Ch.6 Registers and Counters - part A

Registers

: 임시 저장소, ff 여러 개를 하나로 (ff 확장)

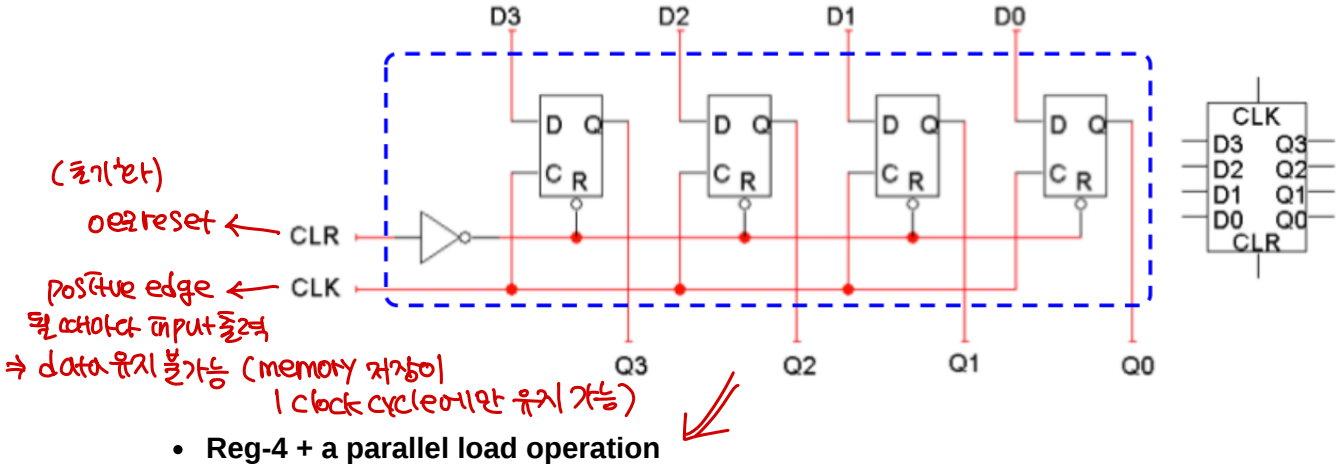
→ ff보다 더 많은 양의 메모리를 저장(대부분의 컴퓨터는 32bit보다 큼, ff는 1bit)

→ processor에서 일시적인 저장소로 사용됨 ⇒ CPU 내부에서 제일 바쁜 memory

→ 메인 메모리보다 빠르기에 복잡한 연산을 빨리 하는데 도와줌

a basic register - Reg-4

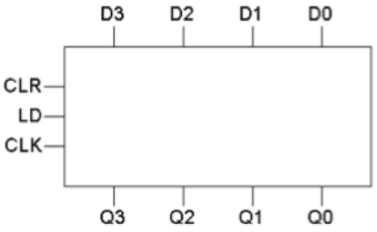
- **Reg-4 (a 4-bit register)** (범용 register)
  - D ff 사용 : ff input equations 없이 데이터 저장 쉬움
  - ff 간의 CLK, CLR signal 공유

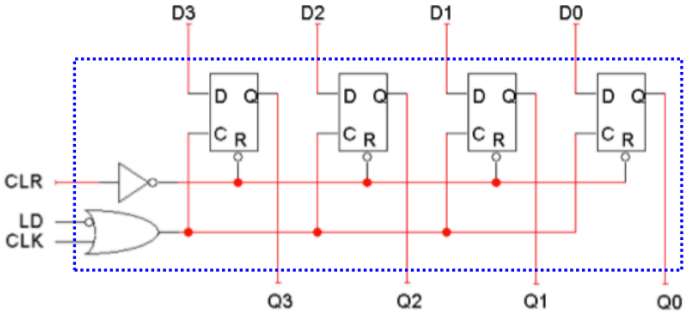


- **Reg-4 + a parallel load operation**
  - input : D3-D0
  - output : Q3-Q0
  - clock cycle마다 input을 output에 copy하는 operation
  - **LD signal**

현재값
next input

LD	Q(t+1)
0	Q(t)
1	D_3-D_0





- LD = 0
  - CLK이 무엇이든 다 1 → data 유지(load x)
- LD = 1
  - CLK이 positive edge일때만 1 → data load

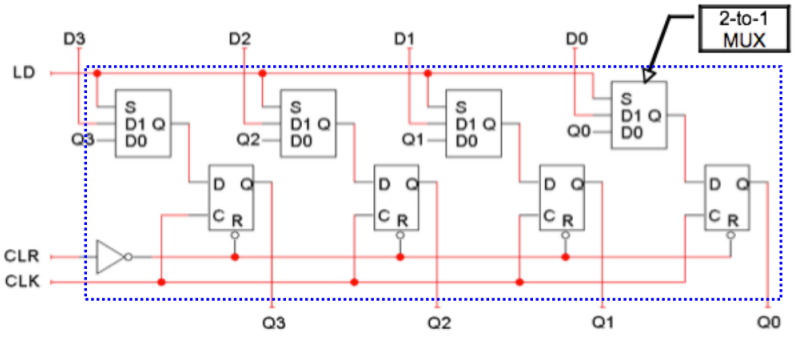
⇒ clock gating is bad !

