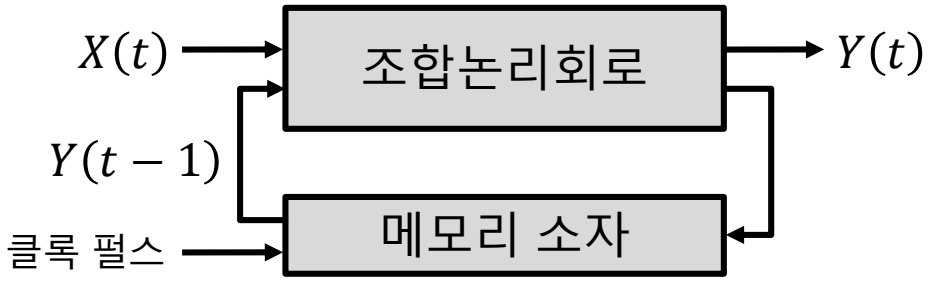
Lecture 08

동기 순서논리회로

개요

- 조합논리회로(**combinational** logic circuit)
  - 임의의 시점에서 이전 입력값에 관계없이 현재 입력값에 따라 출력이 결정되는 논리회로임
- 순서논리회로(**sequential** logic circuit)
  - 현재 입력값과 이전 출력 상태에 따라 출력값이 결정되는 논리회로임
  - 신호의 타이밍(**timing**)에 따라 다음과 같이 나눌 수 있음
    - 동기(synchronous) : 클록 펄스 입력을 통해서 동작함
    - 비동기(asynchronous) : 입력이 변하는 순간에 따라 동작함



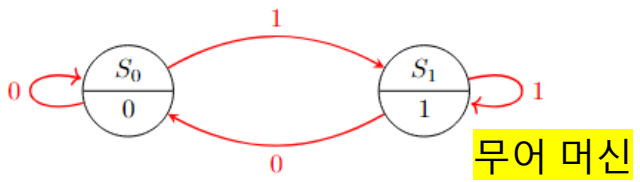
$X(t)$ 입력

$Y(t)$ 출력 (현재 상태)

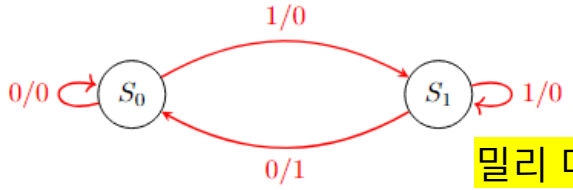
$Y(t-1)$ 출력 (이전 상태)

동기 순서회로 해석

- 순서논리회로 해석 과정
  ① 회로 입력과 출력에 대한 변수 명칭 부여
  ② 조합논리회로가 있으면 조합논리회로의 불 대수식 유도
  ③ 회로의 상태표 작성
  ④ 상태표를 이용해 상태도 작성
  ⑤ 상태 방정식 유도
  ⑥ 상태표와 상태도를 분석하여 회로의 동작 설명
- 순서논리회로 형태
  - 무어 머신(**Moore machine**) : 출력이 플립플롭의 현재 상태에만 의존함
  - 밀리 머신(**Mealy machine**) : 출력이 플립플롭의 현재 상태와 입력들에 모두 의존함



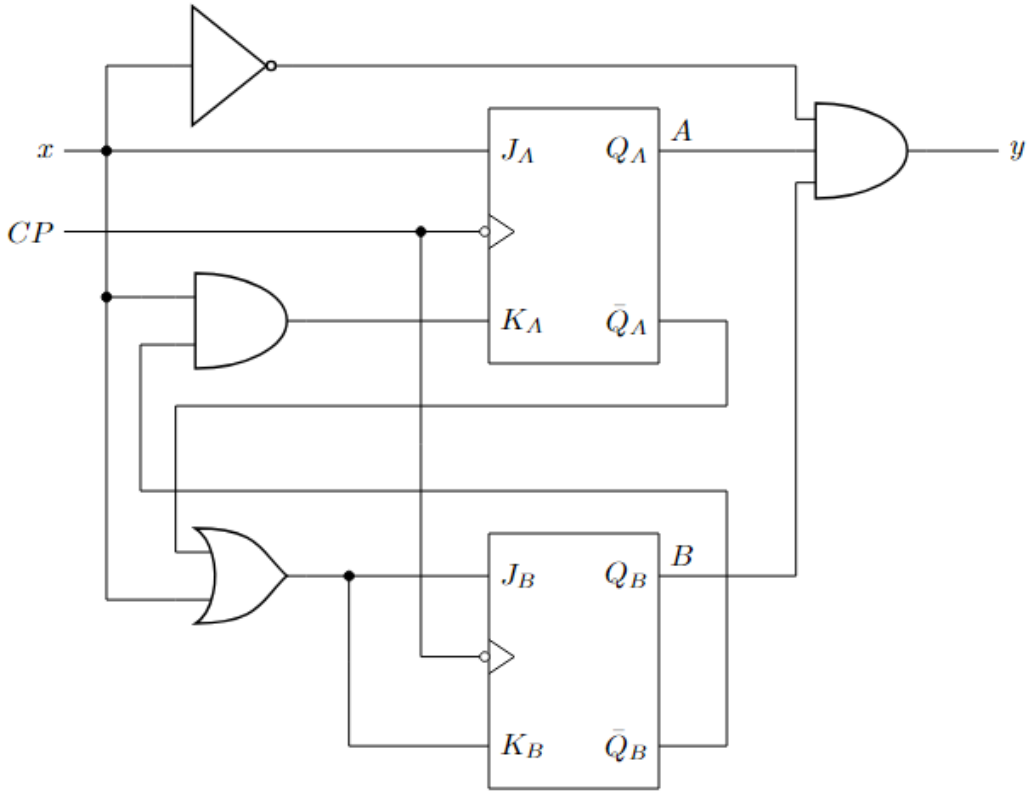
무어 머신



밀리 머신

동기 순서회로 해석

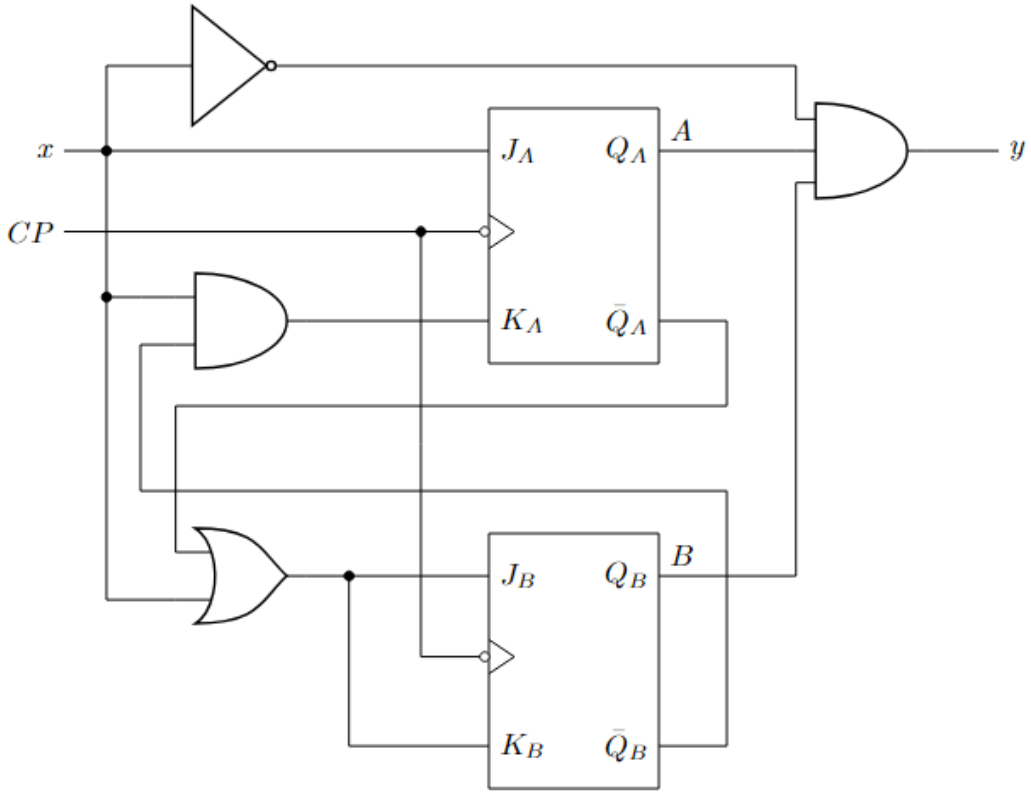
- 예, JK 플립플롭을 사용한 밀리 머신
  - ① 변수 명칭 부여
    - 입력 : x
    - 출력 : y
    - 플립플롭 출력 : A, B
  - ② 불 대수식 유도
    - 플립플롭 A의 입력
      $J_A = x, K_A = x\bar{B}$
    - 플립플롭 B의 입력
      $J_B = K_B = x + \bar{A}$
    - 시스템 출력
      $y = AB\bar{x}$



