2018년 2학기 컴퓨터공학실험Ⅱ
CSE3016-05반 13주차 예비 보고서

학번: 20171665
이름: 이 선 호

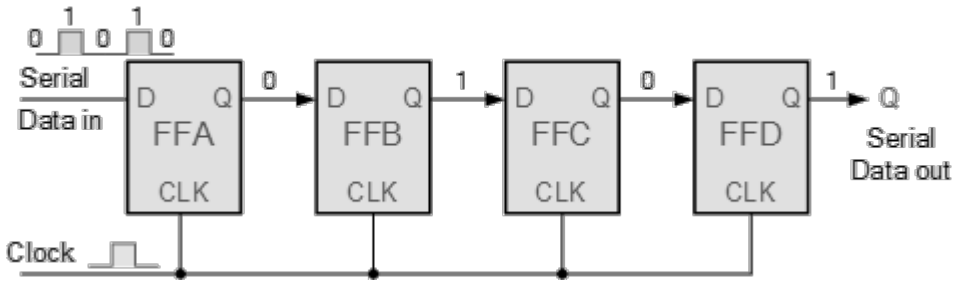
2018. 12. 13

목 차

Ⅰ Shift Register

1. Shift Register

데이터가 개개의 플립플롭에 저장될 수도 있지만, 용량이 큰 데이터를 저장할 때는 주로 레지스터를 사용한다. 레지스터는 플립플롭들을 단순히 모아놓은 것으로, 각 플립플롭은 보통 공통적으로 연결된 Clock을 사용하며 레지스터 이름에 첨자를 사용하여 구분한다. 례를 들어, 하나의 컴퓨터에서 두 개의 32 비트 입력을 가진 덧셈기가 두 개의 레지스터를 이용해 만들어지면 각 레지스터는 32개의 플립플롭으로 구성된다. 이때 시스템을 구성하는 각각의 플립플롭과 게이트를 모두 블록으로 표현하는 것은 거의 불가능하기 때문에 레지스터라는 것을 정의하여 하나의 블록으로 표현한다.



Shift Register의 가장 대표적인 형태는 shift 입력이 들어오거나 각 Clock에 맞춰서 데이터가 한 정소에서 오른쪽의 다른 장소로 이동하도록 플립플롭을 배치한 것이다. 일반적으로 대부분의 Shift 레지스터는 SR 플립플롭을 이용해 구현되지만, JK 플립플롭 또는 D 플립플롭을 사용할 수도 있다. D 플립플롭에서는 위의 그림과 같이 하나의 플립플롭의 출력 Q를 다음 플립플롭의 D 입력으로 연결하면 된다. 위의 회로도에서 왼쪽부터 오른쪽으로 차례대로 플립플롭을 q1, q2, q3, q4라고 가정하면, 입력한 데이터는 q1로 이동하고 플립플롭의 내용은 한 칸씩 오른쪽으로 이동한다. 모든 플립플롭의 초기 상태가 0이라고 가정했을 때 Timing Trace는 아래와 같다.

data	1 0 1 1 1 0 1 1 1 1 0 0 0
q_1	0 1 0 1 1 1 0 1 1 1 1 0 0 0
q_2	0 0 1 0 1 1 1 0 1 1 1 1 0 0 0
q_3	0 0 0 1 0 1 1 1 0 1 1 1 1 0 0 0
q_4	0 0 0 0 1 0 1 1 1 0 1 1 1 1 0 0 0

대부분의 컴퓨터에서는 왼쪽 shift, 오른쪽 shift, 그리고 순환 rotate 명령어가 사용되는데, 이들을 구현하기 위해서는 좌우 shift register를 사용할 수 있다. 좌우 shift

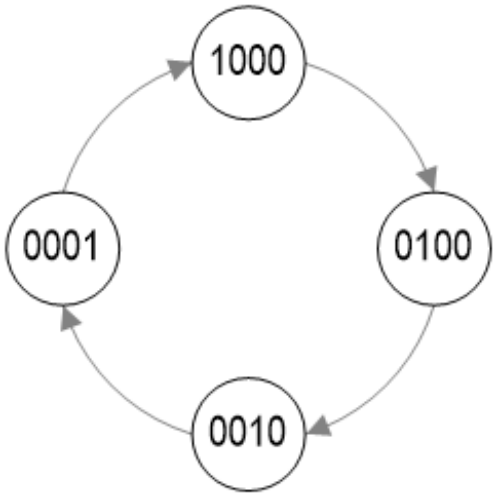
register는 각 비트에 대해 입력이 세 개인 멀티플렉서가 필요한데, 이는 왼쪽에서 온 1 비트를 받을지, 오른쪽에서 온 1 비트를 받을지, 아니면 입력으로 들어온 비트를 받을지 결정해야 되기 때문이다.