→ ~~timing problems~~가 발생

- clock 자체는 길지 않지만 연산할 때마다 약간 delay (LD는 1 clock cycle 이내에서만 1 유지)
- 여러 개의 register가 같이 clock 사용하면 약간의 차이가 생김 → clock skew

① reg :
② reg :
차이

- **LD + 2-to-1 MUX**



- LD = 0
  - input : Q3 - Q0 → data 유지(load x) ⇒ MUX가 D_0 선택
- LD = 1

- input : D3 - D0 → data load (new data) ⇒ MUX가 D_1 선택

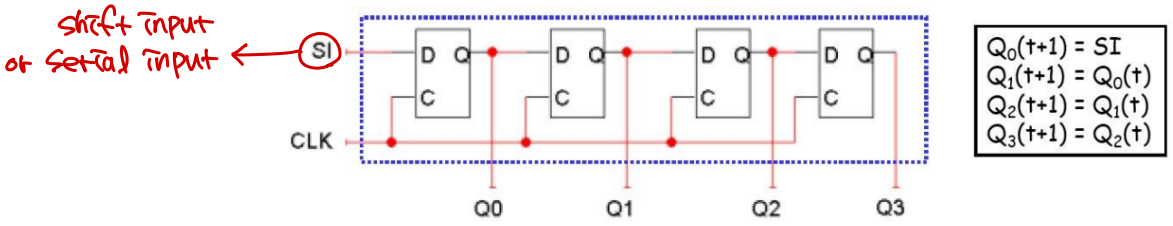
⇒ clock gating X

→ timing problems가 발생 X (좋다 안정적)

Shift register

clock cycle마다 output을 하나씩 'shift'

ex) right로 shift



- **SI input** : shift 할 때 빈 자리에 새로 넣을 input

"into"

SI = 1

Q_0-Q_3 = 0110 → Lost

(clock cycle ↓) will be:

Q_0-Q_3 = 1011

- **Shift direction**

① right shift

LSB를 Lost

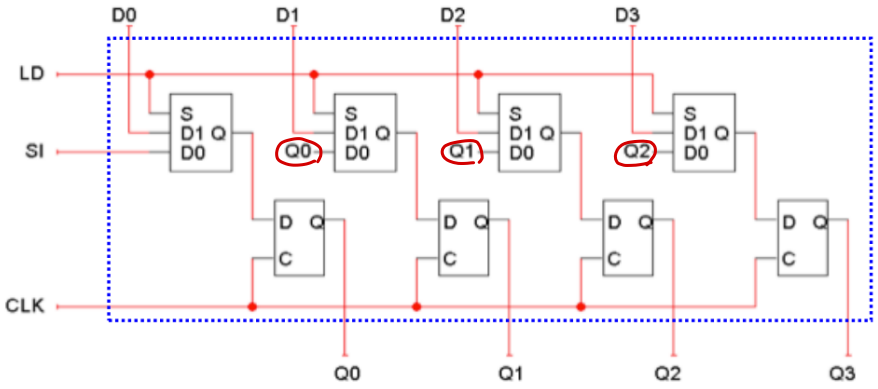
Present Q_0-Q_3	SI	Next Q_0-Q_3
ABCD	X	XABC

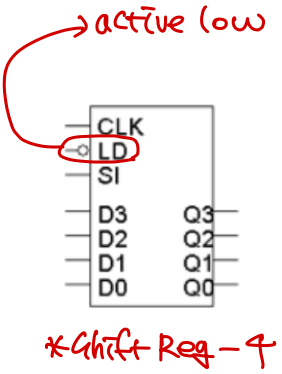
② left shift

MSB를 Lost

Present Q_3-Q_0	SI	Next Q_3-Q_0
DCBA	X	CBAX

- **shift register + parallel load** → regular register





  - LD = 0 ⇒ MUX : P_0
    - input : SIQ0Q1Q2

      → 다음 positive edge에서 shift
  - LD = 1 ⇒ MUX : P_1
    - input : D0 - D3(new value) → 새로운 input

