DDR 알고리즘에 기반한 교착상태배제 래더다이어그램 설계

Synthesis of Deadlock-Free Ladder Diagrams for PLCs Based on Deadlock Detection and Recovery (DDR) Algorithm

차 종 호, 조 광 현
(Jong-Ho Cha and Kwang-Hyun Cho)

Abstract : In general, a deadlock in flexible manufacturing systems (FMSs) is caused by a resource limitation and the diversity of routings. However, the deadlock of industrial controllers such as programmable logic controllers (PLCs) can occur from different causes compared with those in general FMSs. The deadlock of PLCs is usually caused by an error signal between PLCs and manufacturing systems. In this paper, we propose a deadlock detection and recovery (DDR) algorithm to resolve the deadlock problem of PLCs at design stage. This paper employs the MAPN (modified automation Petri net), MTPL (modified token passing logic), and ECC (efficient code conversion) algorithm to model manufacturing systems and to convert a Petri net model into a desired LD (ladder diagram). Finally, an example of manufacturing systems is provided to illustrate the proposed DDR algorithm.

Keywords : FMS, PLC, Petri net, LD, MAPN, MTPL, ECC, deadlock, DDR algorithm

I. 서론

유연생산시스템(FMS flexible manufacturing systems, 이하 FMS)은 과거의 획일화된 단일생산시스템과는 달리 다양하고 유동적인 생산요구에 맞춰 시스템의 효율성과 유연성을 증가시키고 있다. 하지만 생산공정시 발생하는 교착상태(deadlock)는 시스템을 정지시켜 생산량 감소와 이로 인한 생산비용 증대를 가져온다. 따라서 교착상태문제를 해결하는 것이 시스템의 효율성과 유연성 증가를 위해서 반드시 필요하다. FMS에서 교착상태는 두개 이상의 부품이 서로 공유자원을 기다리면서 동일한 상태에 지속적으로 머무는 것을 말한다. 유연생산시스템의 교착상태를 다룬 논문은 연구방향에 따라 교착상태를 검출하고 올바른 상태로 되돌리는 데 초점을 둔 논문들[1], [2]과 교착상태를 방지하거나 회피하는데 초점을 둔 논문들[3]-[8]로 나눌 수 있다. 또한 시스템모델링방법에 따라 페트리네트(Petri net)[1][3][5][8], 유한 오토마타[6], 그래프 이론[2] 등을 이용한 것으로 나눌 수 있다. 하지만 최근까지 FMS의 교착상태 관련 연구들은 시스템 내의 자원 공유상태로 인해 발생하는 교착상태에만 관심을 두고 있어 PLC와 같은 실제 제어기상에서 발생가능한 교착상태를 다루고 있지는 않다.

PLC 상의 교착상태문제는 FMS 내의 공유자원으로 인해 발생하는 자원분배문제와는 다르게 주로 잘못된 신호의 전달로 발생되는 오동작에 의한 것이다. 왜냐하면 이 오동작이 시스템의 상태변화를 막아 현상태를 계속 유지하게 함으로써 시스템을 교착상태에 빠뜨리기 때문이다. PLC가 사용되는 곳의 작업환경은 큰 부하로 인해 많은 노이즈를 발생한다. 이 때문에 산업현장에서 주로 사용되는 주파수 발진 센서의 경우 노이즈에 약한 특성으로 인해 원하지 않는 결과를 가져와 때때로 시스템을 다운시킨다. 그림 1은 잘못된 입력신호 Ic에 의해 발생하는 타이밍도상의 출력접점의 오동작을 보여주고 있다. 출력접점 O1은 입력접점 P1과 Ic가 모두 on 상태일 때 on이 된다. 하지만 입력신호값의 측정 과정에서 잘못된 값 측정이 이루어지면 O1은 on이 되지 못하므로 교착상태에 빠지게 된다. 따라서 이와 같은 교착상태가 발생했을 때 시스템을 모두 다운시킨 후 문제를 해결한다는 것은 시스템의 생산성 측면에서 나쁜 결과를 초래한다. 따라서 열악한 작업환경 상의 노이즈로 인한 교착상태발생은 송 · 배전 상의 단선문제가 아니므로 PLC와 같은 제어기 스스로가 문제를 해결할 수 있도록 설계된다면 생산성 향상을 기대할 수 있을 것이다.

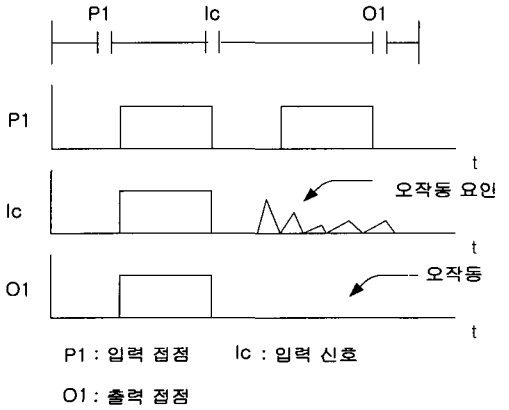


그림 1. 노이즈로 인한 출력접점의 오동작.
Fig. 1. An error of the output contact caused by noise.