단일 모듈 shift register를 구현하는 Verilog 코드를 작성하면 다음과 같다.

```
module shift (Q, x, ck, CLR);
    input x, clock, CLR;
    output [7:0]Q;
    wire [7:0]Q;
    reg [7:0]Q;
always (@ negedge ck) // 매번 Clock이 1에서 0으로 하강할 때 블록 내 구문 실행
    begin
        Q[0] <= Q[1];
        Q[1] <= Q[2];
        Q[2] <= Q[3];
        Q[3] <= Q[4];
        Q[4] <= Q[5];
        Q[5] <= Q[6];
        Q[6] <= Q[7];
        Q[7] <= x;
    end
endmodule
```

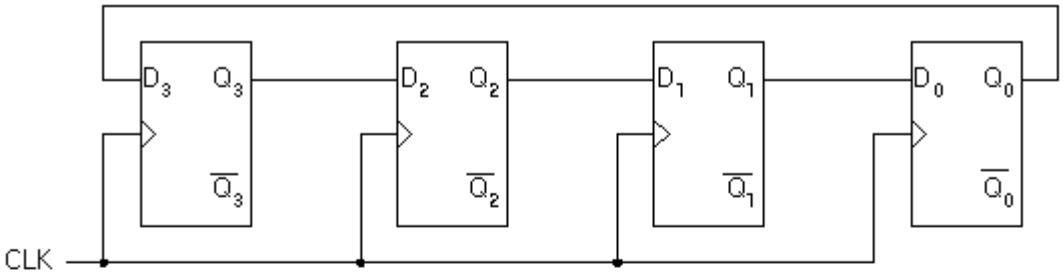
Ⅱ Counter

1. Ring Counter

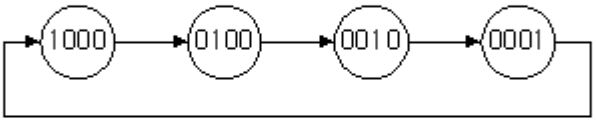


앞선 12주차 예비보고서 기타 이론에서 조사한 바가 있다. Ring Counter는 임의의 시간에 한 개의 플립플롭만 논리 1이 되고 나머지 플립플롭은 논리 0이 되는 Counter이며, Decoder와 같은 효과를 지닌다. 그래서 전체적으로 데이터가 회전하는 Shift Register가 이에 해당하며, 입력된 데이터는 Clock의 pulse마다 한 칸씩 이동하게 된다. State Diagram을 그리면 왼쪽 그림과 같다. 실제 회로도에서는 Ring Counter의 각 D 플립플롭이 자신의 왼쪽에 있는 플

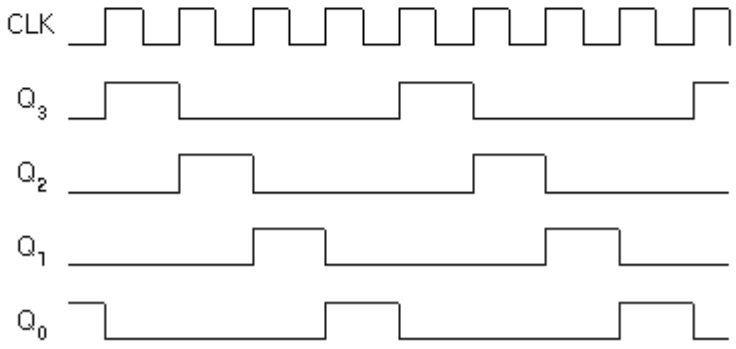
립플롭의 출력을 입력으로 받아들이도록 차례대로 연결되어 있으며, 맨 오른쪽 플립플롭의 출력은 맨 왼쪽 플립플롭의 입력으로 연결되어 있으므로 Clock이 상승 또는 하강할 때마다 Q3 값은 Q2로, Q2 값은 Q1로, Q1 값은 Q0으로, Q0 값은 다시 Q3로 순환된다. 따라서 만약 회로의 초기 상태 값이 (Q3, Q2, Q1, Q0 순으로) 1000 이었다면 Clock이 상승 또는 하강할 때마다 1000, 0100, 0010, 0001, 1000, … 와 같은 순서로 변화할 것이다. 그런데 만약 회로의 초기 상태 값이 0000이거나 1111인 경우에는 clock이 변화하더라도 상태 값에 변화가 없게 된다. 따라서 Ring Counter가 일반적으로 의미가 있으려면 상태 값을 0000이나 1111이 아닌 다른 값으로 초기화한 후에 사용해야 한다.



