Ch.5 Synchronous Sequential Logic - part D

Finite-state Machine(FSM)

state

주어진 시점에서의 memory에 저장된 contents

→ in sequential logic, information from past inputs is stored in memory.(ex. ff)

→ 회로에서 접근했던 모든 정보를 contain

Finite-state machine(FSM, FSA, simply a state machine)

주어진 시간 내에 finite한 개수의 state 중 정확히 한 곳에 갈 수 있는 절대적인 machine

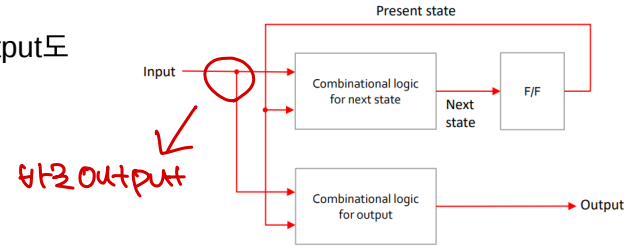
(ex. vending machine, elevators)

- **transition** : input에 대한 state 이동
- **FSM** : list of states, initial state, and input에 대한 transition

Mealy State Machine

Output- > present state, current input

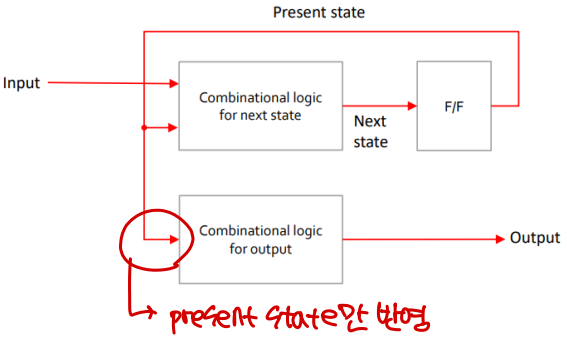
- input change → output change
  - clock input 바뀌길 기다리지 않고 output도 update



Moore State Machine

Output- > present state

- clock input change → output change
  - clock input이 바뀔 때만 output도 update



State Reduction

sequential circuit에서 **ff의 개수를 reduction**

→ input, output **변화 없이** reduction 해야 함

- equivalent state : 같은 상태에서 정확히 같은 input, output → 두 state는 equivalent (state transition 같음)

  ⇒ 하나만 있는 것이 효율적 (don't care 수) → 회로가 더 간단

  ⇒ state가 줄어들면 ff 개수도 줄어듦

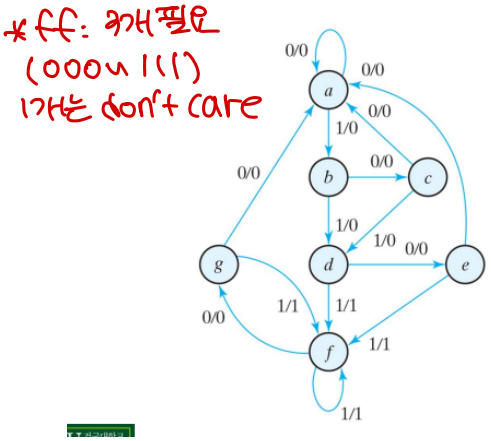
- **example** → 지울 때 동일한 transition 이외의 다른 transition 꼭 가져올 것

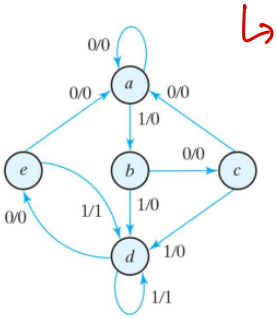
Present State	Next State $x=0$	Next State $x=1$	Output $x=0$	Output $x=1$
a	a	b	0	0
b	c	d	0	0
c	a	d	0	0
d	e	d	0	1
e	a	d	0	1
f	e	f	0	1
g	a	f	0	1

Present State	Next State $x=0$	Next State $x=1$	Output $x=0$	Output $x=1$
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