논문접수 : 2001. 11. 2., 채택확정 : 2002. 5. 14.
차종호, 조광현 : 울산대학교 전기전자정보시스템공학부 (cjh@sys.ulsan.ac.kr/ckh@mail.ulsan.ac.kr)
※ 본 연구는 2001년도 한국학술진흥재단의 지원(KRF-2001-04-E00279)에 의하여 수행되었습니다.

체계적인 래더다이어그램(LD : ladder diagram, 이하 LD) 설계에 관한 논문으로는 Uzam[9]의 LD변환 연구와 [10]이 있다. [10]은 Uzam 논문에서 나타나는 문제점을 해결함으로써 좀 더 효율적이고 실제 프로그램과 유사한 형태의 결과물을 가지도록 한 것이다. 본 논문은 [10]의 결과에 기반하여 교착상태를 배제한 LD 설계에 관한 연구를 보여준다. 본 논문의 구성은 다음과 같다 : 2절에서는 플랜트를 모델링하는 기존의 방법들에 관해서 설명한다. 특히, 2.1절에서는 MAPN(modified automation Petri net)과 MTPL(modified token passing logic)을 소개하고, 2.2절에서는 ECC(efficient code conversion) 알고리즘을 소개한다[10]. 3절에서는 교착상태 문제를 해결하기 위한 DDR(deadlcok detection and recovery) 알고리즘을 소개하고 4절에서는 생산시스템에 DDR 알고리즘을 모의 적용시켜본 후 결과를 분석한다. 마지막으로 5절에서는 결론을 맺고 향후 연구과제를 논한다.

II. 플랜트 모델링

1. MAPN & MTPL

MAPN은 페트리네트를 사용하여 시스템을 모델링하는 방법으로서 플랜트를 생산물 이동라인과 액츄에이터 동작라인으로 나누어 설계한다. 생산물 이동라인은 부품과 같은 실제 자원이 이동하는 플랜트 상의 이동경로를 의미하고 액츄에이터 동작라인은 플랜트를 구성하는 액츄에이터들의 동작들을 의미한다. 이때 모델링시 사용되는 페트리네트는 일반적인 페트리네트와는 달리 실제 제어기의 액츄에이터 출력신호, 센서 입력신호를 다루기 위해 APN(automation Petri net)[9]를 사용한다. MTPL은 MAPN에 의해 완성된 시스템모델을 LD로 변환하는 방법이다. MTPL은 알고리즘 상의 주어진 타입에 따라 선택적으로 수용하여 페트리네트모델을 LD로 변환한다. MAPN과 MTPL은 시스템을 페트리네트로 모델링하고 다시 LD로 변환함으로써 PLC 제어프로그램의 체계적인 작성을 가능하게 한다.

2. ECC 알고리즘

ECC 알고리즘은 페트리네트를 이용하여 시스템을 모델링하고 LD로 변환하는 알고리즘이다. ECC 알고리즘은 시스템 모델이 오류상태에 빠지지 않도록 금지상태문제(forbidden state problem)를 함께 다루고 있다. ECC 알고리즘을 이용한 시스템 모델링 및 LD 프로그램으로의 변환은 다음과 같은 단계로 구성되어 있다:

단계 1 : 플랜트를 MAPN으로 모델링한다.

단계 2 : 도달가능한 상태표를 만든다.

단계 3 : 금지 접점(inhibitor contact), 동작 접점(action contact)을 찾아서 이에 따른 관리제어기를 구현한다.

단계 4 : 관리제어기와 초기 플랜트 모델을 조합하여 전체 시스템 모델을 만든다.

단계 5 : MTPL방법을 사용하여 설계된 시스템 모델을 LD로 변환한다.

III. DDR 알고리즘

FMS를 구성하는 플랜트와 PLC의 연결신호들은 그림 2와 같이 나타낼 수 있다. 그림 2에서 점선은 플랜트로부터 PLC

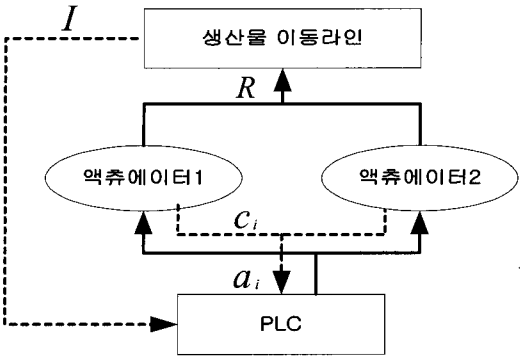


그림 2. FMS의 신호구성도.
Fig. 2. Signaling configuration diagram of FMS.