(a) 회로도



(b) 상태천이도

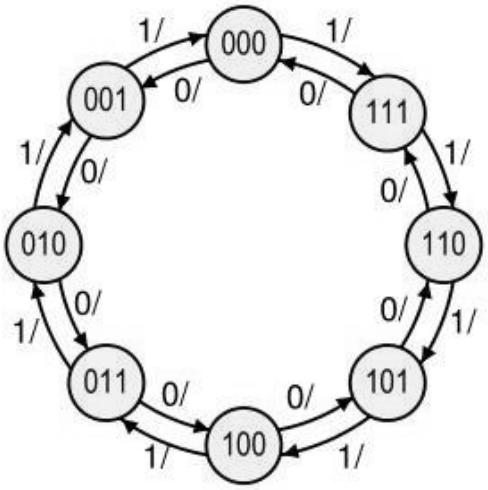


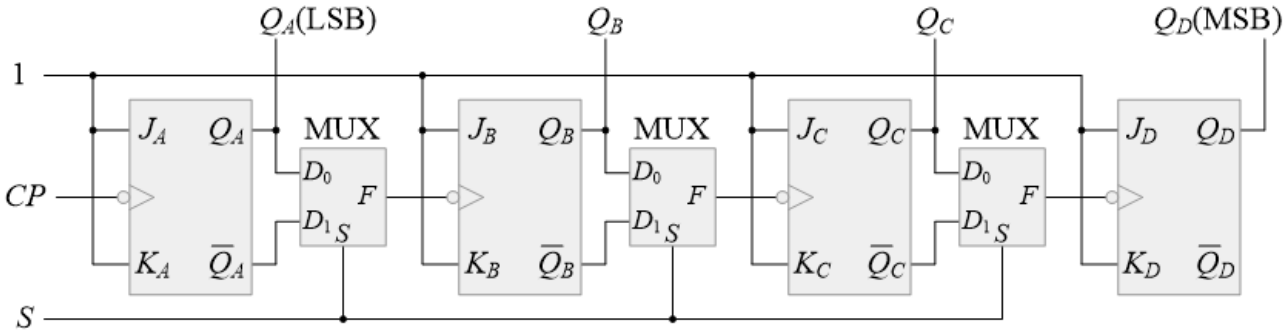
(c) 타이밍도

2. UP DOWN Counter

UP-DOWN(상향-하향) Counter는 어떤 특정 입력이 0으로 들어오면 현재 상태에서

다음 상태로 이동하고, 입력이 1로 들어오면 현재 상태에서 이전 상태로 이동하는 Counter이다. 주로 MUX를 사용하여 선택 입력 값에 따라 순차적으로 연결된 플립플롭의 Clock 입력으로 이전 플립플롭에서 출력된 Q와 Q' 중 하나의 출력을 연결시켜서 구현한다. JK 플립플롭과 MUX를 사용하여 구현한 상향/하향 Counter의 상태도와 논리 회로도를 그리면 아래와 같다.