동기 순서회로 해석

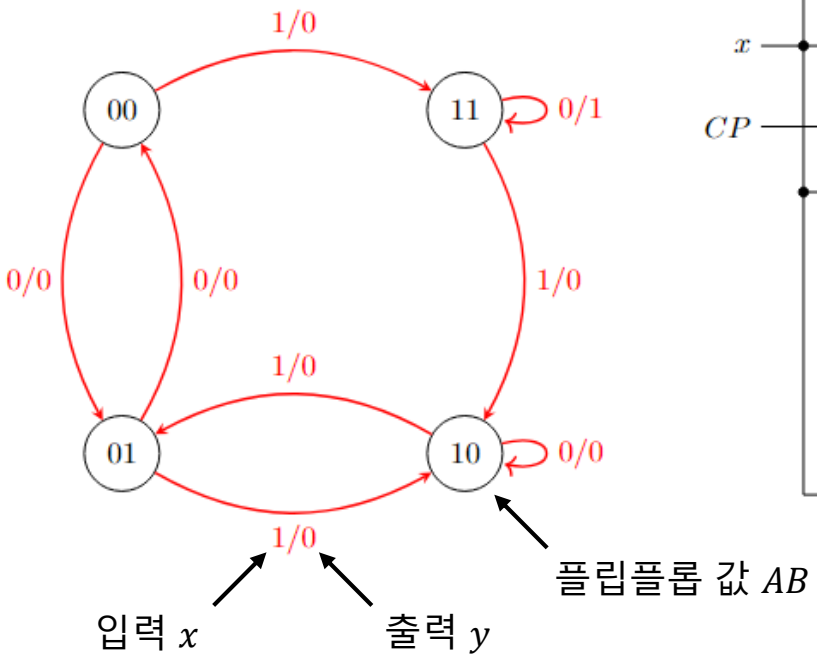
- 예, JK 플립플롭을 사용한 밀리 머신
  ③ 상태표 작성

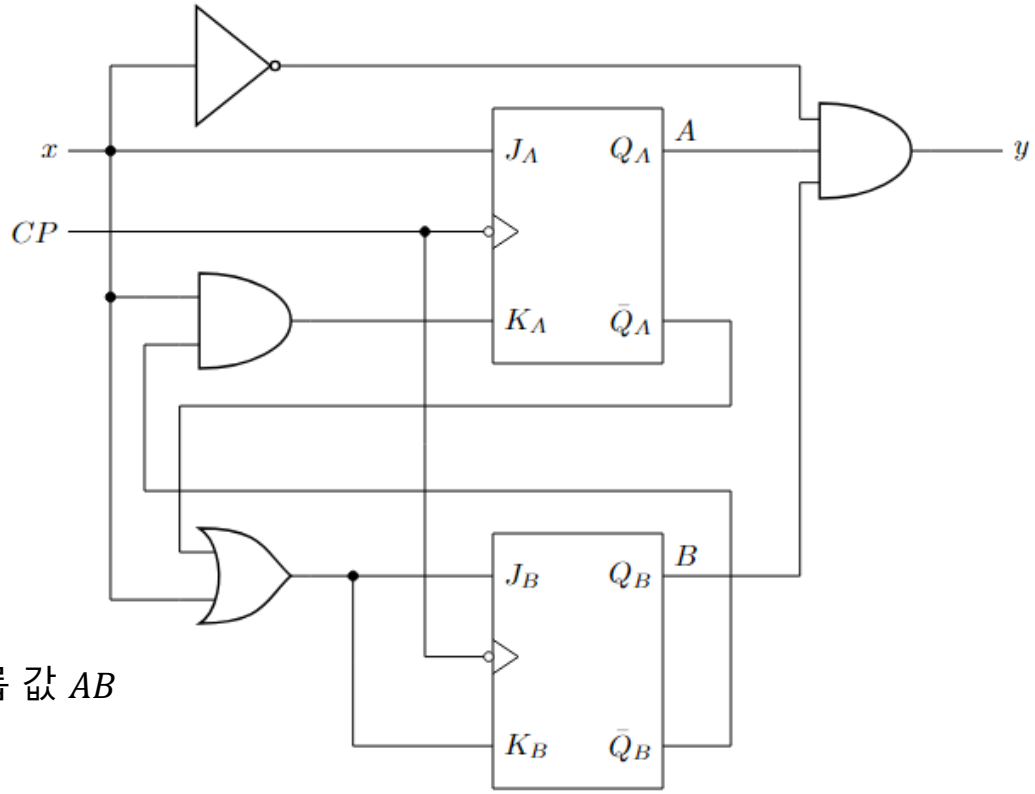
현재 상태		입력	다음 상태		출력
A	***B***	***x***	***A***	***B***	***y***
0	0	0	0	1	0
0	0	1	1	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	0	1	0
1	1	0	1	1	1
1	1	1	1	0	0



동기 순서회로 해석

- 예, JK 플립플롭을 사용한 밀리 머신
  ④ 상태도 작성





동기 순서회로 해석

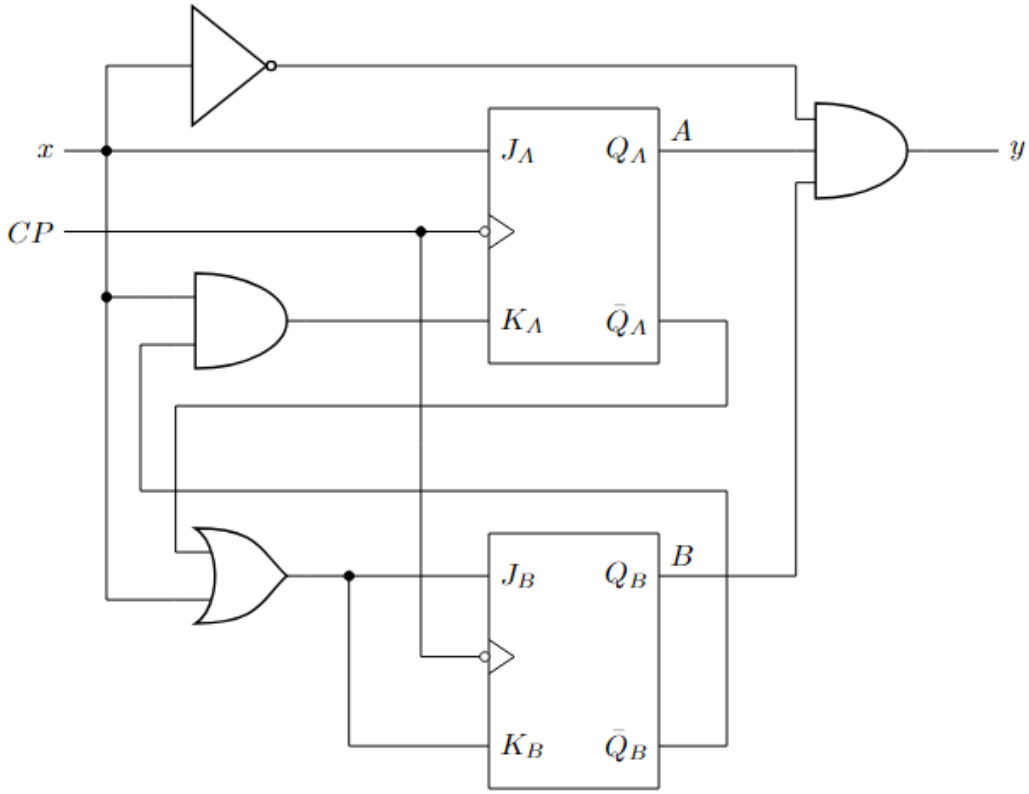
- 예, JK 플립플롭을 사용한 밀리 머신
  - ⑤ 상태 방정식 유도

A \ Bx	00	01	11	10
0		1	1	
1	1		1	1

$$A(t+1) = \bar{A}x + AB + A\bar{x}$$

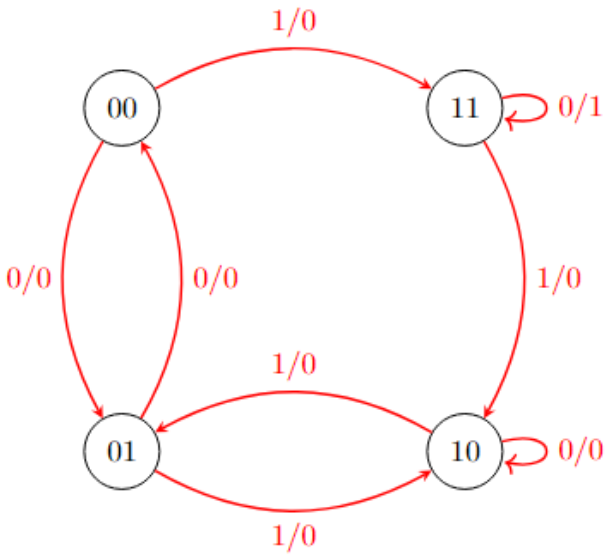
A \ Bx	00	01	11	10
0	1	1		
1		1		1

$$B(t+1) = \bar{A}\bar{B} + \bar{B}x + AB\bar{x}$$



동기 순서회로 해석

- 예, JK 플립플롭을 사용한 밀리 머신
  - ⑥ 회로의 동작 설명
    - 두 플립플롭의 현재 상태가 00이면,
      입력 $x = 0$일 때 클록 펄스가 입력되면
      01로 전이하며, 출력이 $y = 0$이 됨.
      입력 $x = 1$일 때 클록 펄스가 입력되면
      11로 전이하며, 출력이 $y = 0$이 됨.
    - 두 플립플롭의 현재 상태가 11이면,
      입력 $x = 0$일 때 클록 펄스가 입력되면
      11을 유지하며, 출력이 $y = 1$이 됨.
      입력 $x = 1$일 때 클록 펄스가 입력되면
      10로 전이하며, 출력이 $y = 0$이 됨.
    - …



플립플롭의 여기표

- 특성표
  - 현재 상태와 입력값이 주어졌을 때, 다음 상태가 어떻게 변하는가를 나타내는 표임
- 여기표(excitation table)
  - 현재 상태에서 다음 상태로 변했을 때 플립플롭의 입력 조건 상태를 나타내는 표임
  - 순서논리회로를 설계할 때 많이 사용함

플립플롭의 여기표

- SR 플립플롭

특성표

입력		현재 상태	다음 상태
S	R	$Q(t)$	$Q(t+1)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	?
1	1	1	?

여기표

현재 상태	다음 상태	요구 입력	
$Q(t)$	$Q(t+1)$	S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

SR 플립플롭의 진리표

S	R	$Q(t+1)$
0	0	부변
0	1	0
1	0	1
1	1	부정

플립플롭의 여기표

- JK 플립플롭

특성표

입력		현재 상태	다음 상태
J	K	$Q(t)$	$Q(t+1)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

여기표

현재 상태	다음 상태	요구 입력	
$Q(t)$	$Q(t+1)$	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

JK 플립플롭의 진리표

J	K	$Q(t+1)$
0	0	부변
0	1	0
1	0	1
1	1	토글

플립플롭의 여기표

- D 플립플롭

특성표

입력	현재 상태	다음 상태
D	$Q(t)$	$Q(t+1)$
0	0	0
0	1	0
1	0	1
1	1	1

여기표

현재 상태	다음 상태	요구 입력
$Q(t)$	$Q(t+1)$	D
0	0	0
0	1	1
1	0	0
1	1	1

D 플립플롭의 진리표

D	$Q(t+1)$
0	0
1	1

플립플롭의 여기표

- T 플립플롭

특성표

입력	현재 상태	다음 상태
T	$Q(t)$	$Q(t+1)$
0	0	0
0	1	1
1	0	1
1	1	0

여기표

현재 상태	다음 상태	요구 입력
$Q(t)$	$Q(t+1)$	T
0	0	0
0	1	1
1	0	1
1	1	0

T 플립플롭의 진리표

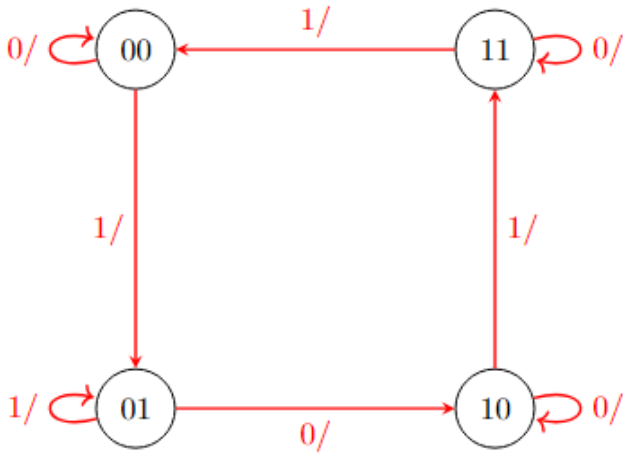
T	$Q(t+1)$
0	부변
1	토글

동기 순서회로 설계

- 동기 순서논리회로 = 조합논리회로 + 기억 소자
  - 조합논리회로 : AND나 OR 같은 기본 논리 게이트들의 결합으로 구성됨
  - 기억 소자 : 플립플롭 1개 이상이 병렬 또는 직렬로 결합되어 구성됨
- 동기 순서논리회로 설계 과정
  ① 회로 동작 기술(상태도 작성)
  ② 정의된 회로의 상태표 작성
  ③ 필요한 경우 상태 축소 및 사태 할당
  ④ 플립플롭의 수와 종류 결정
  ⑤ 플립플롭의 입력, 출력 및 각각의 상태에 문자기호 부여
  ⑥ 상태표를 이용해 회로의 상태 여기표 작성
  ⑦ 간소화 방법을 이용해 출력 함수 및 플립플롭의 입력 함수 유도
  ⑧ 순서논리회로도 작성