에 입력되는 신호를 의미하고 실선은 PLC로부터 플랜트에 입력되는 제어신호와 플랜트 내부 구성요소 간의 통신신호를 의미한다. 그림 2에서 선 위에 표시된 기호의 의미는 다음과 같다 :

a_i : PLC에서 액츄에이터로 보내는 i번째 동작신호($i \in N$, N: 자연수 집합),

c_i : 액츄에이터로부터 PLC로 보내는 액츄에이터 동작상태를 나타내는 i 번째 신호($i \in N$, N: 자연수 집합),

R : 액츄에이터들에 의해 발생된 상태변화신호,

I : 생산물 이동라인에 존재하는 토큰의 상태신호.

위 신호들을 4절에서 보일 그림 5의 예시플랜트에 기초하여 수학적으로 기술하면 다음과 같다. 이때 생산물 이동라인과 연결된 I는 동적상태함수(dynamic state function, Df)와 정적상태함수(static state function, Sf)를 이용하여 기술한다.

S_k : 플레이스 순열집합, k는 플레이스 순열집합의 시간에 따른 구별을 의미함,

$S_k \in B$, 이때 B는 이진집합,

$Pk_t \in \{0, 1\}$, k는 플레이스 번호, t는 시간변화를 의미함,

$S_{t-1} = P1_{t-1}P2_{t-1}P3_{t-1}$,

$S_t = P1_t P2_t P3_t$,

$S_{t+1} = P1_{t+1}P2_{t+1}P3_{t+1}$.

S_{t-1}(이전상태), S_t(현상태), S_{t+1}(다음상태)에 의해 Df는 다음과 같다.

$$Df(t) = \begin{bmatrix} S_{t-1} \\ S_t \\ S_{t+1} \end{bmatrix}, \text{ 또는}$$

$$Df(t) = \begin{bmatrix} Ps_{t-1} & P1_{t-1} & P2_{t-1} & P3_{t-1} \\ Ps_t & P1_t & P2_t & P3_t \\ Ps_{t+1} & P1_{t+1} & P2_{t+1} & P3_{t+1} \end{bmatrix}$$

t_k : T_k의 k 번째 부분트랜지션(partial transition),

i_{ck} : k 번째 금지 신호값, 또는 금지제어명령(inhibitor control command),

$\overline{i_{ck}}$: 역 신호 값으로 i_{ck}가 1이면 $\overline{i_{ck}}$는 0의 값을 가진다

Ps_t : 소스플레이스, t는 시간변화를 의미함

t_k, $i_k \in B$, $k \in N$, (N: 자연수 집합)

B는 이진집합이고, k는 부분 트랜지션의 구별을 의미하며, t_k, $i_{ck} \in \{0, 1\}$.

4절의 예시플랜트로부터 생산물 이동라인 내의 각 트랜지션을 다음과 같은 과정으로 i_{ck}, t_k의 곱으로 표현할 수 있다. 4절의 그림 5로부터 T_1은 t_1과 i_{c1}이 모두 on 상태일 때 활성화되며 이를 곱으로 표현한다. i_{c1}은 P1에 토큰이 존재하지 않으면 1이 된다.

$$T_1 = t_1 \times i_{c1} \tag{1}$$

T_2는 i_{c2}가 on 상태이고 T_6가 활성화될 때 활성화된다. T_6는 단순히 시간지연만 함으로 T_5가 활성화되어야 활성화시킬 수 있다. T_5는 $\overline{i_{c1}}$, i_{c2}, i_{c7}, i_{c5}가 모두 on 상태일 때 활성화된다.

$$\begin{aligned} T_5 &= \overline{i_{c1}} \times i_{c2} \times i_{c7} \times i_{c5}, \\ T_2 &= i_{c2} \times (\overline{i_{c1}} \times i_{c2} \times i_{c7} \times i_{c5}) \end{aligned} \tag{2}$$

T_3는 i_{c3}가 on 상태이고 T_8가 활성화될 때 활성화된다. T_8는 단순히 시간지연만 함으로 T_7가 활성화되어야 활성화시킬 수 있다. T_7는 $\overline{i_{c2}}$, i_{c3}, i_{c7}, i_{c5}가 모두 on 상태일 때 활성화된다.

$$\begin{aligned} T_7 &= \overline{i_{c2}} \times i_{c3} \times i_{c7} \times i_{c5}, \\ T_3 &= i_{c3} \times (\overline{i_{c2}} \times i_{c3} \times i_{c7} \times i_{c5}) \end{aligned} \tag{3}$$

T_4는 t_4가 on 상태일 때 활성화된다.