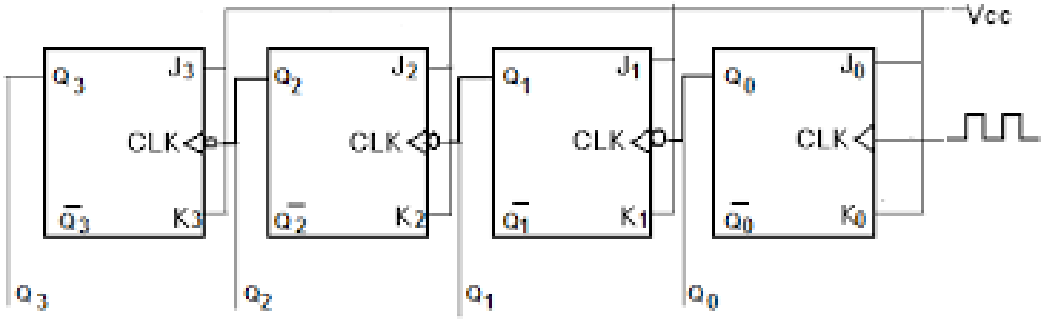


위의 3-Bit 상향/하향 Counter의 회로도를 참고하면 총 3개의 MUX가 각 플립플롭의 출력과 연결되어 있다는 사실을 알 수 있다. 만약 S를 0으로 설정하면 MUX의 입력인 D0과 출력 F가 연결이 되어서 상향 Counter로 작동한다. S를 1로 설정하면 MUX의 입력인 D1과 출력 F가 연결이 되어 하향 Counter로 작동한다.

3. Ripple Counter

연속적인 Clock 입력에 따라 상태가 고정된 sequence에 따른 움직임을 보이는 장치가 동기식 Counter이면, 비동기식 Counter는 동기식 Counter에서 사용된 것과 동일하게 Clock에 의해 동작하는 플립플롭을 이용하지만, 실제로 모든 플립플롭에 관해 Clock을

사용하는 것이 아니라 각 플립플롭이 이전 플립플롭의 천이의 의해 활성화(Trigger)되는 장치를 말한다. 그래서 이를 Ripple Counter라고도 부른다. Ripple Counter의 예를 회로도로 그리면 아래와 같다.



예를 들어, 첫 번째 플립플롭의 입력에만 Clock Pulse가 입력이 되고, 다른 플립플롭은 각 플립플롭의 출력을 다음 플립플롭의 Clock 입력으로 사용하는 것이 비동기식 Counter이다. 앞에서 설명한 Decade Counter에서 낮은 자리의 Counter가 9에서 0으로 되면서 reset될 때, 그 신호를 다음 자리의 Counter에 전하여 그 자리의 Count를 1만큼 증가시키는 것도 Ripple Counter의의 예시가 된다.

Ripple Counter는 동기식 Counter에 비해 회로가 간단하지만 전달 지연이 커진다는 단점이 있다.

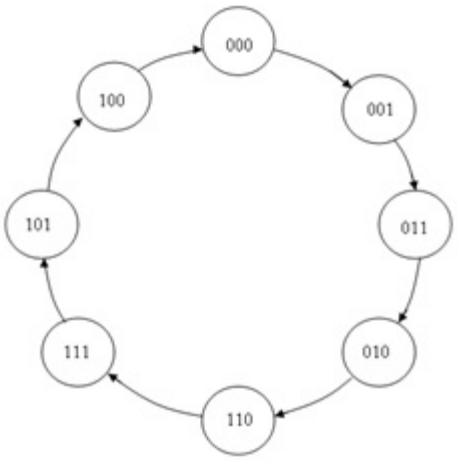
Ⅲ 기타 이론

1. Gray Code와 Gray Code Counter

그레이 코드는 연속적인 비트 코드들 간에 하나의 비트만 변화하여 새로운 코드가 되도록 하는 것이다. 연산에는 쓰이지 않고 주로 데이터 전송, 입출력 장치, 아날로그와 디지털 간 변환과 주변 장치에서 많이 쓰인다. 장점은 현재 비트에서 1개의 비트만 변화하여 다음 상태의 수를 표현할 수 있기 때문에 변환해야 할 비트의 개수가 매번 적다는 점에서 효율적이다. 4비트 그레이 코드를 예를 들면 다음과 같다.

10진 코드	2진 코드	그레이 코드
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

그레이 코드 Counter를 state diagram과 state table, 그리고 회로도를 나타내면 아래와 같다.



Present State			Next State		
Q2	Q1	Q0	Q2	Q1	Q0
0	0	0	0	0	1
0	0	1	0	1	1
0	1	1	0	1	0
0	1	0	1	1	0
1	1	0	1	1	1
1	1	1	1	0	1
1	0	1	1	0	0
1	0	0	0	0	0