Ch.6 Registers and Counters - part B

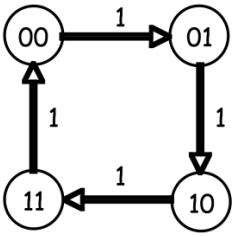
Counters

clock cycle마다 하나씩 count up 하는 specific type of sequential circuit.

→ register처럼 별도의 output 필요 없이 state 자체를 output으로 !

→ 가장 큰 값에 도달하면 0인 state로 돌아옴

	Present State		Next State	
	A	B	A	B
0	0	0	0	1
1	0	1	1	0
2	1	0	1	1
3	1	1	0	0

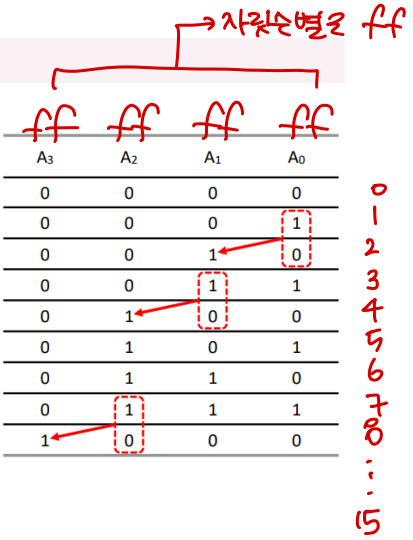


- counter는 **시계**처럼 사용 가능
  - 실행 시간 측정 가능
  - 모든 processor들은 program counter, PC를 가지고 있음
    - 몇번째 명령어인지 check

Ripple Counter ⇒ (싹) 계속증가하는 counter

다른 ff를 triggering 하기 위한 ff output transition server

→ 자신의 하위 bit가 1→0으로 바뀌면 자신은 complement



(series connection of complementing ff)

① A_0 : 계속 complement ⇒ T input에 1을 줌

② A_1 : A_0가 1에서 0으로 바뀔 때 → A_1 complement (negative edge)

⇒ A_1의 clk에 A_0을 not로 연결

③ A_2 : A_1 〃 → A_2 〃

⇒ A_2 〃 A_1 〃

④ A_3도 동일

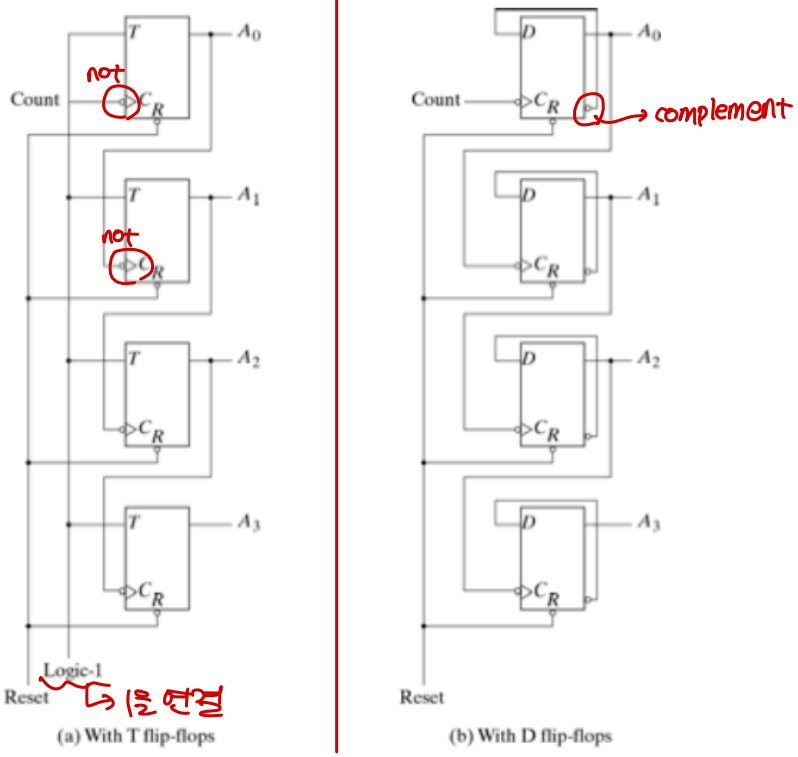
- **4bit Binary Ripple Counter**

  → clock을 변경하는 회로

  → 바람직하지 x

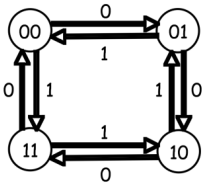
*T ff : ① T=0 → no change
② T=1 → complement