$$T_4 = t_4 \tag{4}$$

위의 식 (1), (2), (3), (4)로부터 T_1, T_2, T_3, T_4는 다음과 같다.

$$\begin{aligned} T_1 &= t_1 \times i_{c1}, \\ T_2 &= i_{c2} \times (\overline{i_{c1}} \times i_{c2} \times i_{c7} \times i_{c5}), \\ T_3 &= i_{c3} \times (\overline{i_{c2}} \times i_{c3} \times i_{c7} \times i_{c5}), \\ T_4 &= t_4. \end{aligned}$$

T_1, T_2, T_3, T_4에 의해 Sf는 다음과 같다:

$$Sf = [\ T_1 \quad T_2 \quad T_3 \quad T_4]^T$$

따라서 I는 다음과 같이 표현된다 :

$$I(t) = Df(t) \times Sf$$

$$= \begin{bmatrix} Ps_{t-1}T_1 & P1_{t-1}T_2 & P2_{t-1}T_3 & P3_{t-1}T_4 \\ Ps_tT_1 & P1_tT_2 & P2_tT_3 & P3_tT_4 \\ Ps_{t+1}T_1 & P1_{t+1}T_2 & P2_{t+1}T_3 & P3_{t+1}T_4 \end{bmatrix},$$

$$I(t) = [\ I_1 \quad I_2 \quad I_3]^T.$$

액츄에이터 동작라인과 연결된 a_i와 c_i, R은 다음과 같이 수학적으로 표현된다. 이는 4절의 예시플랜트로부터 액츄에이터 동작라인 내의 각 트랜지션을 i_{ck}와 t_k의 곱으로 표시한 것이다 :

$$\begin{aligned} T_5 &= t_5 \times \overline{i_{c1}} \times i_{c2} \times i_{c7} \times i_{c5}, \\ T_6 &= t_6, \\ T_7 &= t_7 \times \overline{i_{c2}} \times i_{c3} \times i_{c5} \times i_{c7}, \\ T_8 &= t_8. \end{aligned}$$

그리고 a_i와 c_i는 다음과 같이 표현된다:

$$a_i = [\ T_5 \quad T_6 \quad T_7 \quad T_8]^T,$$

$$c_i(t) = \begin{bmatrix} P4_{t-1} & P5_{t-1} & P6_{t-1} & P7_{t-1} \\ P4_t & P5_t & P6_t & P7_t \\ P4_{t+1} & P5_{t+1} & P6_{t+1} & P7_{t+1} \end{bmatrix}.$$

따라서 R은 다음과 같이 표현된다:

$$R(t) = c_i \times a_i$$

$$= \begin{bmatrix} P4_{t-1}T_5 & P5_{t-1}T_6 & P6_{t-1}T_7 & P7_{t-1}T_8 \\ P4_tT_5 & P5_tT_6 & P6_tT_7 & P7_tT_8 \\ P4_{t+1}T_5 & P5_{t+1}T_6 & P6_{t+1}T_7 & P7_{t+1}T_8 \end{bmatrix},$$

$$R(t) = [\ R_1 \quad R_2 \quad R_3]^T.$$

DDR 알고리즘은 그림 3과 같이 시스템에서 토큰의 이동이 없는 교착상태를 배제하기 위한 알고리즘이다. 그림 3에서 Ps는 소스플레이스, P1, P2는 공정을 나타내는 플레이스로서 Ps의 토큰을 먼저 차지하기 위해 P1과 P2는 경쟁을 통해 T1과 T2를 동시에 활성화시키면 교착상태를 발생시킨다. 이때 DDR 알고리즘이 교착상태를 배제하는 방법은 전체시스템의 논리적 동작을 변경하지 않으면서 교착상태를 가지는 플레이스의 출력 측에 연결된 트랜지션을 강제로 활성화시키는 것이다. 하지만 강제로 트랜지션을 활성화시키는 것

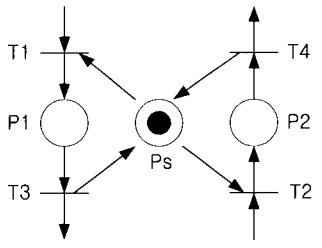


그림 3. 교착상태.

Fig. 3. Deadlock model.

은 시스템 내의 토큰흐름에 방해가 될 수 있으므로 정확하게 교착상태인 지점을 우선 검출하는 것이 중요하다. 따라서 DDR 알고리즘은 교착상태 위치검출과 교착상태배제, 두 부

분으로 나누어져 있다. 이 두 부분을 포함한 DDR 알고리즘은 다음 단계들로 이루어져 있다:

단계 1 : 주어진 시스템 모델로부터 a_i, c_i, R, I 함수 를 구한다.

