Ch.5 Synchronous Sequential Logic - part B

introduce

- **latch's problem**
  1. when to enable a latch
  2. need to quickly diable a latch

  → difficult to control the timing of latches in a large circuit.

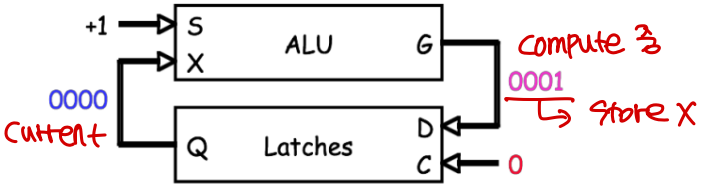
① ②

⇒ clocks, and filp-flops 해결!

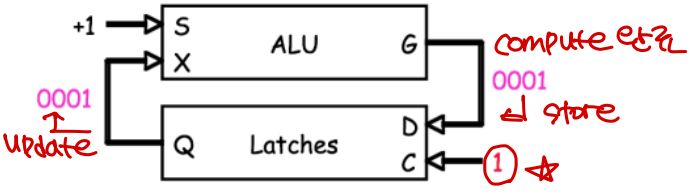
1. **clock** : 언제 memory에 쓸 지 알려줌
2. **ff** : 정해진 시간 내에 정확히 memory에 빨리 wirte 할 수 있도록 해줌

- **example** - ALU
  - latch = ALU의 memory로 사용 , 0000으로 초기값 저장하는 4개의 latch + 0001로 연산!
    1. latch가 disable되어 원치 않는 data가 저장되는 것을 막아야 함

       → ALU가 새로운 값을 계산할 동안 latch가 disable ⇒ 저장될 수 x ☆



    2. ALU의 계산이 끝나면 latch가 enable 되어 새로운 값이 저장되어야 함



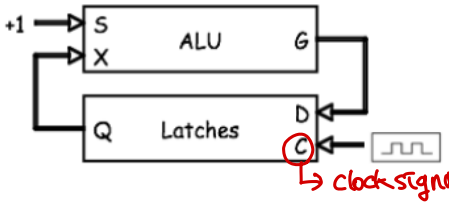
    3. latch가 계속 빠르게 disable되어야

ALU가 새로운 값을 읽고 새로운 결과 값을 낼 시간이 생김

- latch를 사용하기 위해 꼭 필요한 것
  1. 새로운 값이 저장될 준비가 될 때까지 latch가 disable
     - ready 되었는지 어떻게? → add another signal to our circuit. → clock 사용 ★
       - signal = 1 → latch가 ALU 계산 끝 + data가 저장될 준비 완료로 판단

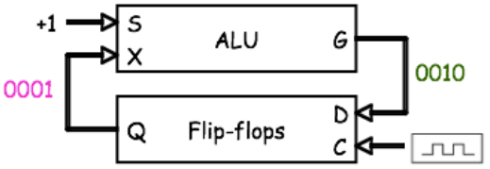
(clock)

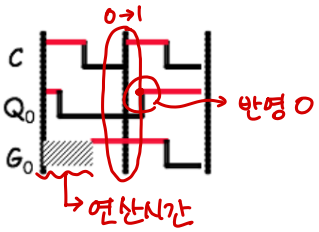
* clock period
  알맞게 설정해야함
  ⇒ 연산 끝나는 시간에 맞게
  [너무 짧으면 연산 끝나기 전에 write
  [너무 길면 또 새로운 result를 만들어냄



  2. latch가 새로운 값이 update 될 동안 적당히 enable
     - 어떻게 enable, 또 얼마나 빠르게 disable? → ff 사용으로 해결!

① direct input : 0000
② positive edge에만 반영





Clock

a special device that whose output continuously alternates btw 0 and 1.

- **clock period**(clock cycle time) : clock이 바뀌는데 걸리는 시간
- **clock frequency** : clock period의 역수 (**=hertz**)

• symbol

★
→ synchronize cirucits에 주로 사용!

1. generate a repeating, predictable pattern of 0s and 1s

   → can trigger certain events in a circuit.

   ex. writing to latch

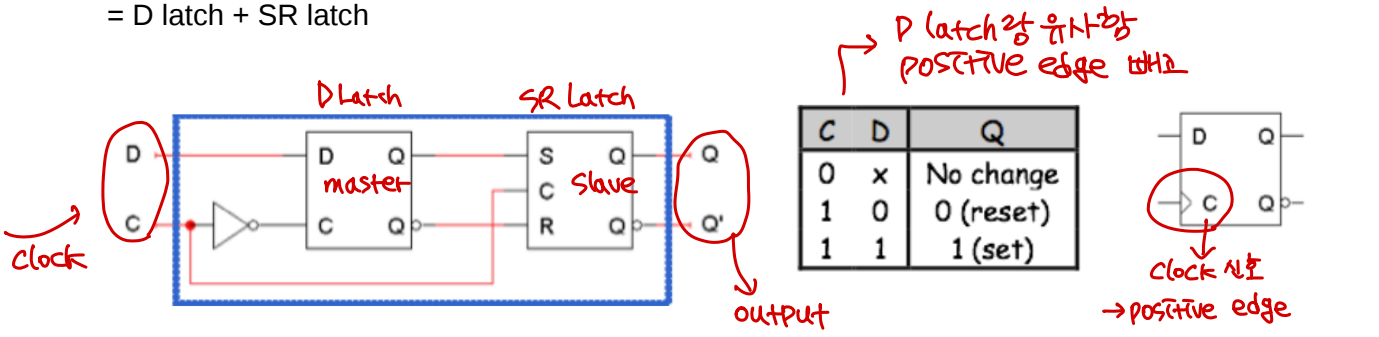
2. 만약 여러 circuits이 같은 clock signal을 사용 → 서로 동작 조정 가능