동기 순서회로 설계

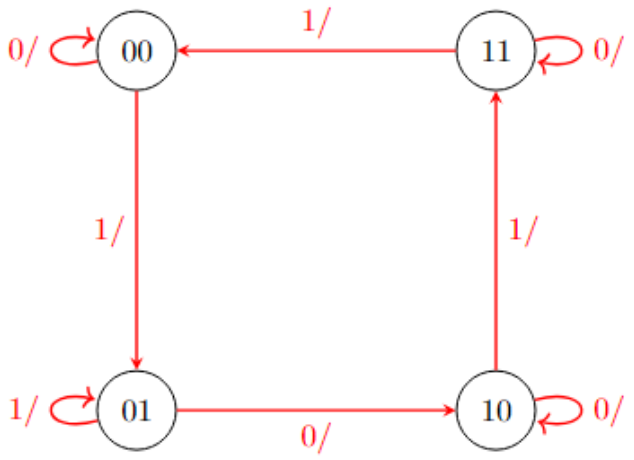
- JK 플립플롭을 이용한 순서논리회로 설계
  - ① 회로 동작 기술
    - 조합논리회로 설계와 달리 현재 상태가 다음 상태에 영향을 미침
    - 모든 상태와 이들 상태에 대한 전이 관계를 명확히 정의해 야 함
    - 플립플롭의 상태도나 다른 정보를 포함할 수 있음
    - 예, 다음과 같은 출력이 없는 상태도를 고려함



동기 순서회로 설계

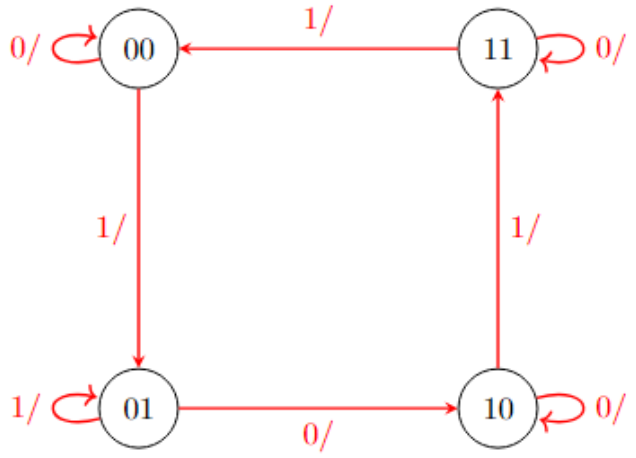
- JK 플립플롭을 이용한 순서논리회로 설계
  ② 상태표 작성

현재 상태		입력	다음 상태	
A	*B*	*x*	*A*	*B*
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	1
1	1	1	0	0



동기 순서회로 설계

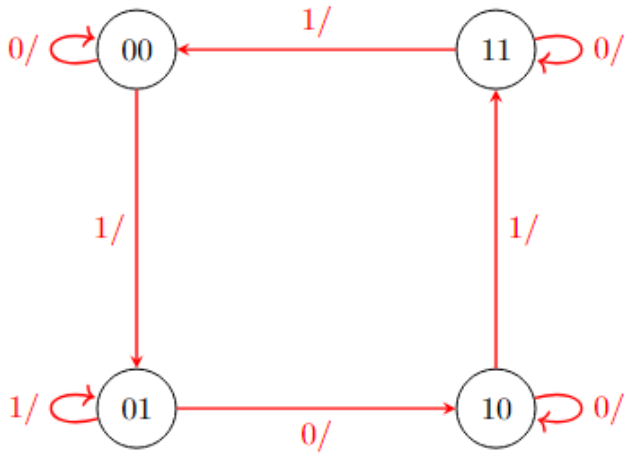
- JK 플립플롭을 이용한 순서논리회로 설계
  - ③ 플립플롭의 수와 형태 결정
    - 플립플롭의 수는 상태 수에 따라 결정됨
    - 상태 수는 N이면 플립플롭의 수는 $\lceil \log_2 N \rceil$이 됨
    - 예, $N = 4$이면 플립플롭의 수는 $\lceil \log_2 N \rceil = 2$가 됨
    - 설계할 회로 특성에 알맞으면서도 구현이 용이한 플립플롭을 선택함
    - 고려하는 예시에서 JK 플립플롭을 이용함



동기 순서회로 설계

- JK 플립플롭을 이용한 순서논리회로 설계
  ④ 상태 여기표 유도

현재 상태		입력	다음 상태		플립플롭 입력			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	x	0	x
0	0	1	0	1	0	x	1	x
0	1	0	1	0	1	x	x	1
0	1	1	0	1	0	x	x	0
1	0	0	1	0	x	0	0	x
1	0	1	1	1	x	0	1	x
1	1	0	1	1	x	0	x	0
1	1	1	0	0	x	1	x	1



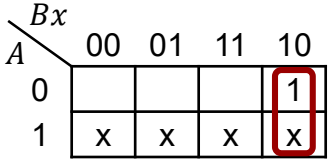
$Q(t)$	$Q(t+1)$	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

JK 플립플롭의 여기표

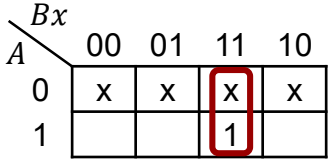
동기 순서회로 설계

- JK 플립플롭을 이용한 순서논리회로 설계
  - ⑤ 플립플롭의 입력 함수 및 회로의 출력 함수 유도

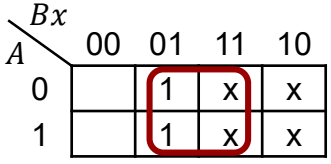
현재 상태		입력	다음 상태		플립플롭 입력			
A	B	x	A	B	J_A	K_A	J_B	K_B
0	0	0	0	0	0	x	0	x
0	0	1	0	1	0	x	1	x
0	1	0	1	0	1	x	x	1
0	1	1	0	1	0	x	x	0
1	0	0	1	0	x	0	0	x
1	0	1	1	1	x	0	1	x
1	1	0	1	1	x	0	x	0
1	1	1	0	0	x	1	x	1



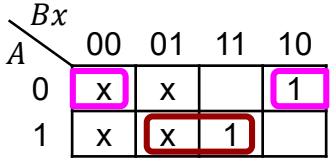
$J_A = B\bar{x}$



$K_A = Bx$



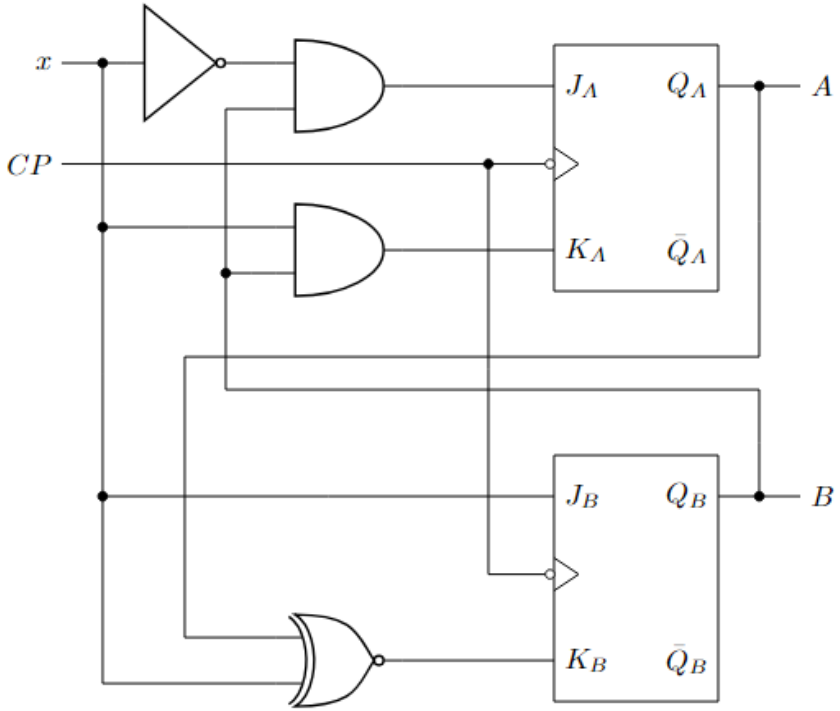
$J_B = x$

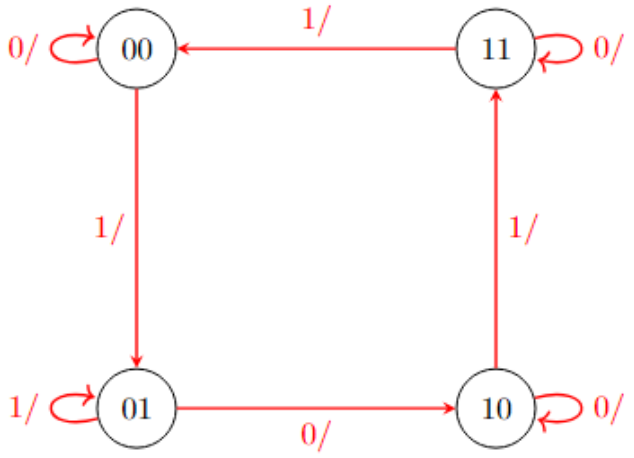


$K_B = Ax + \bar{A}\bar{x} = A \odot x$

동기 순서회로 설계

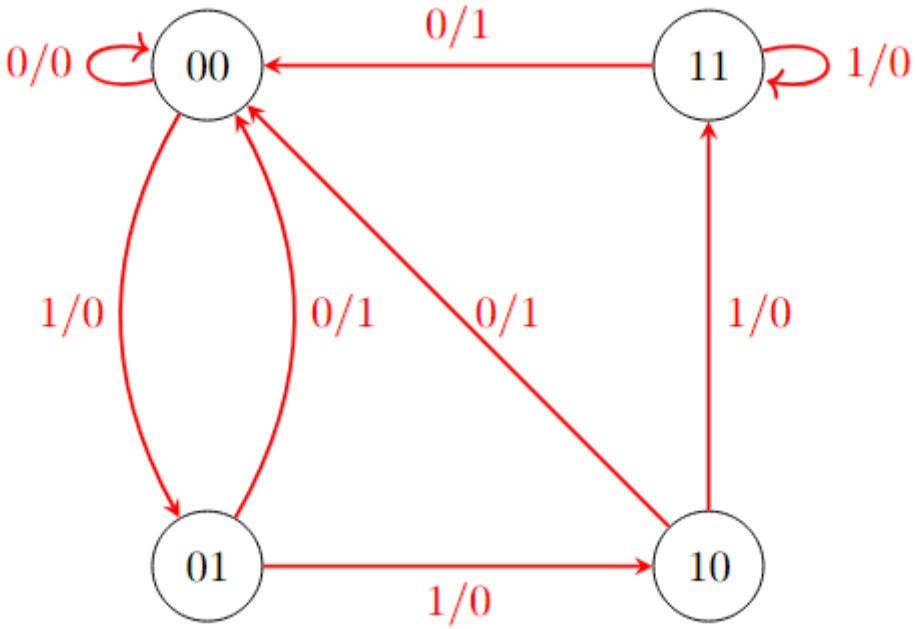
- JK 플립플롭을 이용한 순서논리회로 설계
  - ⑥ 논리회로의 구현





동기 순서회로 설계

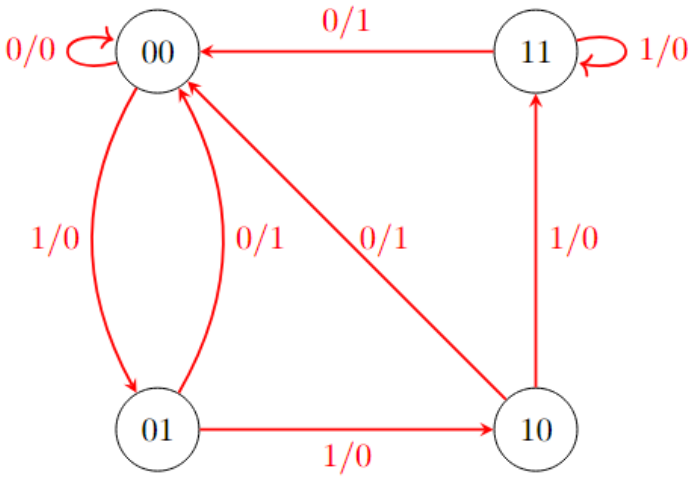
- D 플립플롭을 이용한 순서논리회로 설계
  - ① 회로 동작 기술
    - 입력 변수와 출력 변수가 모두 있는 상태표를 고려함