단계 2 : 교착상태지점을 생산물 이동라인과 액츄에이터 동작라인으로 나누어 검출하고 교착상태 유형을 정한다:

1)생산물 이동라인에서의 교착상태 지점검출

① I의 I_{t-1}, I_t, I_{t+1}를 각각 비교한 후, 교착 상태지점을 검출한다, 이때 $t-1$, t, $t+1$은 시간변화를 의미한다.

② I에 에러가 발생하면 Sf와 Df를 조사한다.

③ Sf에 에러가 있으면 정적인 교착상태로 Df에 에러가 있으면 동적인 교착상태로 각각 정의한다.

2) 액츄에이터 동작라인에서의 교착상태 지점검출

① R의 R_{t-1}, R_t, R_{t+1}를 각각 비교한 후 교착상태지점을 검출한다, 이때 $t-1$, t, $t+1$은 시간변화를 의미한다.

② R에 에러가 발생하면 c_i와 a_i를 조사한다,

③ c_i에 에러가 있으면 정적인 교착상태로 a_i 에 에러가 있으면 동적인 교착상태로 각각 정의한다.

단계 3 : 단계 2의 정보를 통해 교착상태를 유발시킨 비활성화된 트랜지션을 강제로 활성화시키는 제어 식을 만든다.

단계 4 : 단계 3의 제어식을 기초로 하여 LD상에서 교착상태 발생시 문제를 해결할 수 있도록 수정한 다.

위의 단계 1과 2는 교착상태 위치검출, 단계 3은 교착상태 배제방법, 단계 4는 교착상태 발생시 대처할 수 있는 LD 생성방법을 각각 설명한 것이다.

IV. 시스템 모의 적용

본 절에서는 앞에서 제시한 DDR 알고리즘을 생산시스템에 모의 적용하여 그 효용성을 검증해 본다.

그림 4의 생산시스템의 예는 페그(peg)와 링(ring)이 혼합되어 입력라인으로 유입되면 분류영역에서 페그와 링을 구분하여 이동시키고 결합영역에서 각각의 페그와 링을 결합시켜 원하는 물품을 조립하는 시스템이다. 이때 분류영역과 결합영역에는 각각 분류 혹은 결합동작을 수행하는 솔레노이드(solenoid)가 존재한다.

그림 5는 MAPN과 MTPL, ECC 알고리즘에 의해 완성된 생산시스템의 페트리네트모델이다[10]. 그림 5의 생산시스

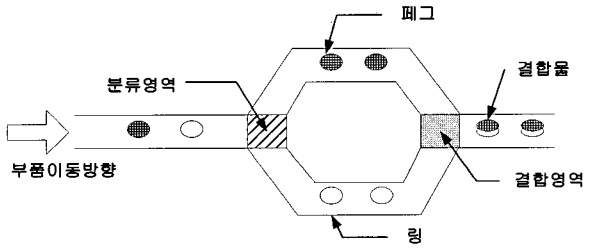


그림 4. 생산시스템 예의 구성도.
Fig. 4. Configuration of an example manufacturing system.

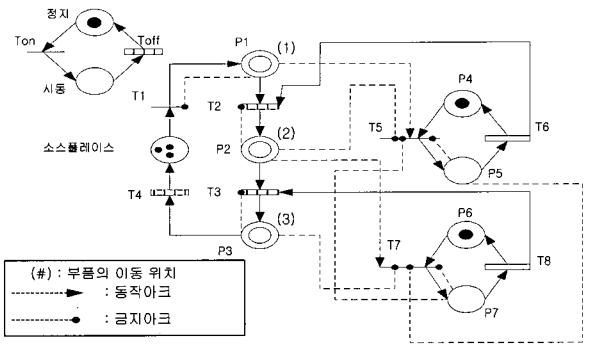


그림 5. 생산시스템의 페트리네트 모델.
Fig. 5. Petri net model of the manufacturing system.

템 모델은 오신호로 인해 교착상태에 빠질 경우 시스템을 다운시킨 후 문제점을 해결해야 한다. 시스템을 다운시키지 않고 그림 6에서처럼 PLC와 같은 제어기 스스로가 오신호로 인한 교착상태를 감지하고 해결할 수 있다면 시스템의 생산성을 높일 수 있을 것이다. 위의 그림 5로부터 DDR 알고리즘을 이용하여 교착상태문제를 해결한 시스템모델을 생성하면 그림 6과 같다.

그림 6에서 D_{s1}, D_{s2}, D_{s3}, D_{s4}, D_{s5}는 정적인 교착상태를 제어하는 플레이스, D_{d1}, D_{d2}, D_{d3}, D_{d4}, D_{d5}는 동적인 교착상태를 제어하는 플레이스이다. 이때 D_{s1}, D_{s2}, D_{s3}와 D_{d1}, D_{d2}, D_{d3}는 생산물 이동라인에 존재하는 교착상태를, D_{s4}, D_{s5}와 D_{d4}, D_{d5}는 액츄에이터 동작라인에 존재하는 교착상태를 나타내는 플레이스이다. 각 교착상태를 제어하는 플레이스의 제어식은 다음과 같은 과정으로 만들어진다.