Filp-Flops

latch를 어떻게 잠깐 enable 시킬 지 → ff가 해결!

D filp-flop

= D latch + SR latch



input이 그대로 저장

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

- **C=0**
  - D latch(master) : enable → D input이 바뀌면 output도 바뀜
  - SR latch(slave) : disable → D의 output이 바뀌어도 영향 x, 현재 값 유지

  ⇒ No change
- **C=1**
  - D latch(master) : disable → C=1이 되기 전의 마지막 D input 유지, no change
  - SR latch(slave) : enable → D의 output 영향 o

  ⇒ output 바뀜

⇒ C가 0에서 1로 바뀌는 순간에만 ff output이 바뀜(new input이 반영 됨)

⇒ ***positive edge-tirggered flip-flop (D latch와 다른 점)***

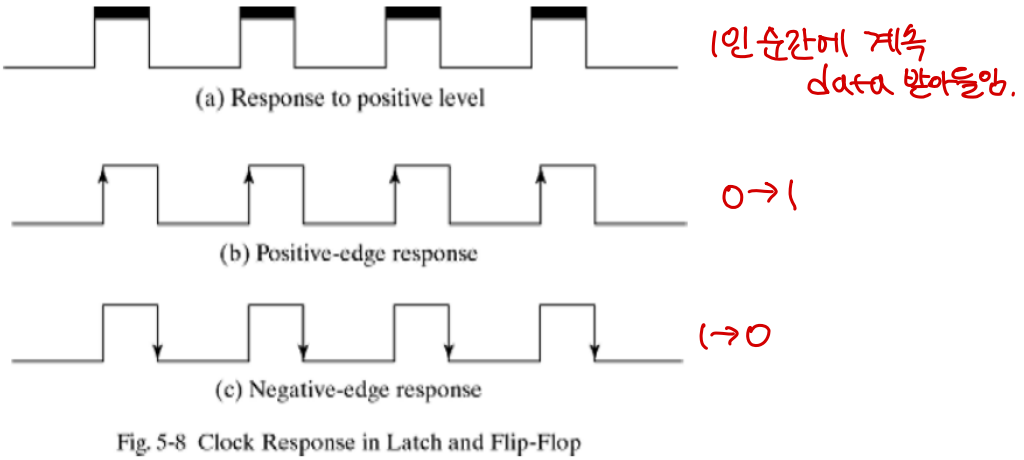


Fig. 5-8 Clock Response in Latch and Flip-Flop

⇒ 언제 data를 받아들이냐에 따라서 Latch와 큰 차이!

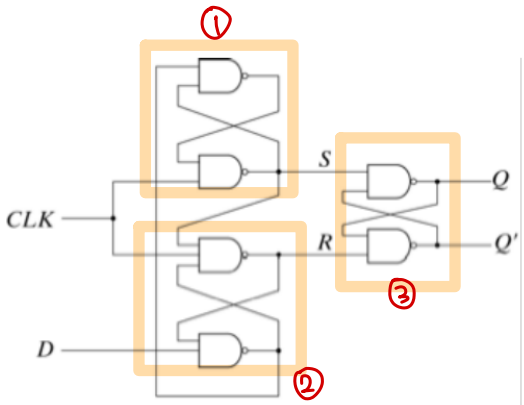


Fig. 5-10 *D*-Type Positive-Edge-Triggered Flip-Flop

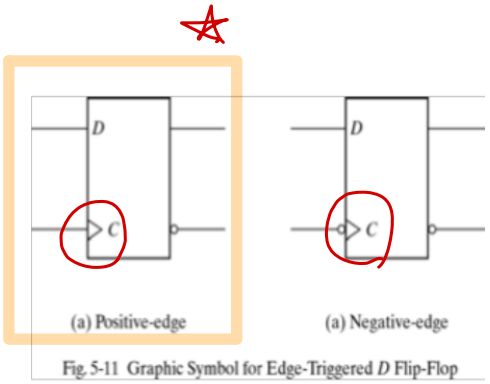
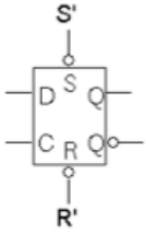