동기 순서회로 설계

- D 플립플롭을 이용한 순서논리회로 설계
  ② 상태표 작성

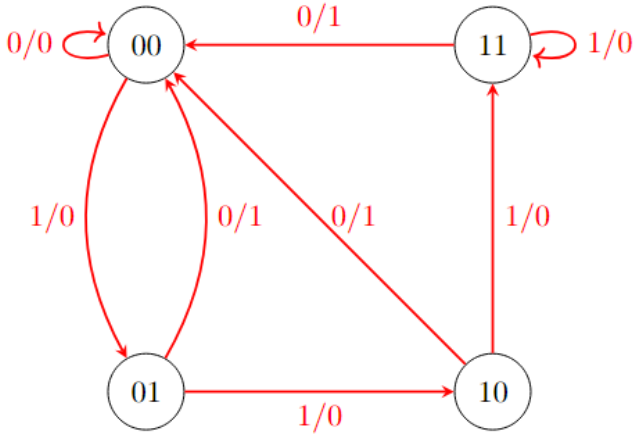
현재 상태		입력	다음 상태		출력
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0



동기 순서회로 설계

- D 플립플롭을 이용한 순서논리회로 설계
  - ③ 플립플롭의 수와 형태 결정 : D 플립플롭 2개가 필요함
  - ④ 상태 여기표 유도

현재 상태		입력	다음 상태		플립플롭 입력		출력
A	B	x	A	B	D_A	D_B	y
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	0	0	0	0	1
0	1	1	1	1	1	1	0
1	0	0	0	0	0	0	1
1	0	1	1	0	1	0	0
1	1	0	0	0	0	0	1
1	1	1	1	0	1	0	0



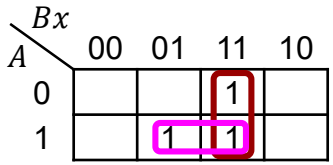
$Q(t)$	$Q(t+1)$	D
0	0	0
0	1	1
1	0	0
1	1	1

D 플립플롭의 여기표

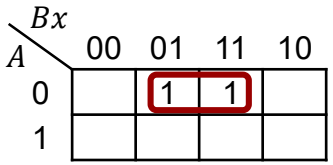
동기 순서회로 설계

- D 플립플롭을 이용한 순서논리회로 설계
  - ⑤ 플립플롭의 입력 함수 및 회로의 출력 함수 유도

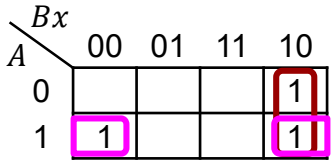
현재 상태		입력	다음 상태		플립플롭 입력		출력
A	B	x	A	B	D_A	D_B	y
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	0	0	0	0	1
0	1	1	1	1	1	1	0
1	0	0	0	0	0	0	1
1	0	1	1	0	1	0	0
1	1	0	0	0	0	0	1
1	1	1	1	0	1	0	0



$D_A = Ax + Bx = (A + B)x$



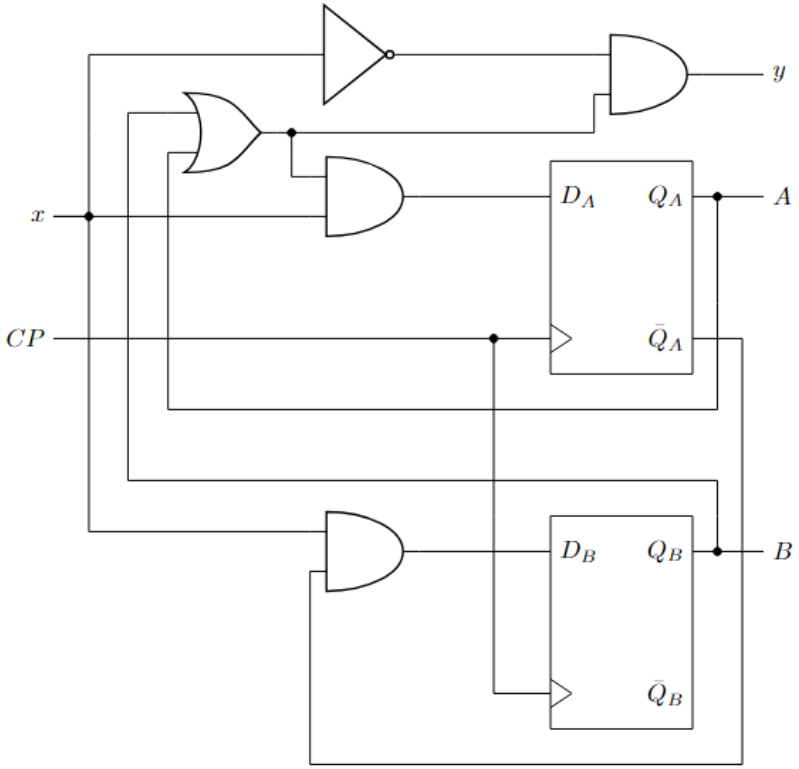
$D_B = \bar{A}x$

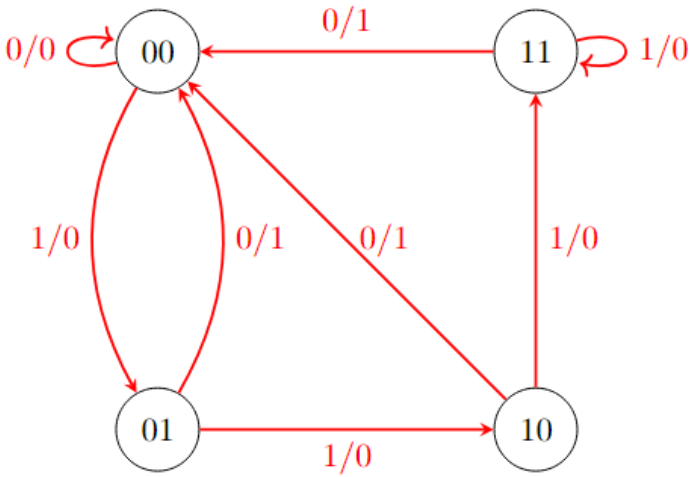


$y = A\bar{x} + B\bar{x} = (A + B)\bar{x}$

동기 순서회로 설계

- D 플립플롭을 이용한 순서논리회로 설계
  - ⑥ 논리회로의 구현

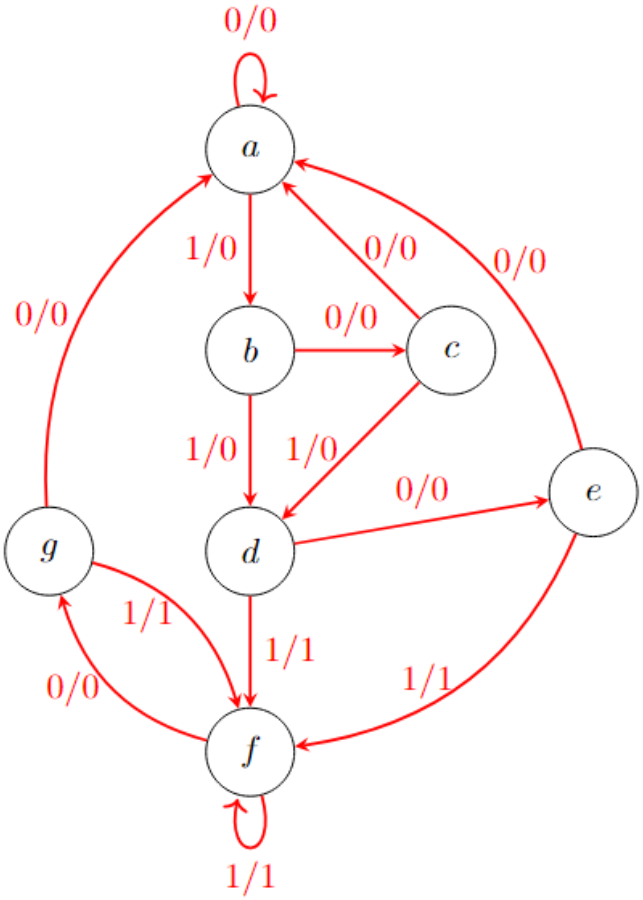




상태 축소 및 할당

- 상태 축소

현재 상태	다음 상태		출력(y)	
	$x=0$	$x=1$	$x=0$	$x=1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1



상태 축소 및 할당

- 상태 축소

현재 상태	다음 상태		출력(y)	
	$x=0$	$x=1$	$x=0$	$x=1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g	f	0	1
g	a	f	0	1

현재 상태	다음 상태		출력(y)	
	$x=0$	$x=1$	$x=0$	$x=1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g → e	f	0	1

e와 g는 서로 등가이므로 이 중에서 한 가지 상태를 제거하면 됨

상태 축소 및 할당

- 상태 축소

현재 상태	다음 상태		출력(y)	
	$x=0$	$x=1$	$x=0$	$x=1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f	0	1
e	a	f	0	1
f	g → e	f	0	1

현재 상태	다음 상태		출력(y)	
	$x=0$	$x=1$	$x=0$	$x=1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	f → d	0	1
e	a	f → d	0	1

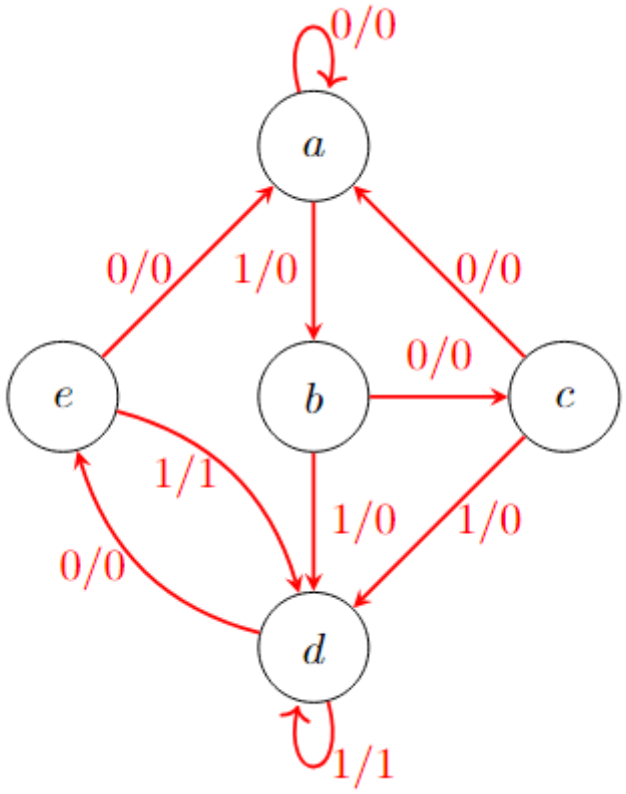
f와 d는 서로 등가이므로 이 중에서 한 가지 상태를 제거하면 됨

상태 축소 및 할당

- 상태 축소

현재 상태	다음 상태		출력(y)	
	$x=0$	$x=1$	$x=0$	$x=1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1