그림 6으로부터 (1)지역의 정적인 교착상태를 해결할 수 있는 방법은 강제적으로 T_2를 활성화시켜주면 된다.

$$D_{s1} = T_2 = t_2 \times i_{c2} \times (t_5 \times \overline{i_{c1}} \times i_{c2} \times i_{c4} \times i_{c5})$$

(2)지역의 정적인 교착상태를 해결할 수 있는 방법은 강제적으로 T_3를 활성화시켜주면 된다.

$$D_{s2} = T_3 = t_3 \times i_{c3} \times (t_7 \times \overline{i_{c2}} \times i_{c3} \times i_{c4} \times i_{c5})$$

(3)지역의 정적인 교착상태를 해결할 수 있는 방법은 강

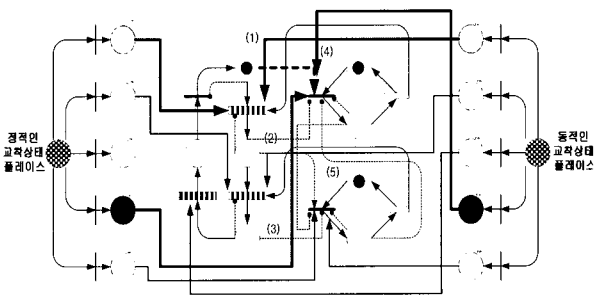


그림 6. DDR 알고리즘을 포함한 페트리네트 모델.
Fig. 6. Petri net model via DDR algorithm.

제적으로 T_4를 활성화시켜주면 된다.

$$D_{s3}=T_4=t_4$$

액츄에이터 1의 정적인 교착상태를 해결할 수 있는 방법은 강제적으로 T_5를 활성화시켜주면 된다.

$$D_{s4}=T_5=t_5\times\overline{i_{c1}}\times i_{c2}\times i_{c4}\times i_{c5}$$

액츄에이터 2의 정적인 교착상태를 해결할 수 있는 방법은 강제적으로 T_7를 활성화시켜주면 된다.

$$D_{s5}=T_7=t_7\times\overline{i_{c2}}\times i_{c3}\times i_{c4}\times i_{c5}$$

(1)지역의 동적인 교착상태를 해결할 수 있는 방법은 D_{s1}과 함께 시스템 내의 플레이스 상태 변화도 강제로 제어하도록 한다.

$$D_{d1}=(P_1\times\overline{T_1})\times(\overline{P_2}\times T_2)$$

(2)지역의 동적인 교착상태를 해결할 수 있는 방법은 D_{s2}와 함께 시스템 내의 플레이스 상태 변화도 강제로 제어하도록 한다.

$$D_{d2}=(P_2\times\overline{T_2})\times(\overline{P_3}\times T_3)$$

(3)지역의 동적인 교착상태를 해결할 수 있는 방법은 D_{s3}과 함께 시스템 내의 플레이스 상태 변화도 강제로 제어하도록 한다.

$$D_{d3}=(P_3\times\overline{T_3})\times T4$$

액츄에이터 1의 동적인 교착상태를 해결할 수 있는 방법은 D_{s4}와 함께 시스템 내의 플레이스 상태 변화도 강제로 제어하도록 한다.

$$D_{d4}=(P_4\times\overline{T_6})\times(\overline{P_5}\times T_5)$$

액츄에이터 2의 동적인 교착상태를 해결할 수 있는 방법은 D_{s5}와 함께 시스템 내의 플레이스 상태 변화도 강제로 제어하도록 한다.

$$D_{d5}=(P_6\times\overline{T_8})\times(\overline{P_7}\times T_7)$$

위 교착상태 제어식들은 교착상태를 가지는 플레이스의 출력측 트랜지션을 강제로 활성화시켜 교착상태를 해결하도록 한다. DDR 알고리즘에 의해 생성된 제어식들은 그림 6과 같이 교착상태 발생시 상황에 따라 적절히 대응한다. 그러나 그림 6으로부터 PLC의 LD프로그램을 생성할 경우 두 가지 문제점을 가진다. 첫째, PLC의 LD프로그램에서 교착상태를 모니터링할 수 있는 방법이 없다. 둘째, DDR 알고리즘의 제어식으로부터 직접 실제 PLC의 동작을 제어하기에는 다소 문제가 있다. 즉 PLC의 접점은 on, off로 동작을 기술하는데 반해 제어식은 수식의 결과 값에 따라 동작을 기술하기 때문이다. 따라서 완성된 페트리네트모델로부터 교착상태문제를 해결한 LD프로그램을 생성하기 위해서는 위에서 제시한 문제점들을 해결할 필요가 있다.