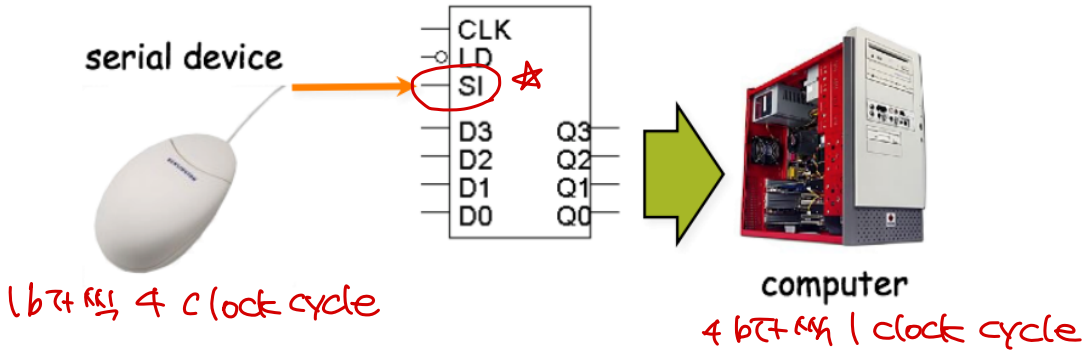
      → 다음 positive edge에서 data load

Serial data transfer

shift register : 'serial data' , 'parallel data' 서로 converting 가능

- computer는 multiple-bit quantities로 일하지만 가끔 serial data로 받는게 필요함
  - ex) keyboard, mouse

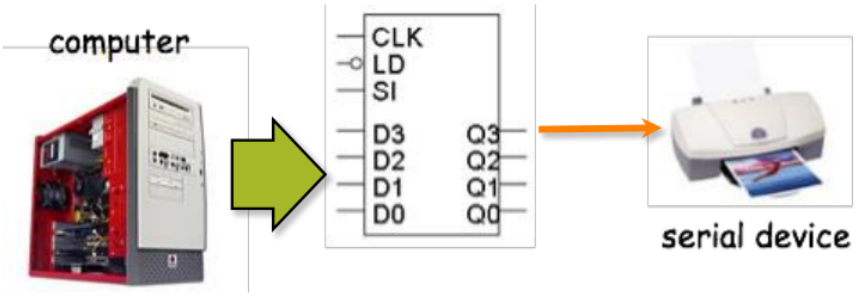
1. receive serial data using a shift register



2. send data serially with a shift register

4bit씩 1 clock cycle

1 bit씩 4 clock cycle



Universal shift register

: 내가 고른 기능이 수행됨

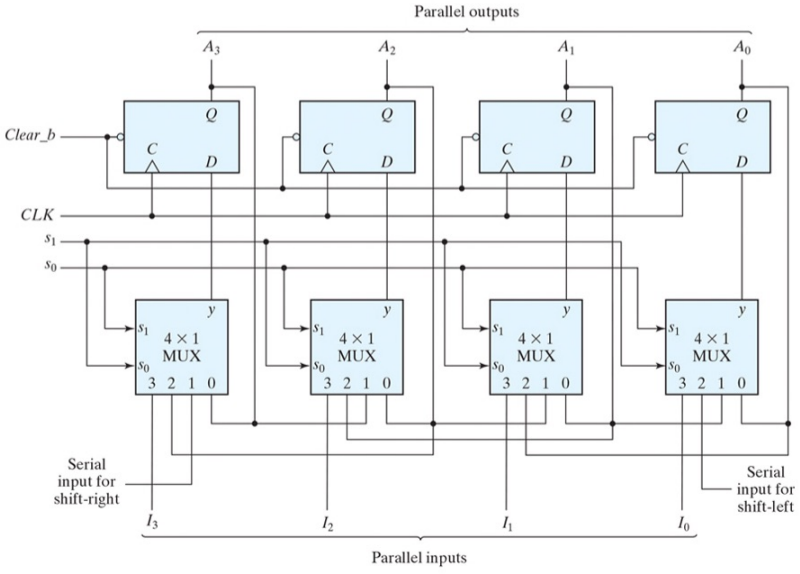


Table 6-3
Function Table for the Register of Fig. 6-7

Mode Control		
s_1	s_0	***Register* Operation**
0	0	No change ①
0	1	Shift right ②
1	0	Shift left ③
1	1	Parallel load ④

Input → clear, CLK, G_1, G_0, $I_3 I_2 I_1 I_0$, G_{IR}, G_{IL}

Output → $A_3 A_2 A_1 A_0$

- clear : register를 0으로 clear
- clock
- shift-right : right enable
- shift-left : left enable
- parallel-load : control to enable
- n parllel input lines

