13주차 예비보고서

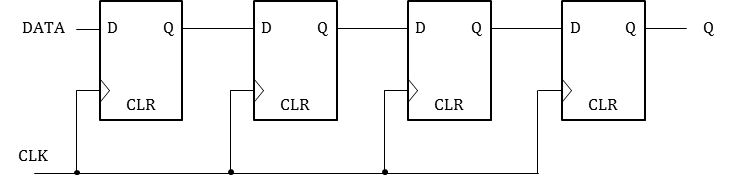
전공: 기계공학과 학년: 3학년 학번: 20191820 이름: 김형준

**1.**

시프트 레지스터(shift register)는 데이터를 저장하거나 데이터를 옆으로 이동할 때 사용되는 회로이다. 단방향 또는 양방향으로 매 클록마다 한 단씩 이진 정보를 이동 시킬 수 있는 레지스터로, 일련의 플립플롭들을 직렬로 연결한 구조로 되어있다.

시프트 레지스터는 다방면으로 사용될 수 있는데, 메모리 구현, 데이터의 직렬-병렬 변환(UART 등), 시프트 카운터(링 카운터 등) 등에 사용된다.

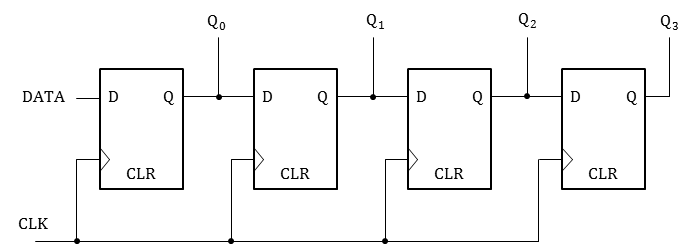
시프트 레지스터는 입출력의 구조(직렬/병렬)에 따라 4가지 종류로 분류된다.



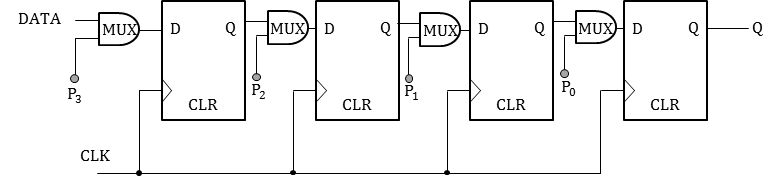
SISO (Serial-in to Serial-out) 시프트 레지스터는 입력데이터를 그대로 저장하고 있다가 클럭(CLK)에 따라 시간 지연(delay)을 거쳐 출력되는 회로이다.

클럭이 발생할 때마다 순차적으로 데이터를 이동시키고, 클럭이 발생하지 않은 경우에는 클럭이 발생할 때까지 데이터를 그대로 유지시키는 기능을 한다.

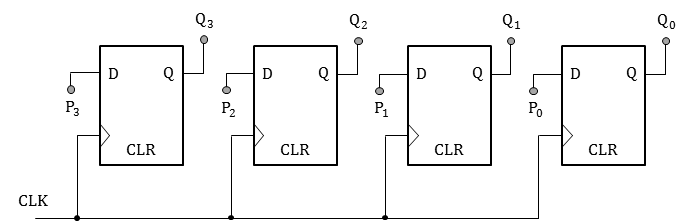
SISO 시프트 레지스터는 중간에 데이터를 출력하지 않고 가장 마지막에 연결된 플립플롭의 출력만을 보기 때문에 단순한 구조로 되어있으며, 데이터 출력시 플립플롭에 저장된 데이터가 변경된다는 단점이 있다.