*D ff : ① D=0 → 0
② D=1 → 1
⇓
그동 n이 걸기



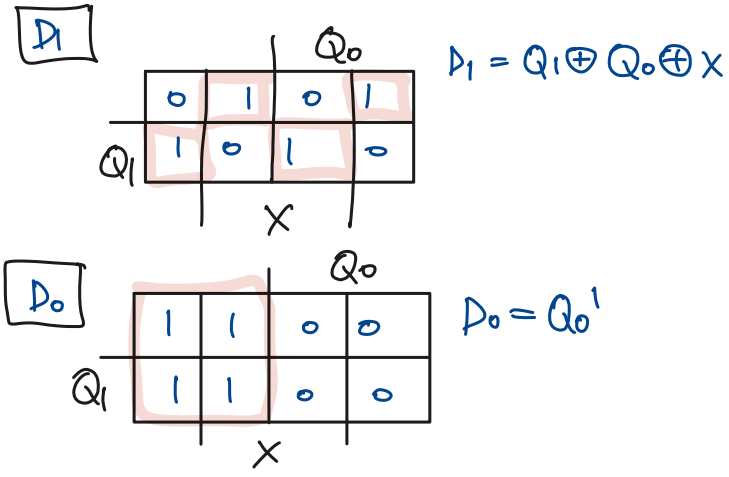
(1씩) ⇒ 4개의 State

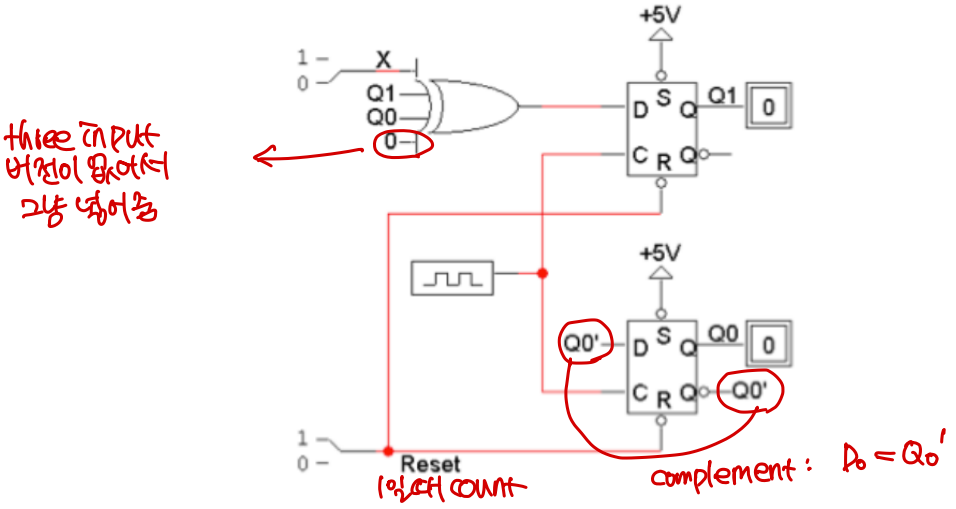
- **two-bit counter → slightly fancier**
  - X = 0 → count up(increment)
  - X = 1 → count down(decrement)



Present State Q_1	Q_0	Inputs X	Next State Q_1	Q_0
0	0	0	0	1
0	0	1	1	1
0	1	0	1	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	0

  - **D ff inputs**



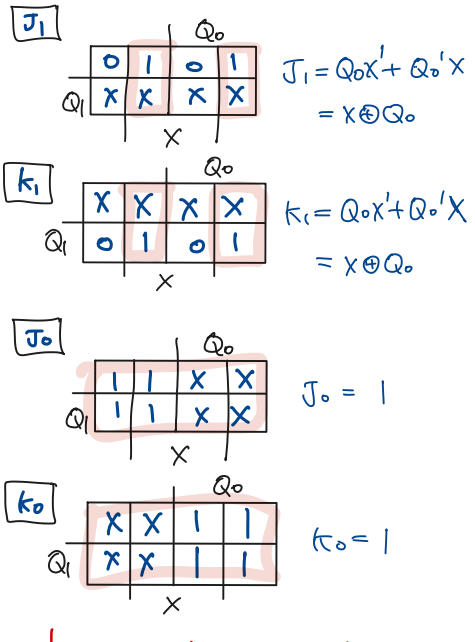


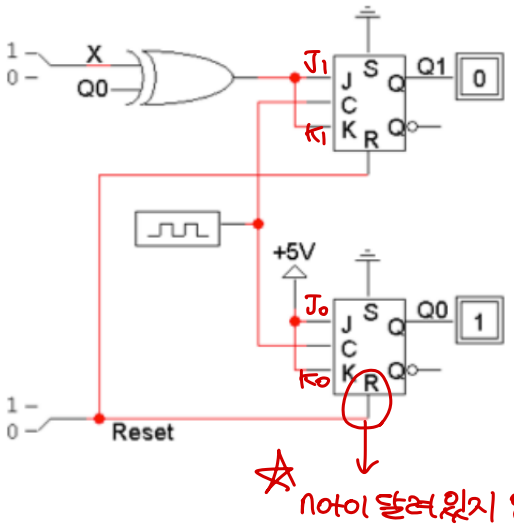
- nomal output, complemented output → Q'0에 inverter 없이 직접 접근 가능
- Reset = 1 → count
- Reset = 0 → 00으로 초기화
- input 한 개만 필요