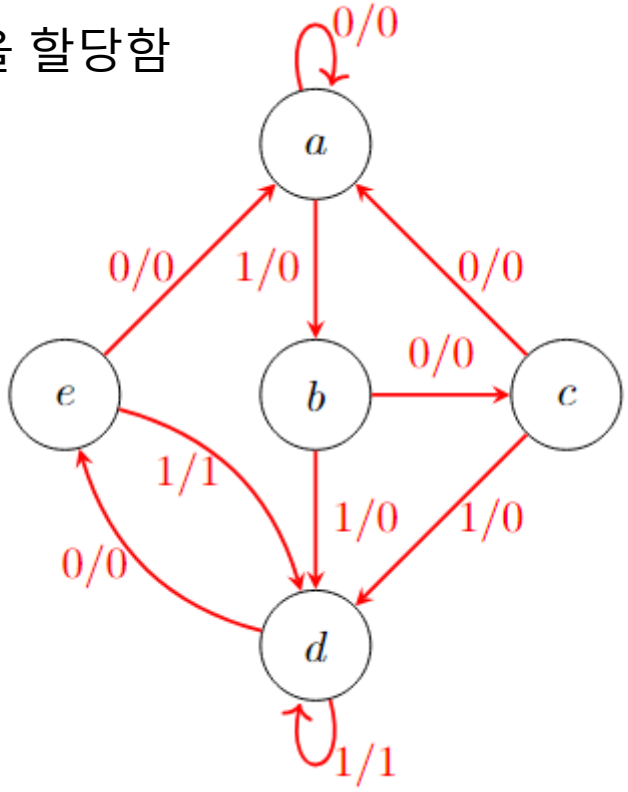
최종 상태표



상태 축소 및 할당

- 상태 할당
  - 각 상태에 대해서 2진수(2진 코드)의 값을 할당함
  - 각 상태에 고유번호를 할당하면 됨

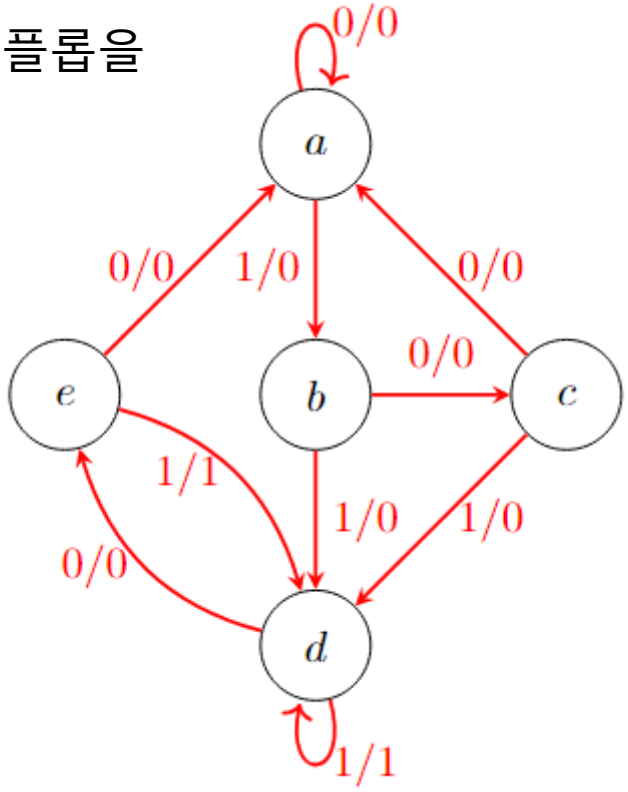
현재 상태	다음 상태		출력(y)	
	$x=0$	$x=1$	$x=0$	$x=1$
a (000)	a (000)	b (001)	0	0
b (001)	c (010)	d (011)	0	0
c (010)	a (000)	d (011)	0	0
d (011)	e (100)	d (011)	0	1
e (100)	a (000)	d (011)	0	1



상태 축소 및 할당

- 플립플롭의 수와 형태 결정
  - 상태의 수가 5가지이므로 3개의 SR 플립플롭을 사용함

현재 상태	다음 상태		출력(y)	
	$x=0$	$x=1$	$x=0$	$x=1$
a (000)	a (000)	b (001)	0	0
b (001)	c (010)	d (011)	0	0
c (010)	a (000)	d (011)	0	0
d (011)	e (100)	d (011)	0	1
e (100)	a (000)	d (011)	0	1



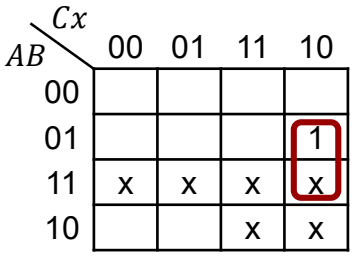
상태 축소 및 할당

- 플립플롭의 수와 형태 결정
  - 상태의 수가 5가지이므로 3개의 SR 플립플롭을 사용함

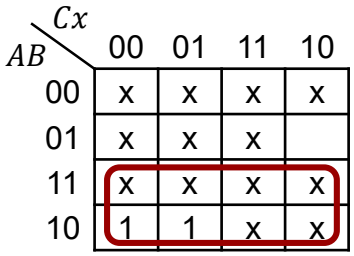
	현재 상태			외부 입력	다음 상태			플립플롭 입력						출력
	A	B	C	x	A	B	C	S_A	R_A	S_B	R_B	S_C	R_C	y
a	0	0	0	0	0	0	0	0	x	0	x	0	x	0
	0	0	0	1	0	0	1	0	x	0	x	1	0	0
b	0	0	1	0	0	1	0	0	x	1	0	0	1	0
	0	0	1	1	0	1	1	0	x	1	0	x	0	0
c	0	1	0	0	0	0	0	0	x	0	1	0	x	0
	0	1	0	1	0	1	1	0	x	x	0	1	0	0
d	0	1	1	0	1	0	0	1	0	0	1	0	1	0
	0	1	1	1	0	1	1	0	x	x	0	x	0	1
e	1	0	0	0	0	0	0	0	1	0	x	0	x	0
	1	0	0	1	0	1	1	0	1	1	0	1	0	1
무관항	1	0	1	0	x	x	x	x	x	x	x	x	x	x
	1	0	1	1	x	x	x	x	x	x	x	x	x	x
	1	1	0	0	x	x	x	x	x	x	x	x	x	x
	1	1	0	1	x	x	x	x	x	x	x	x	x	x
	1	1	1	0	x	x	x	x	x	x	x	x	x	x
	1	1	1	1	x	x	x	x	x	x	x	x	x	x