*ff: 2개필요 (000 111) 1개는 don't care

*ff: 2개필요 (3개 don't care) → 회로가 더 간단





State Assignment

M state는 log2M bits의 코드

state 개수 ┌ 1~2 : ff 1개
3~4 : ff 2개
5~8 : ff 3개
9~16 : ff 4개 ···>

Table 5-9
Three Possible Binary State Assignments ⇒ 다양한 방법이 있음

State	Assignment 1 Binary	Assignment 2 Gray code	Assignment 3 One-hot
a	000	000	00001
b	001	001	00010
c	010	011	00100
d	011	010	01000
e	100	110	10000

↳ 상태할당

Table 5-10
Reduced State Table

Present State	Next State $x=0$	Next State $x=1$	Output $x=0$	Output $x=1$
000	000	001	0	0
001	010	011	0	0
010	000	011	0	0
011	100	011	0	1
100	000	011	0	1

Sequential circuit design ↔ analysis

1. state diagram → state table 만들기 → present state, input | next state, output
2. state table로 state에 binary code 할당(n개의 state → log2n개의 ff)
3. 각 ff의 input value 찾기(ff excitation table 활용)
4. ff의 input, output equation 찾기
5. 회로 설계

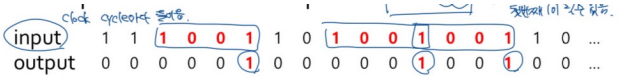
Sequence recognizers

↳ DFA 만들 때 s_0, s_1 ··· setting 했던 것처럼 가능한 경우 생각

a special kind of sequentail circuit that looks for a special bit pattern in some input

→ 지금까지 들어온 input에 대해 'remember' 해야 함.

↳ 한 bit씩 check하기에는 상태수 → 너무 많아짐

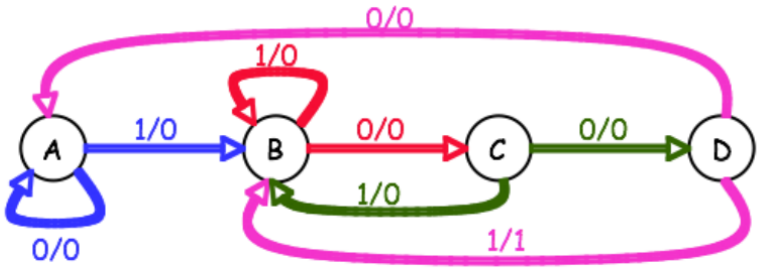


↳ output은 1개

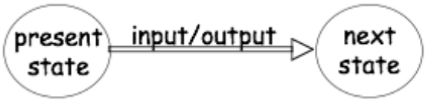
example - JK ff

"1001"을 recognize

1. making a state table
   ㄱ) sequence recognizer 생각해본 것 ⇒ 각 state에서 2^n개 arrow
   ① start state
   ② 1만 들어왔을 때 → 1이 들어오면 ①로, 0이면 다음 ③으로
   ③ 10이 들어왔을 때 → 1이 〃 ②로, 0이면 ④로
   ④ 100 〃 → 1이 〃 ①로, 0이면 ①로
   ⇒ 1001 ↳ 처음부터 다시



Remember how the state diagram arrows correspond to rows of the state table:



→ transition 하나가 row 하나

Present State	Input	Next State	Output
A	0	A	0
A	1	B	0
B	0	C	0
B	1	B	0
C	0	D	0
C	1	B	0
D	0	A	0
D	1	B	1

2. assigning binary codes to states

2가지의 state → 2개의 ff

Present State	Input	Next State	Output
A	0	A	0
A	1	B	0
B	0	C	0
B	1	B	0
C	0	D	0
C	1	B	0
D	0	A	0
D	1	B	1

Present State Q_1	Q_0	Input X	Next State Q_1	Q_0	Output Z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	1	0	0
0	1	1	0	1	0
1	0	0	1	1	0
1	0	1	0	1	0
1	1	0	0	0	0
1	1	1	0	1	1

3. finding ff input values

ff input 형이 달라짐

Present State Q_1	Q_0	Input X	Next State Q_1	Q_0	Flip flop inputs J_1	K_1	J_0	K_0	Output Z
0	0	0	0	0					0
0	0	1	0	1					0
0	1	0	1	0					0
0	1	1	0	1					0
1	0	0	1	1					0
1	0	1	0	1					0
1	1	0	0	0					0
1	1	1	0	1					1