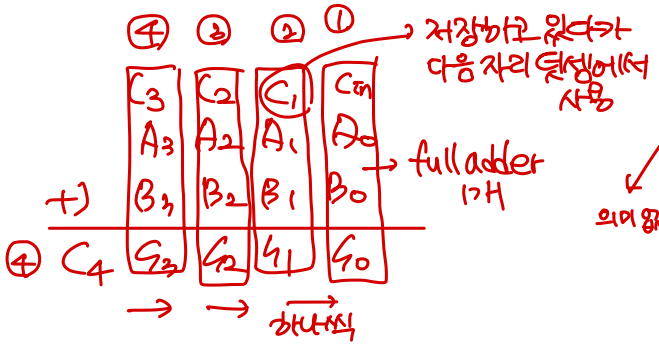
→ a control state : register의 값을 unchanged

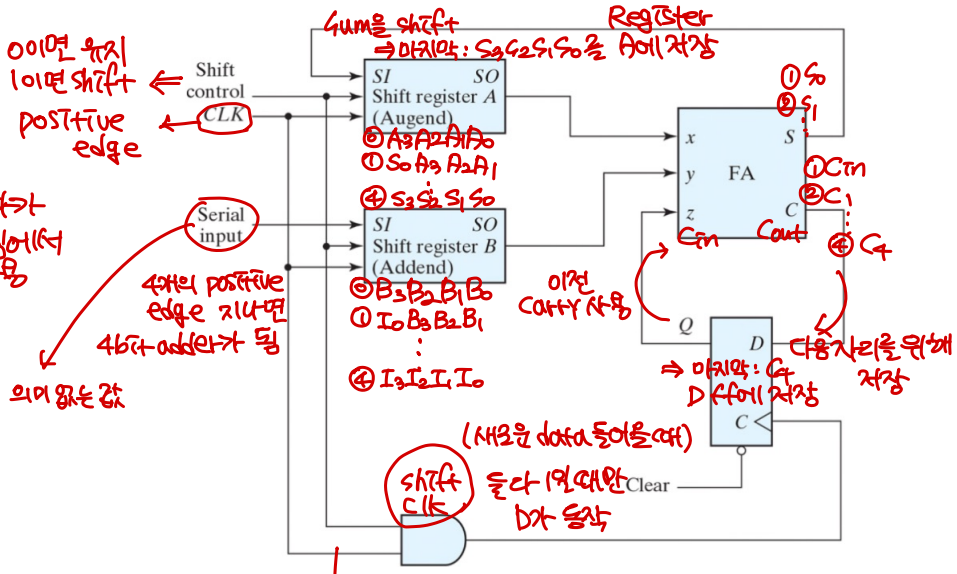
example - serial addition

↳ 4 bit full adder 가능

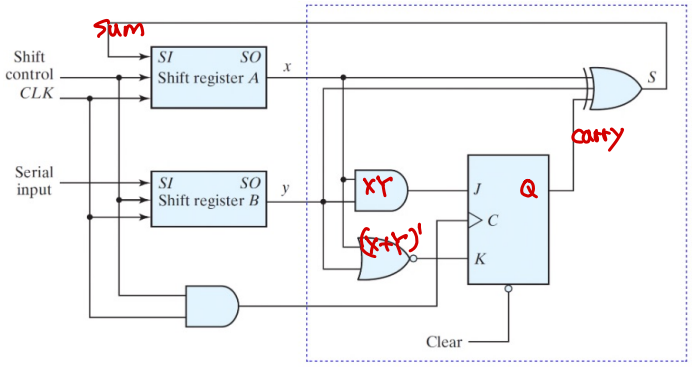
1. using D ff

   a. A + B = S + C





2. using JK ff → full adder 없이



Present State	Inputs		Next State	Output	Flip-Flop Inputs	
Q	x	y	Q	S	J_Q	K_Q
0	0	0	0	0	0	X
0	0	1	0	1	0	X
0	1	0	0	1	0	X
0	1	1	1	0	1	X
1	0	0	0	1	X	1
1	0	1	1	0	X	0
1	1	0	1	0	X	0
1	1	1	1	1	X	0

$$J_Q = xy$$
$$K_Q = x'y' = (x+y)'$$
$$S = x \oplus y \oplus Q$$

+) 8bit adder

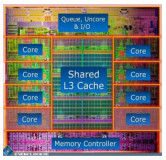
Registers in Modern Hardware

Registers store data in the CPU

- Used to supply values to the ALU.
- Used to store the results.

If we can use registers, why bother with RAM?

Processer	GPR	Mode
Intel Core i7	8	32bit
	16	64bit
AMD Ryzen 7	8	32bit
	16	64bit



Answer: Registers are expensive!

- Registers occupy the most expensive space on a chip – the core.
- L1 and L2 are very fast RAM – but not as fast as registers.