LD프로그램 상의 교착상태를 모니터링하기 위해서 그림 7과 같이 모니터링 부분을 설계한다. 시스템의 상태는 다음 6개의 X1, X2, X3, X4, X5, X6 접점이 on, off 되는 것으로 표현한다. X1은 시스템의 초기상태로 P1, P2, P3, P5, P7에 모두 토큰이 없음을 나타낸 것이다. 교착상태는 X1과 같은 접점이 일정시간이 지난 뒤에도 계속 on 상태에 남아있다면 교착상태에 빠진 것으로 판단할 수 있으며, 이때 D1 접점이 on 된다. 따라서 그림 7의 왼쪽은 시스템 상에 존재하는 상태를 접점으로 표시한 것이고 오른쪽은 이 접점을 기반으로 교착상태지점을 검출하는 접점을 표시한 것이다.

그림 8은 그림 7의 시스템 모니터링 부분을 바탕으로 DDR 알고리즘의 제어식을 LD프로그램에 적용시키는 방법을 보여준다. 이를 위해 우선 LD 프로그램에 적용시키는 교착상태 제어식을 다음과 같이 수정한다 :

$$D_{ds}=Ps\times(\overline{P1}\times T2)$$
$$D_{d1}=P1\times(\overline{P2}\times T2)$$
$$D_{d2}=P2\times(\overline{P3}\times T3)$$
$$D_{d3}=P3\times T4$$
$$D_{d4}=P4\times(\overline{P5}\times\overline{T5})$$
$$D_{d5}=P6\times(\overline{P7}\times T7)$$

수정한 제어식은 그림 7에 따라 교착상태를 모니터링할

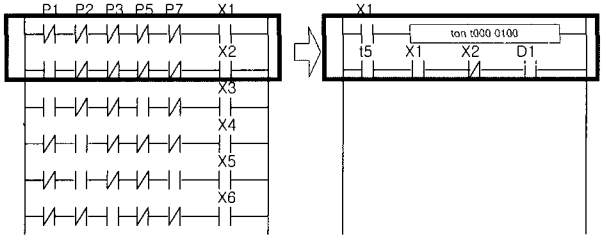


그림 7. LD의 시스템 모니터링 부분 설계.
Fig. 7. Design of system monitoring parts in LD.

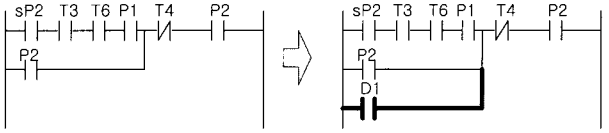


그림 8. LD상의 DDR 알고리즘 적용방법.
Fig. 8. Application of DDR algorithm to LD.

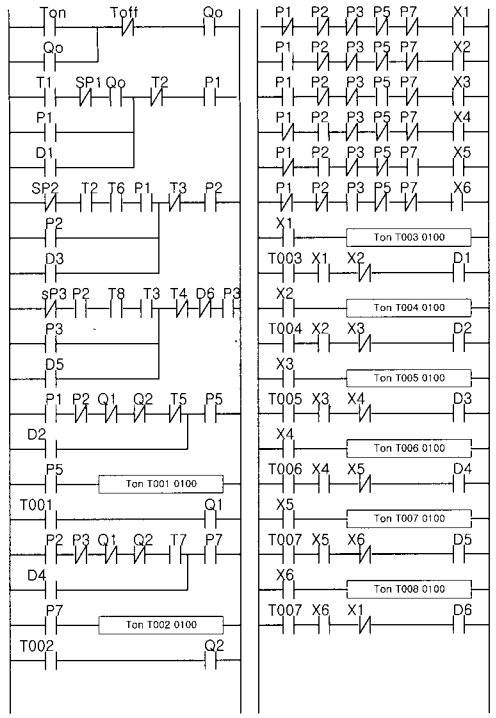


그림 9. 교착상태를 배제한 LD.
Fig. 9. Deadlock-free LD.

때 시스템의 시동신호를 함께 고려해주기 위해서 소스플레이스의 교착상태 제어식을 추가하고 프로그램 상에서 교착상태의 혼동을 막기 위해 동적인 교착상태만 고려한다. 그림 9는 MTPL과 ECC 알고리즘에 의해 LD를 생성하고 그림 7과 8, 위의 수정된 교착상태 제어식에 따라 교착상태를 배제한 최종적인 LD프로그램을 나타낸 것이다. 그림 9는 왼쪽 LD의 시스템 모니터링 부분과 교착상태검출부분을 통해서 오른쪽 LD가 교착상태 발생시 위의 교착상태 제어식들을 통해서 시스템 동작을 원활하게 하도록 한다.