- **JK ff inputs**

여기표로 계산할것

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





- JK ff n.i. RS 사용

    - n.i. RS : input R, S에 non-inverted, or active high
  - Reset = 0 → count
  - Reset = 1 → 00으로 초기화

Unused states

2^n개의 state가 있을 때 n개의 ff이 사용되지만 몇몇 state들이 unused

- example ↳ 6개의 상태 → 3개의 ff 필요함
  - 0(000), 5(101) unused → don't care condition으로 변경! (더 간단한 회로)
  - **but**, unused에서 계속 머무르고 빠져나올 수 없을 수도 있음 (정상이 보장되도록)

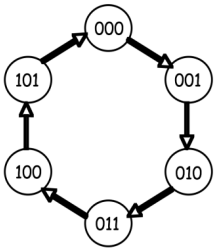
    → self-starting counter : don't care를 starting state로 수정!

    → unused state에 들어갔다는 것을 예측 가능

① don't care

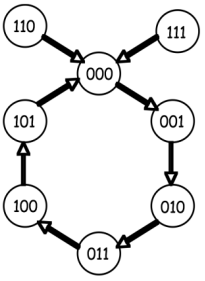
Present State			Next State		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	0	1	1
0	1	1	1	0	0
1	0	0	1	0	1
1	0	1	0	0	0
1	1	0	x	x	x
1	1	1	x	x	x

2개는 어떻게?



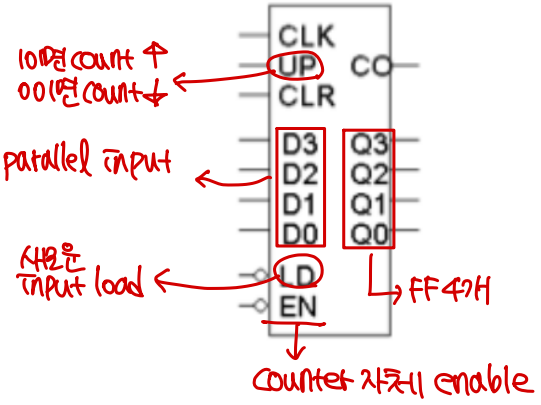
② self starting

Present State			Next State		
Q_2	Q_1	Q_0	Q_2	Q_1	Q_0
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	0	0	0
1	1	0	0	0	0
1	1	1	0	0	0



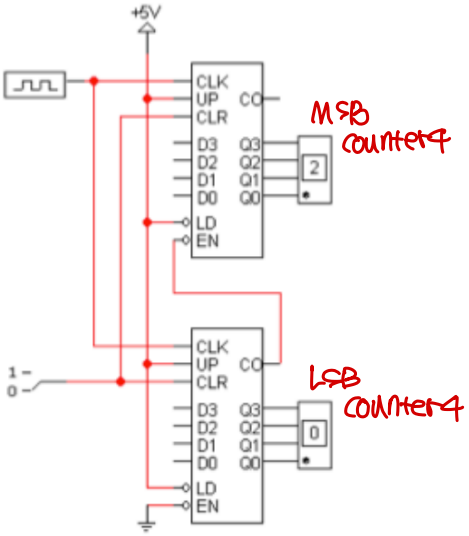
More complex counters

- **Counter-4** (0~15까지 count 가능)
  - UP input
    - 1 → count up / 0 → count down
  - CLR input
    - 1 → 0000으로 setting (asynchronously clear)
  - LD input (active low)
    - 0 → 새로운 input load ⇒ count 초기값 설정
  - EN input
    - enable, or disable
    - counter가 disable되면 output이 계속 같은 값만 출력
  - D3-D0 output



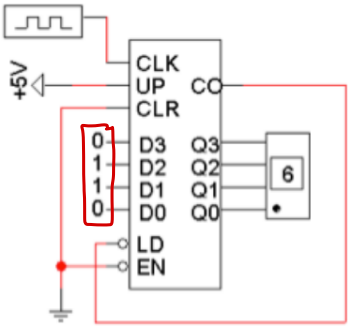
- CO output
  - 0 → 최댓값(1111)에 도달, 1 → normal

- **8-bit counter**
  - two 4-bit counters
  - 아래쪽(하위비트) counter가 1111에 도달(CO=0)
    → 위쪽(상위비트) counter enable
  - clock, clear signal 공유하여 사용
  - Hex(16bit)도 이와 같이 사용



- **restricted 4-bit counter**

1. **start** : 초기값을 결정할 수 있는 제한된 counter



2. **count up** : 최댓값이 제한된 counter → 최대에 도달하면 0000으로 돌아옴

⇒최댓값 : 1100