1) ff 종류고르기

: ff 종류에 따라 효율이 달라짐 → 이 예시에서는 JK 두 개 사용

- JK ff input values
  - characteristic table

(여기표)

- excitation table

J	K	Q(t+1)	Operation
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	Set
1	1	Q'(t)	Complement

변형해서 채우기

다 하기에는 번거로움
ex) 0→0
① no change
② reset
둘 중 하나
⇒ J | K
0 | 0
0 | 1 ↳ don't care
⇒ J | K
0 | X

Q(t)	Q(t+1)	J	K	Operation
0	0	0	x	No change/reset
0	1	1	x	Set/complement
1	0	x	1	Reset/complement
1	1	x	0	No change/set

⇒ 표 채우기

Present State Q_1	Q_0	Input X	Next State Q_1	Q_0	Flip flop inputs J_1	K_1	J_0	K_0	Output Z
0	0	0	0	0	0	x	0	x	0
0	0	1	0	1	0	x	1	x	0
0	1	0	1	0	1	x	x	1	0
0	1	1	0	1	0	x	x	0	0
1	0	0	1	1	x	0	1	x	0
1	0	1	0	1	x	1	1	x	0
1	1	0	0	0	x	1	x	1	0
1	1	1	0	1	x	1	x	0	1

4. find equations for the ff input, output

: k-map 이용하여 방정식 찾음 ⇒ present state, input 으로 each 에 대한 k-map

J_1

Q_1 \ Q_0	0	1
0	X	X
1	X	X

X

$J_1 = Q_0 X'$

J_0

Q_1 \ Q_0	0	1
0	1	X
1	1	X

X

$J_0 = X + Q_1$

K_1

Q_1 \ Q_0	0	1
0	X	X
1	1	1

X

$K_1 = Q_0 + X$

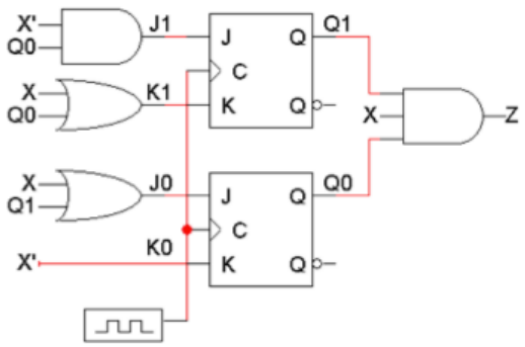
K_0

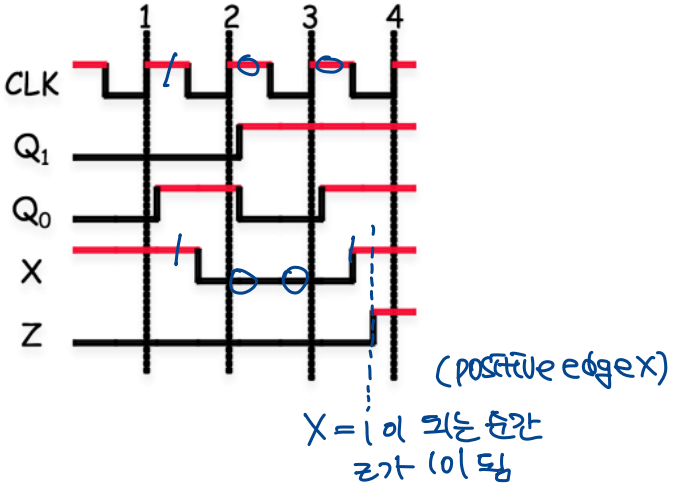
Q_1 \ Q_0	0	1
0	X	1
1	X	1

X

$K_0 = X'$

5. build the circuit





example - D ff

Present State Q_1	Q_0	Input X	Next State Q_1	Q_0	Flip-flop inputs D_1	D_0	Output Z
0	0	0	0	0	0	0	0
0	0	1	0	1	0	1	0
0	1	0	1	0	1	0	0
0	1	1	0	1	0	1	0
1	0	0	1	1	1	1	0
1	0	1	0	1	0	1	0
1	1	0	0	0	0	0	0
1	1	1	0	1	0	1	1

