Petri Nets and Performance Models*, Proc. 5th International Workshop on , pp. 170~179, 1993.
[15] S. R. Ladd, C++ techniques and application, Prentice Hall, UK, 1990.
[16] S. Prata, C++ Primer Plus, The Waite Group, 1995
[17] 이상엽, Visual C++ Programming bible ver 6.x, 영진출판사, 서울, 2000.

박 기 범
1998년 충북대학교 제어계측공학과 학사. 2001년 동 대학원 석사. 2001년~현재, (주)삼성중공업 연구원. 관심분야는 공장 자동화/CIM/FMS., 모델링 및 시뮬레이션.

박 태 형
1988년 서울대학교 제어계측공학과 학사. 1990년 동 대학원 석사 및 1994년 동 대학원 박사, 1992년~1994년 제어계측신기술 연구센터 연구원, 1994년~1997년 삼성항공산업(주) 정밀기기연구소 선임연구원, 1997년~현재 충북대학교 전기전자공학부 조교수. 관심분야는 반도체 및 전자 조립 시스템, 최적화 알고리즘 응용.