상태 축소 및 할당

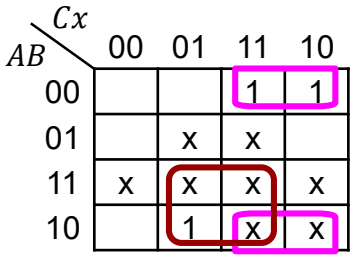
- 플립플롭의 입력 함수 및 회로의 출력 함수 유도



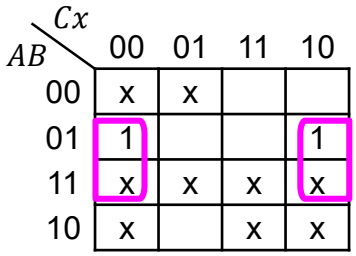
$S_A = BC\bar{x}$



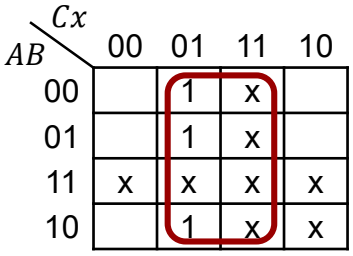
$R_A = A$



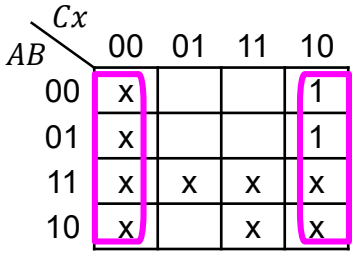
$S_B = Ax + \bar{B}C$



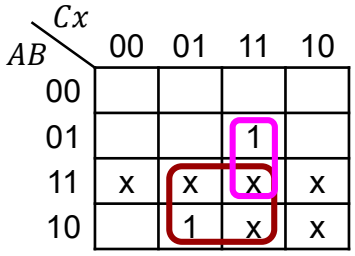
$R_B = B\bar{x}$



$S_C = x$



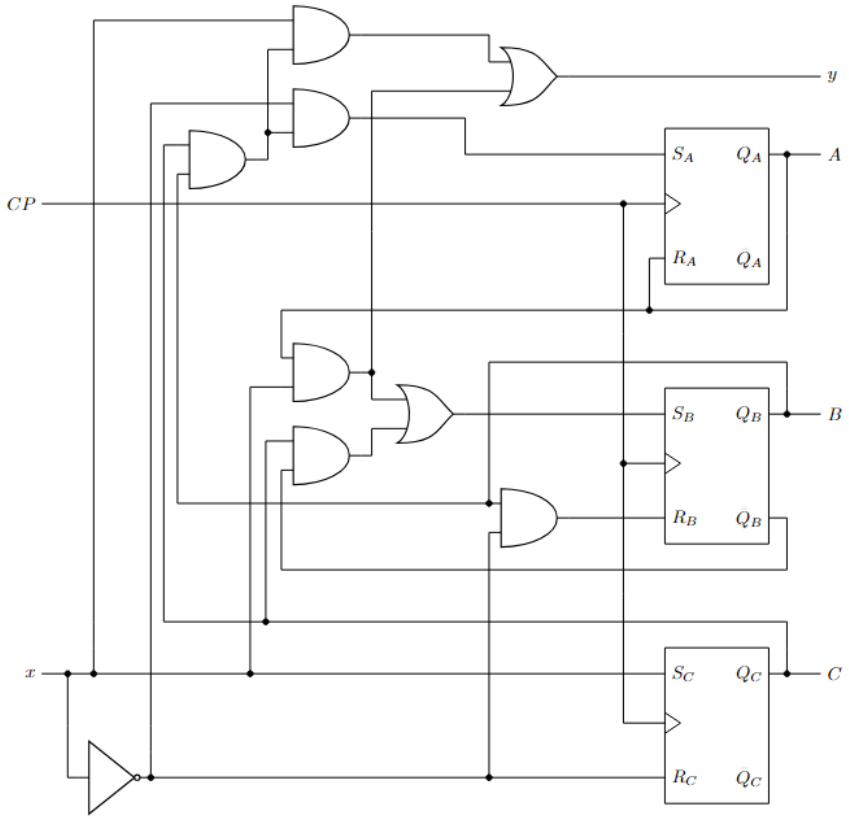
$R_C = \bar{x}$



$y = Ax + BCx$

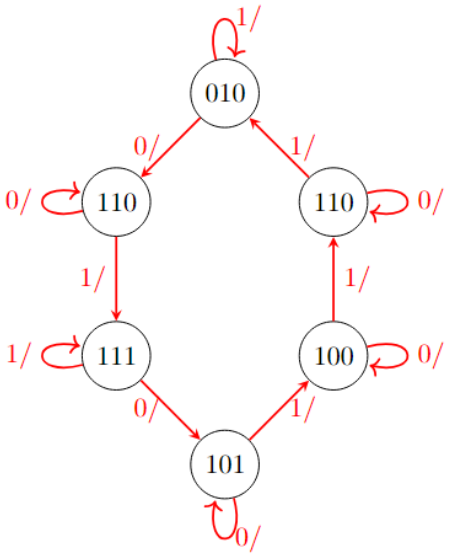
상태 축소 및 할당

- 논리회로의 구현



미사용 상태의 설계

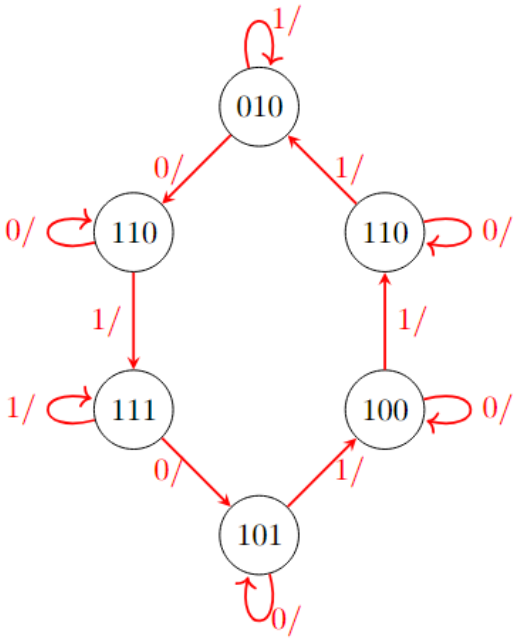
- 순서논리회로
  - m개의 플립플롭을 가지면 최대 2^m가지 현재 상태와 다음 상태를 가짐
  - 모두 상태를 사용하지 않을 수 있지만, 미사용 상태가 순서논리회로의 초기 상태가 되면 대응이 필요함
  - 예, 출력이 없고 2가지 상태(000, 001)는 사용되지 않은 상태도를 고려함



미사용 상태의 설계

- 여기표

현재 상태			입력	다음 상태			플립플롭 입력					
A	B	C	x	A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	1	0	0	0	1	1	0	x	x	0	1	x
0	1	0	1	0	1	0	0	x	x	0	0	x
0	1	1	0	0	1	1	0	x	x	0	x	0
0	1	1	1	1	1	1	1	x	x	0	x	0
1	0	0	0	1	0	0	x	0	0	x	0	x
1	0	0	1	1	1	0	x	0	1	x	0	x
1	0	1	0	1	0	1	x	0	0	x	x	0
1	0	1	1	1	0	0	x	0	0	x	x	1
1	1	0	0	1	1	0	x	0	x	0	0	x
1	1	0	1	0	1	0	x	1	x	0	0	x
1	1	1	0	1	0	1	x	0	x	1	x	0
1	1	1	1	1	1	1	x	0	x	0	x	0



$Q(t)$	$Q(t+1)$	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

미사용 상태의 설계

- 플립플롭의 입력 함수 유도

AB \ Cx	00	01	11	10
00	x	x	x	x
01			1	
11	x	x	x	x
10	x	x	x	x

$J_A = Cx$

AB \ Cx	00	01	11	10
00	x	x	x	x
01	x	x	x	x
11		1		
10				

$K_A = B\bar{C}x$

AB \ Cx	00	01	11	10
00	x	x	x	x
01	x	x	x	x
11	x	x	x	x
10		1		

$J_B = \bar{C}x$

AB \ Cx	00	01	11	10
00	x	x	x	x
01				
11				1
10	x	x	x	x

$K_B = AC\bar{x}$

AB \ Cx	00	01	11	10
00	x	x	x	x
01	1		x	x
11			x	x
10			x	x

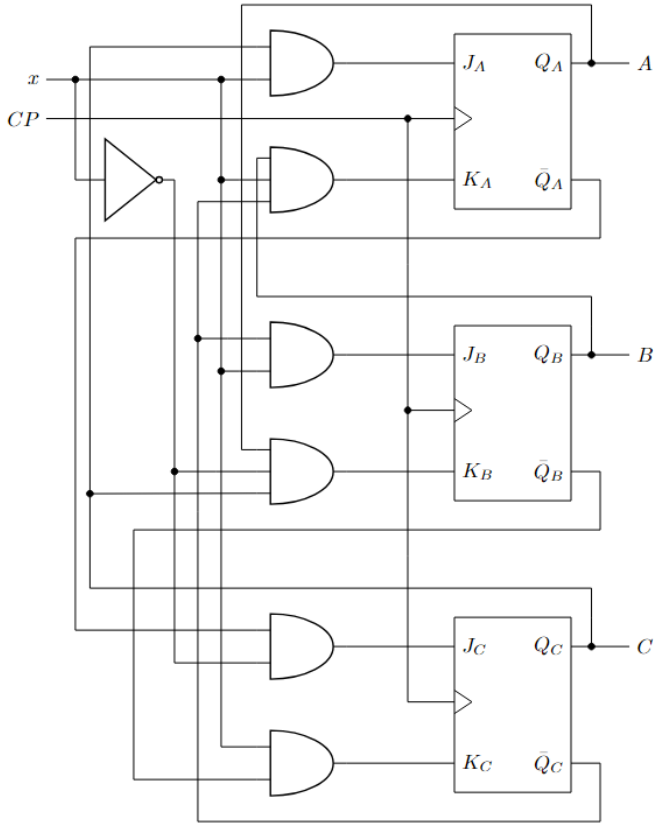
$J_C = \bar{A}\bar{x}$

AB \ Cx	00	01	11	10
00	x	x	x	x
01	x	x		
11	x	x		
10	x	x	1	

$K_C = \bar{B}x$

미사용 상태의 설계

- 논리회로의 구현

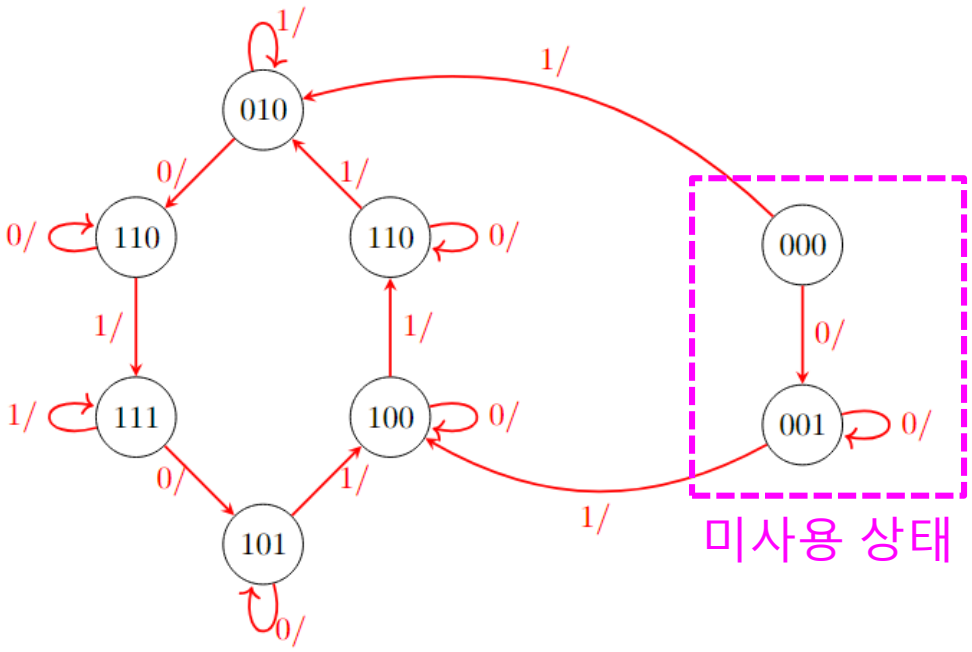


미사용 상태는 논리회로에 대입하여
다음 상태가 어떤지 구할 수 있음

현재 상태			입력	다음 상태		
A	***B***	***C***	***x***	***A***	***B***	***C***
0	0	0	0	0	0	1
0	0	0	1	0	1	0
0	0	1	0	0	0	1
0	0	1	1	1	0	0

미사용 상태의 설계

- 미사용 상태를 포함하는 상태도

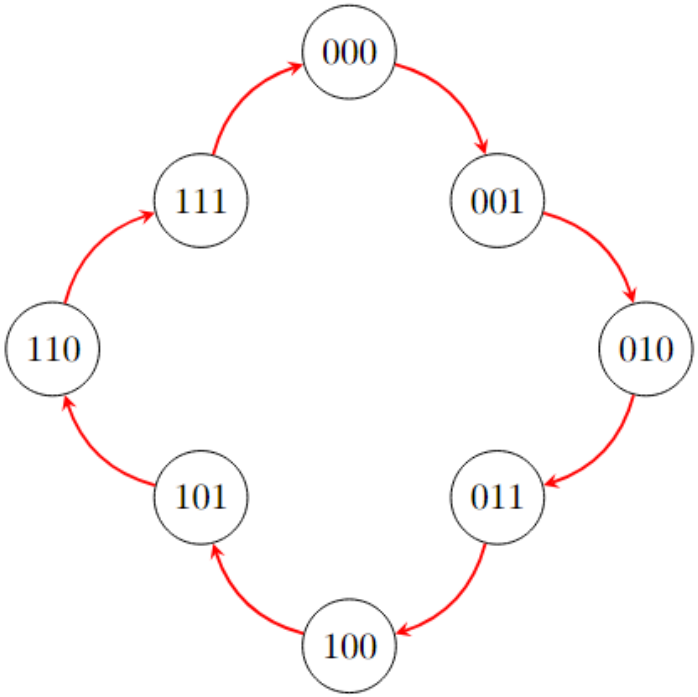


카운터의 설계

- 카운터
  - 플립플롭을 사용한 순서논리회로로, 클록 펄스가 입력될 때마다 미리 정해진 일련의 순서에 따라 상태가 변하게 됨
- 2잔 카운터
  - 설계가 가장 간단한 카운터
  - n비트 2진 카운터는 플립플롭 n개로 구성되며 0에서 $2^n - 1$까지 순서를 가질 수 있음
  - 예, 2비트 2진 카운터 : 0(00) → 1(01) → 2(10) → 3(11) → 0(00) → …
- 카운터 설계
  - 여기 가지 플립플롭을 사용할 수 있으나, 주로 토글 상태가 있는 T 플립플롭이나 JK 플립플롭을 사용하면 쉽게 설계할 수 있음