① D ff characteristic table

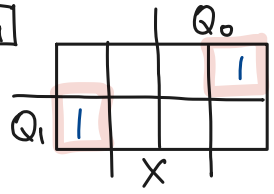
D	Q(t+1)	operation
0	0	Reset
1	1	set

② D ff excitation table

Q(t)	Q(t+1)	D	operation
0	0	0	Reset
0	1	1	set
1	0	0	Reset
1	1	1	set

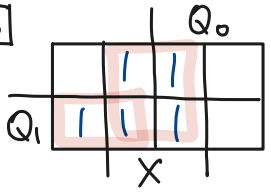
③ D_1, D_0 equation → k map

D_1



$D_1 = Q_0Q_1'X' + Q_0'Q_1X'$

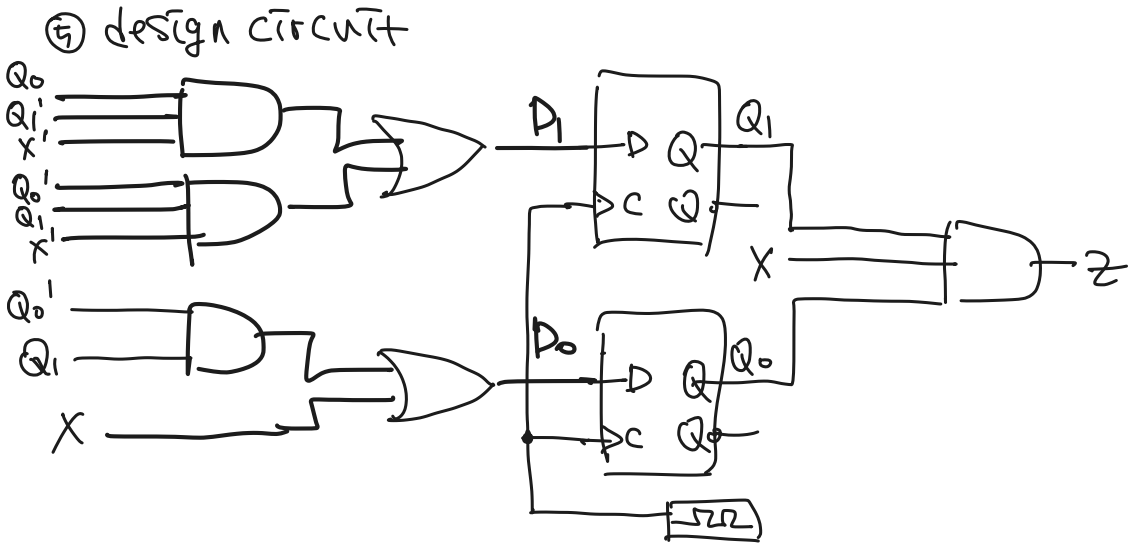
D_0



$D_0 = X + Q_0'Q_1$

④ Z equation

$Z = Q_1Q_0X$

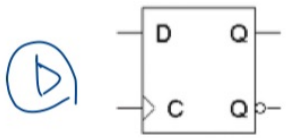


ff comprison

- **JK ff** : don’t care 값이 많아서 간단한 회로를 그릴 수 있음
- **D ff** : input이 하나만 있어서 input equation을 만드는데 쉬움

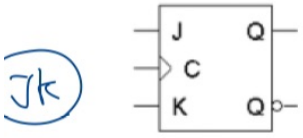
→ 실전에서 D가 더 자주 쓰임.

excitation table



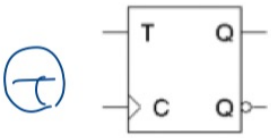
Q(t)	Q(t+1)	D	Operation
0	0	0	Reset
0	1	1	Set
1	0	0	Reset
1	1	1	Set

$Q(t+1) = D$



Q(t)	Q(t+1)	J	K	Operation
0	0	0	x	No change/reset
0	1	1	x	Set/complement
1	0	x	1	Reset/complement
1	1	x	0	No change/set

$Q(t+1) = K'Q(t) + JQ(t)'$



Q(t)	Q(t+1)	T	Operation
0	0	0	No change
0	1	1	Complement
1	0	1	Complement
1	1	0	No change

$Q(t+1) = T \oplus Q(t)$