Fig. 5-11 Graphic Symbol for Edge-Triggered *D* Flip-Flop

- **Direct Inputs**
  - Q의 starting value → direct(asynchronous) input
    - ff 초기화하기 위해 set을 초기값으로 줌
    - next Q부터는 synchronous하게 동작

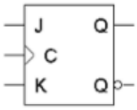


S'	R'	C	D	Q	
0	0	x	x	Avoid!	
0	1	x	x	1 (set)	Direct inputs to set or reset the flip-flop
1	0	x	x	0 (reset)	
1	1	0	x	No change	S'R' = 11 for "normal" operation of the D flip-flop
1	1	1	0	0 (reset)	
1	1	1	1	1 (set)	

JK flip-flop

SR latch와 비슷하지만,

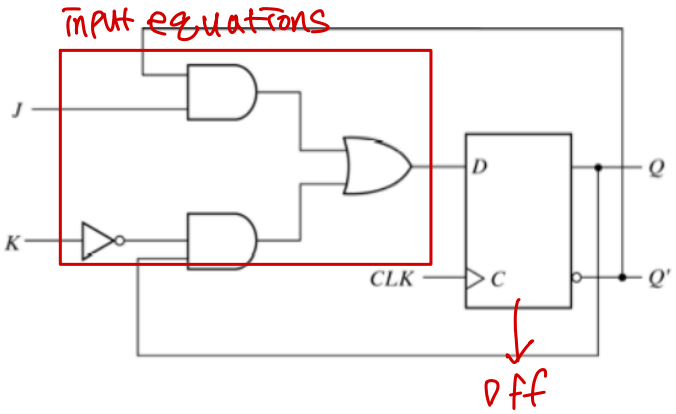
JK=11 → current의 complement 출력



C	J	K	Q_{next}
0	x	x	No change
1	0	0	No change
1	0	1	0 (reset)
1	1	0	1 (set)
1	1	1	$Q'_{current}$

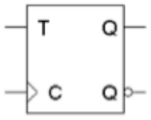
J	K	Q(t+1)	Operation
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	Set
1	1	Q'(t)	Complement

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



T flip-flop

current를 유지하거나 current의 complement만 출력



C	T	Q_{next}
0	x	No change
1	0	No change
1	1	$Q'_{current}$

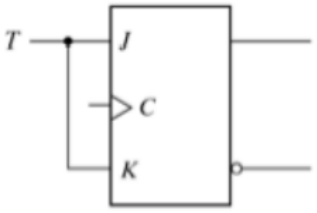
0→1

T	Q(t+1)	Operation
0	Q(t)	No change
1	Q'(t)	Complement

$Q(t+1) = TQ(t) + TQ'(t)$
$= T \oplus Q(t)$

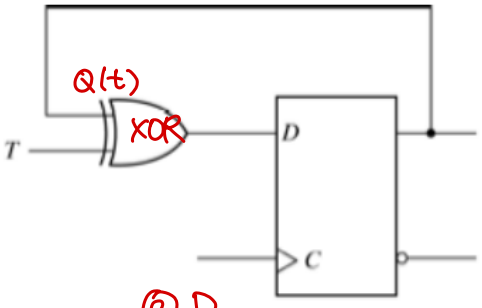
T | Q(t+1)
0 | Q(t)
1 | Q'(t)

J K | Q(t+1)
0 0 | Q(t)
0 1 | 0
1 0 | 1
1 1 | Q'(t)



① JK

(a) From *JK* flip-flop

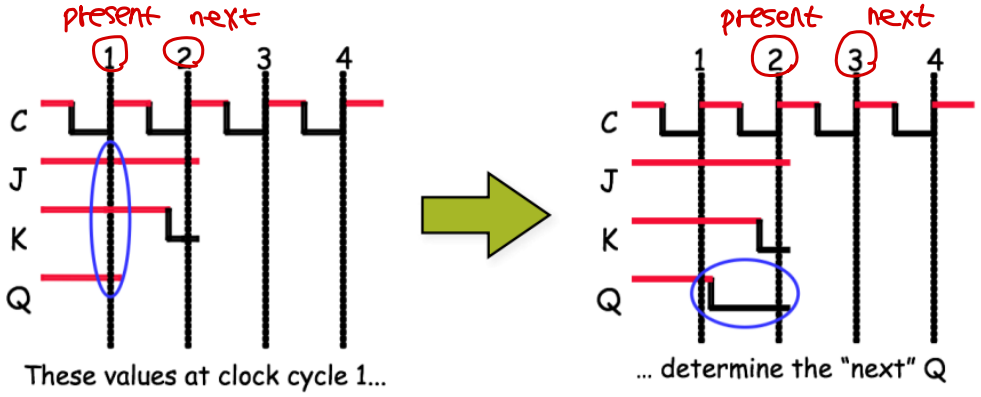


② D

(b) From *D* flip-flop

Timing diagrams

"present", "next" → relative (인접을 현재로 보고 이후에 따라 다름 → 상대적)



⇒ J,K가 어떤 변화가 있더라도 positive edge가 아니면 Q에 반영 X