카운터의 설계

- 예, JK 플립플롭을 사용하는 3비트 2진 카운터

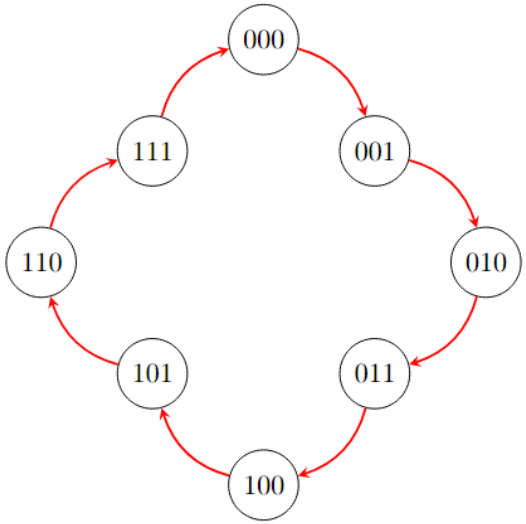


현재 상태			다음 상태		
A	*B*	*C*	*A*	*B*	*C*
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	1	1	1
1	1	1	0	0	0

카운터의 설계

- 예, JK 플립플롭을 사용하는 3비트 2진 카운터

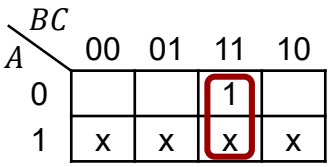
현재 상태			다음 상태			플립플롭 입력					
A	B	C	A	B	C	J_A	K_A	J_B	K_B	J_C	K_C
0	0	0	0	0	1	0	x	0	x	1	x
0	0	1	0	1	0	0	x	1	x	x	1
0	1	0	0	1	1	0	x	x	0	1	x
0	1	1	1	0	0	1	x	x	1	x	1
1	0	0	1	0	1	x	0	0	x	1	x
1	0	1	1	1	0	x	0	1	x	x	1
1	1	0	1	1	1	x	0	x	0	1	x
1	1	1	0	0	0	x	1	x	1	x	1



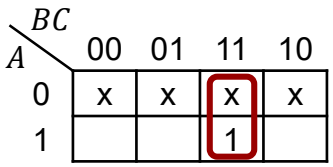
$Q(t)$	$Q(t+1)$	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0

카운터의 설계

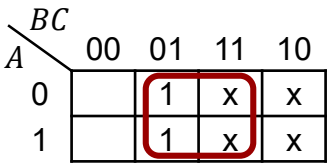
- 예, JK 플립플롭을 사용하는 3비트 2진 카운터



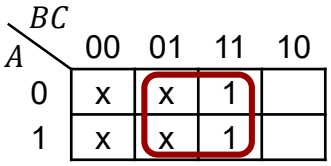
$J_A = BC$



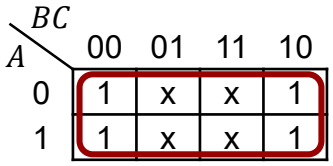
$K_A = BC$



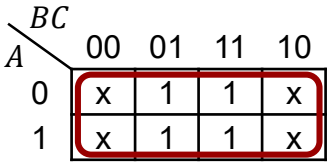
$J_B = C$



$K_B = C$



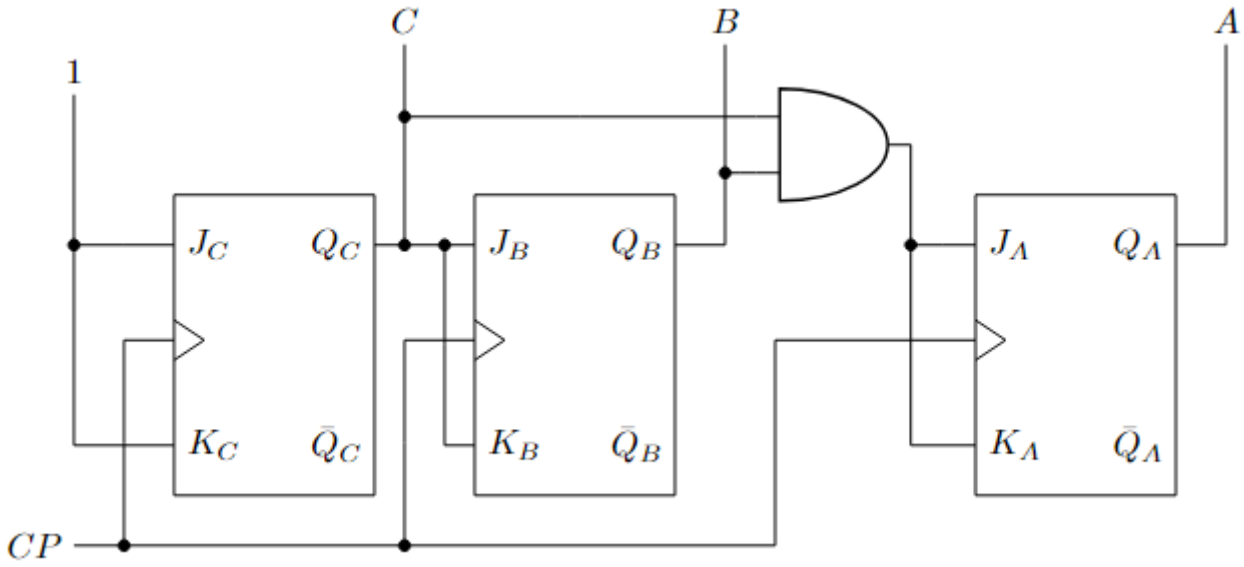
$J_C = 1$



$K_C = 1$

카운터의 설계

- 예, JK 플립플롭을 사용하는 3비트 2진 카운터



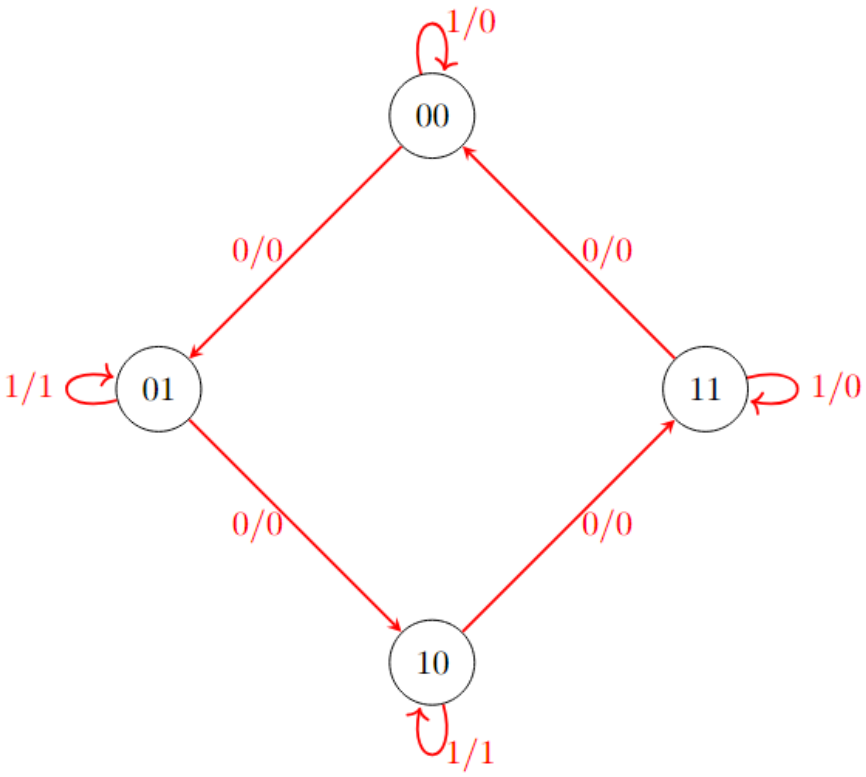
상태 방정식을 이용한 설계

- 상태 방정식
  - 현재 상태와 입력 변수의 함수로 플립플롭의 상태 변화에 관한 조건을 명시하는 불 대수식으로 다음 상태에 관한 조건이 주어지는 식임
  - 여기표보다 상태표에서 더 쉽게 유도할 수 있음
  - 상태표에 표시된 정보와 똑같은 내용을 대수적으로 표시하고 있으며, 플립플롭의 특성 방정식과 형태가 유사함
- 여기표를 사용하지 않고 상태 방정식을 이용해 순서논리회로를 설계할 수 있음
- 상태 방정식을 이용하는 경우 D 플립플롭이나 JK 플립플롭을 사용하여 순서논리회로를 설계하는 것이 더욱 편리함

상태 방정식을 이용한 설계

- JK 플립플롭을 사용한 상태 방정식

현재 상태		입력	다음 상태		출력
A	B	x	A	B	y
0	0	0	0	1	0
0	0	1	0	0	0
0	1	0	1	0	0
0	1	1	0	1	1
1	0	0	1	1	0
1	0	1	1	0	1
1	1	0	0	0	0
1	1	1	1	1	0



상태 방정식을 이용한 설계

- JK 플립플롭을 사용한 상태 방정식

현재 상태		입력	다음 상태		출력
A	B	x	A	B	y
0	0	0	0	1	0
0	0	1	0	0	0
0	1	0	1	0	0
0	1	1	0	1	1
1	0	0	1	1	0
1	0	1	1	0	1
1	1	0	0	0	0
1	1	1	1	1	0

$$A(t+1) = \bar{A}B\bar{x} + A\bar{B}\bar{x} + A\bar{B}x + ABx$$
$$B(t+1) = \bar{A}\bar{B}\bar{x} + A\bar{B}\bar{x} + \bar{A}Bx + ABx$$

JK 플립플롭의 특성 방정식
$$Q(t+1) = J\bar{Q} + \bar{K}Q$$

$$A(t+1) = \bar{A}(B\bar{x}) + A\left(\overline{\bar{B}\bar{x} + \bar{B}x + Bx}\right)$$
$$B(t+1) = \bar{B}(\bar{A}\bar{x} + A\bar{x}) + B\left(\overline{\bar{A}x + Ax}\right)$$

상태 방정식을 이용한 설계

- JK 플립플롭을 사용한 상태 방정식

현재 상태		입력	다음 상태		출력
A	B	x	A	B	y
0	0	0	0	1	0
0	0	1	0	0	0
0	1	0	1	0	0
0	1	1	0	1	1
1	0	0	1	1	0
1	0	1	1	0	1
1	1	0	0	0	0
1	1	1	1	1	0

$$A(t+1) = \bar{A}(B\bar{x}) + A\left(\overline{\bar{B}\bar{x} + \bar{B}x + Bx}\right)$$

$$B(t+1) = \bar{B}(\bar{A}\bar{x} + A\bar{x}) + B\left(\overline{\bar{A}x + Ax}\right)$$