V. 결론

PLC는 오신호로 인해 교착상태에 빠질 수 있다. PLC는 시스템을 제어하는 제어기이므로 PLC의 교착상태 발생은 시스템의 교착상태로 이어진다. 따라서 시스템을 페트리네트로 모델링한 후 LD변환 시에는 교착상태 발생에 대처할 수 있도록 시스템을 구현할 필요가 있다. 본 연구는 이점에 착안하여 교착상태에 대처할 수 있는 시스템 모델링방법과 LD 구현방법을 제시하였다. 시스템의 생산성을 증대시키기 위해서는 고장시간과 지연시간을 줄이는 것이 필요하다. 따라서 시스템의 소스과잉과 교착상태의 방지 및 해결이 무엇보다 중요하다. DDR 알고리즘에 의해 생성된 페트리네트모델과 LD 변환결과는 교착상태를 배제함으로써 시스템의 지연시간과 고장시간을 줄여 생산성을 향상시킬 수 있다. 향후 연구과제로는 보다 많은 장치들을 포함한 실제 플랜트 상의 교착상태 문제해결에 관한 적용연구, DDR 알고리즘의 현장 적용 시 강인성에 관한 입증 등이 있다.

참 고 문 헌

[1] R. A. Wysk, N. S. Yang, and S. Joshi, "Detection of deadlocks in flexible manufacturing cells," *IEEE Trans. on Robotics and Automation,* vol. 7, no. 6, pp. 853-859, 1991.

[2] H. Cho, T. K. Kumaran, and R. A. Wysk, "Graph-theoretic deadlock detection and resolution for flexible manufacturing systems," *IEEE Trans. on Robotics and Auto -mation,* vol. 11, no. 3, pp. 413-421, 1995.

[3] J. Ezpeleta and J. M. Colom, "Automatic synthesis of colored Petri nets for the control of FMS," *IEEE Trans. on Robotics and Automation,* vol. 13, no. 3, pp. 327-337, 1997.

[4] J. W. Choi and J. W. Choi, "An efficient algorithm for control action sequences in FMS using dynamic programming," *Proc. of the 39th SICE Annual Conference,* Lizuka, Japan, pp. 221-224, Oct 2000.

[5] R. P. Judd, Z. Wenle, P. Deering, and R. Lipset, "A scalable deadlock avoidance algorithm for flexible manufacturing systems with free choice in part routing," *Proc. of American Control Conference,* Chicago, U.S.A., vol. 5, pp. 3545-3549, June 2000.

[6] A. Yalcin and T. O. Boucher, "Deadlock avoidance in flexible manufacturing systems using finite automata," *IEEE Trans. on Robotics and Automation,* vol. 16, no. 4, pp. 424-429, 2000.

[7] 최재원, 이만형, 김성권, "유연 생산 시스템의 생산성 향상을 위한 공유 자원 수의 증가 및 교착 상태회피기법 비교," 한국자동제어학술회의논문집, pp. 221-224, vol. 2, 1999.

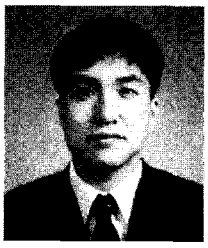
[8] M. V. Iordache, J. O. Moody, and P. J. Antsaklis, "A method for the synthesis of deadlock prevention controllers in systems modeled by Petri nets," *Proc. of the American Control Conference,* Chicago, U.S.A., vol. 5, pp. 3167-3171, June 2000.

[9] M. Uzam, A. H. Jones, and I. Yücel, "Using a Petri-net-based approach for the real-time supervisory control of experimental manufacturing system," *International Journal of Advanced manufacturing Technology,* vol. 16, pp. 498-515, July 2000.

[10] 차종호, 조광현, "효율적 코드변환 알고리즘에 기 반한 PLC의 체계적 설계," 제어·자동화·시스템공학 논문지, 제 7권, 제 12호, pp. 1009-1014, Dec., 2001.

차 종 호

2001년 울산대학교 전자공학과 졸업. 2001년~현재 울산대학교 전기전자정보시스템공학부 석사과정. 관심분야는 이산사건시스템의 관리제어, 유연생산시스템의 설계 및 분석, 생산시스템의 자동화, 바이오정보공학 응용등.



조 광 현

1993년 한국과학기술원 전기및전자공학과 졸업. 동대학원 석사(1995), 동대학원 박사(1998). 1998~1999. 동대학원 위촉연구원 및 연수연구원, 1999년 3월~현재 울산대학교 전기전자정보시스템공학부 조교수. 관심분야는 이산사건시스템의 해석 및 관리제어, 생산시스템 자동화(반도체 생산시스템 자동화), 통신망 분석 및 제어, 광통신망 시스템(스위칭 및 라우팅), 바이오정보공학 응용 등.