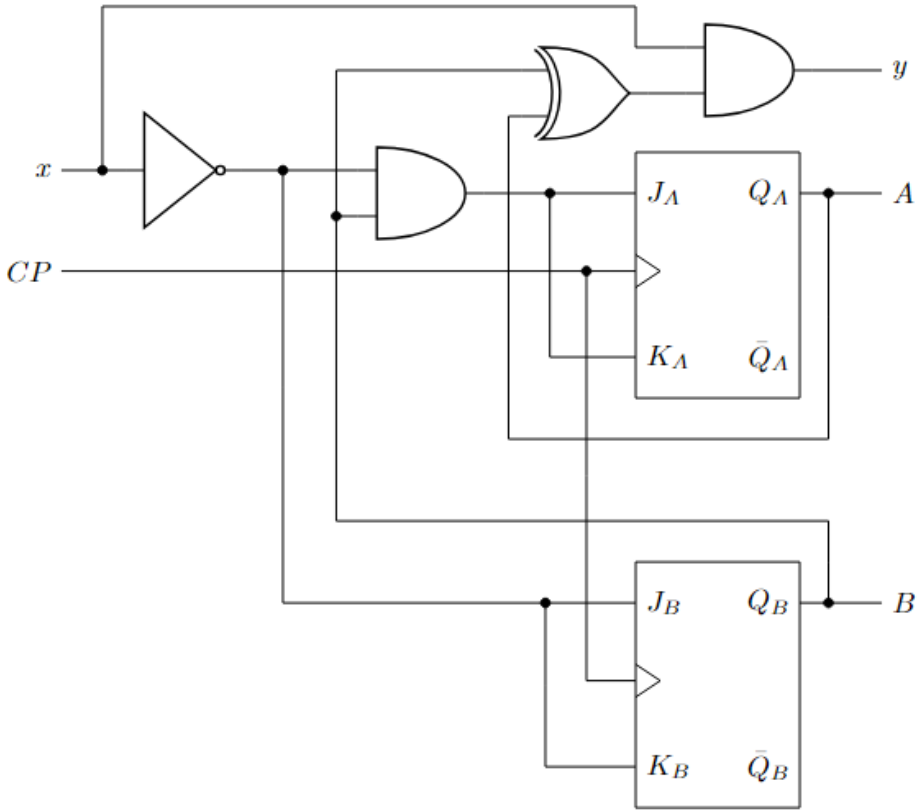
$$J_A = B\bar{x} \qquad K_A = \overline{\bar{B}\bar{x} + \bar{B}x + Bx}$$

$$J_B = \bar{A}\bar{x} + A\bar{x} = \bar{x} \qquad K_B = \overline{\bar{A}x + Ax} = \bar{x}$$

$$y = \bar{A}Bx + A\bar{B}x = x(\bar{A}B + A\bar{B}) = x(A \oplus B)$$

상태 방정식을 이용한 설계

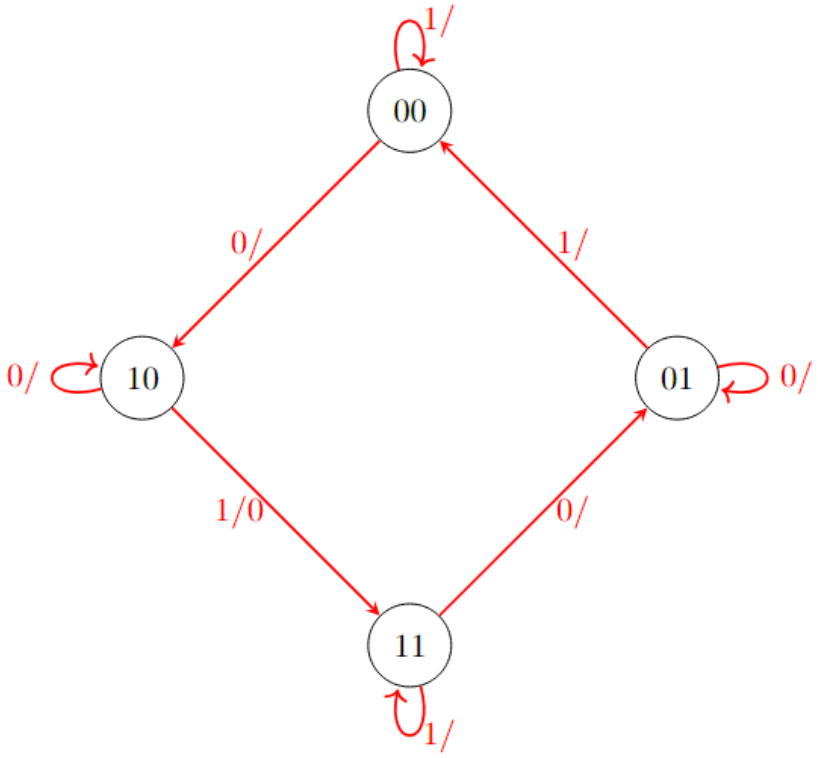
- JK 플립플롭을 사용한 상태 방정식



상태 방정식을 이용한 설계

- D 플립플롭을 사용한 상태 방정식

현재 상태		입력	다음 상태	
A	B	x	A	B
0	0	0	1	0
0	0	1	0	0
0	1	0	0	1
0	1	1	0	0
1	0	0	1	0
1	0	1	1	1
1	1	0	0	1
1	1	1	1	1



상태 방정식을 이용한 설계

- D 플립플롭을 사용한 상태 방정식

현재 상태		입력	다음 상태	
A	B	x	A	B
0	0	0	1	0
0	0	1	0	0
0	1	0	0	1
0	1	1	0	0
1	0	0	1	0
1	0	1	1	1
1	1	0	0	1
1	1	1	1	1

$$A(t+1) = \bar{A}B\bar{x} + A\bar{B}\bar{x} + A\bar{B}x + ABx$$
$$B(t+1) = \bar{A}B\bar{x} + AB\bar{x} + A\bar{B}x + ABx$$

D 플립플롭의 특성 방정식

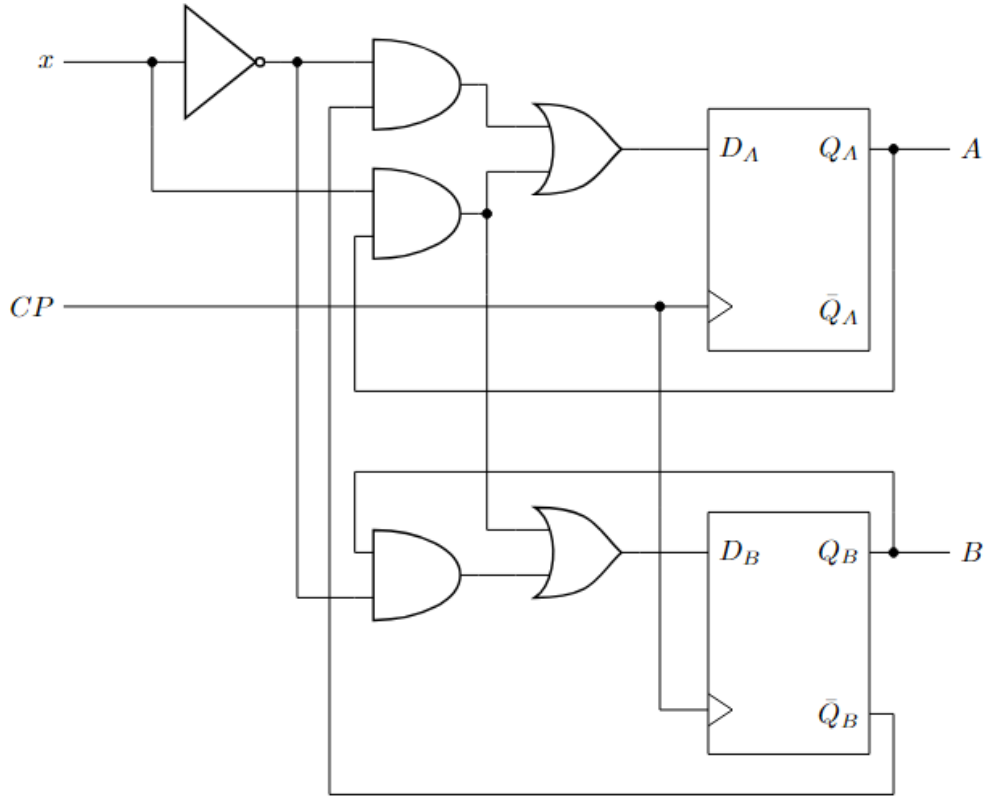
$$Q(t+1) = D$$

$$A(t+1) = \bar{B}\bar{x} + Ax$$
$$B(t+1) = B\bar{x} + Ax$$

$D_A = \bar{B}\bar{x} + Ax$ $D_B = B\bar{x} + Ax$

상태 방정식을 이용한 설계

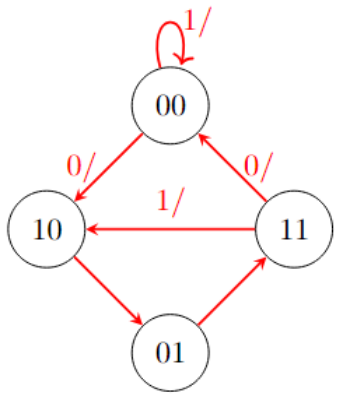
- D 플립플롭을 사용한 상태 방정식



디코더와 플립플롭을 사용한 설계

- 디코더(decoder)
  - 입력 변수 n개에 대한 최소항(minterm) 2^n개를 출력하는 기능을 수행함
  - 임의의 불 함수는 곱의 합(sum of product)형으로 표현할 수 있음
- 디코더와 플립프롭을 사용하면 순서논리회로를 설계할 수 있음
- 예, SR 플립플롭과 디코더를 사용해 다음 상태도를 기반으로 순서논리회로를 설계함

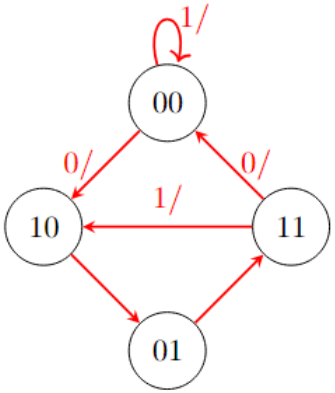
현재 상태		다음 상태			
		$x = 0$		$x = 1$	
A	B	A	B	A	B
0	0	1	0	0	0
0	1	1	1	1	1
1	0	0	1	0	1
1	1	0	0	1	0



디코더와 플립플롭을 사용한 설계

- 여기표

현재 상태		입력	다음 상태		플립플롭 입력			
A	B	x	A	B	S_A	R_A	S_B	R_B
0	0	0	1	0	1	0	0	x
0	0	1	0	0	0	x	0	x
0	1	0	1	1	1	0	x	0
0	1	1	1	1	1	0	x	0
1	0	0	0	1	0	1	1	0
1	0	1	0	1	0	1	1	0
1	1	0	0	0	0	1	0	1
1	1	1	1	0	x	0	0	1



$Q(t)$	$Q(t+1)$	S	R
0	0	0	x
0	1	1	0
1	0	0	1
1	1	x	0

디코더와 플립플롭을 사용한 설계

- 여기표

현재 상태		입력	다음 상태		플립플롭 입력			
A	B	x	A	B	S_A	R_A	S_B	R_B
0	0	0	1	0	1	0	0	x
0	0	1	0	0	0	x	0	x
0	1	0	1	1	1	0	x	0
0	1	1	1	1	1	0	x	0
1	0	0	0	1	0	1	1	0
1	0	1	0	1	0	1	1	0
1	1	0	0	0	0	1	0	1
1	1	1	1	0	x	0	0	1

$$S_A(A,B,x) = \sum m(0,2,3)$$

$$R_A(A,B,x) = \sum m(4,5,6)$$

$$S_B(A,B,x) = \sum m(4,5)$$

$$R_B(A,B,x) = \sum m(6,7)$$

디코더와 플립플롭을 사용한 설계

- 논리회로의 구현