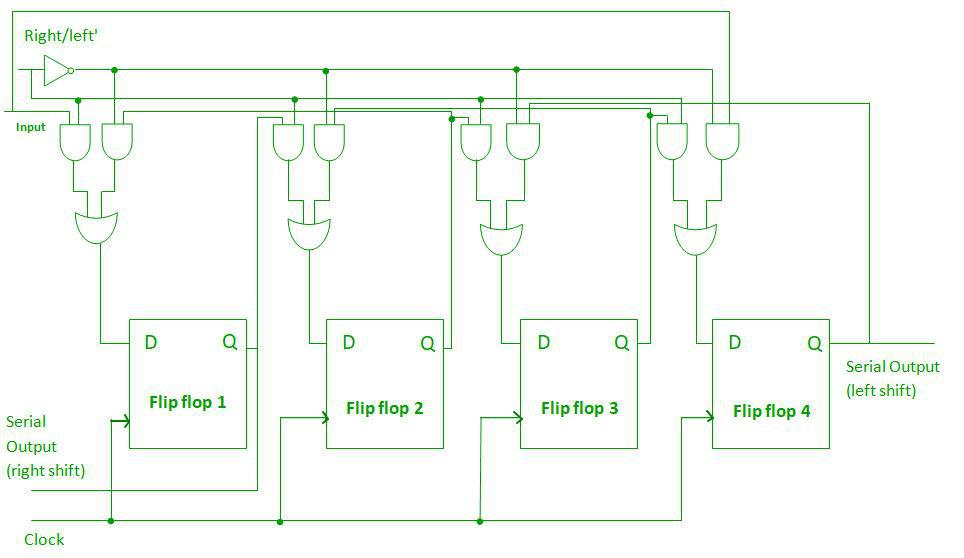
SIPO (Serial-in to Parallel-out) 시프트 레지스터는 단일의 데이터가 시프트 레지스터를 거쳐 다수의 출력으로 나타나는 구조이다. 입력 데이터가 들어오면 다음 클럭이 발생할때까지 대기하다가 클럭이 발생하면 (첫번째 플립플롭의 출력)로 출력되고, 그 다음 클럭이 발생하면 로 출력되던 데이터는 로 출력되고, 새로운 입력이 로 출력되는 방식이다. 즉, 입력된 데이터는 연속되는 클럭을 통해서 LSB부터 MSB까지 1비트씩 옮겨지지만(Shift), 출력은 현재 플립플롭들에 저장된 값이 모두 동시에 병렬적으로 출력된다.



PISO (Parallel-in to Serial-out) 시프트 레지스터는 SIPO 시프트 레지스터와 반대로 다수의 입력데이터를 한 개의 출력으로 내보내는 회로이다. 다수의 데이터들을 병렬로 동시에 받아들이고, 클럭이 발생할 때마다 입력된 데이터를 순서대로 하나씩 출력하는 방식이다.

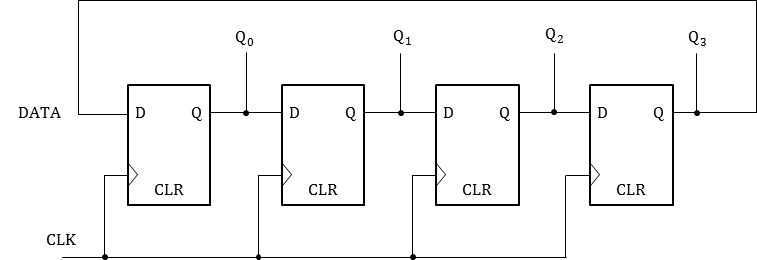


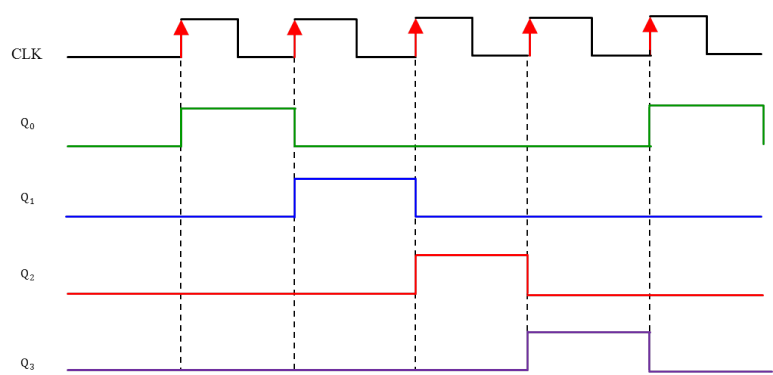
PIPO (Parallel-in to Parallel-out) 시프트 레지스터는 n개의 플립플롭을 이용해 n비트 데이터가 병렬로 동시에 입출력되는 구조이다. 클럭이 발생하면 입력데이터가 병렬로 동시에 각각의 플립플롭에 입력되어 저장되고, 각각의 플립플롭에 저장된 값들은 플립플롭에서 유지되다가 클럭이 발생하면 시간 지연을 거쳐 병렬로 동시에 출력한다.



앞에서 설명한 입출력 구조에 따른 4가지 종류의 시프트 레지스터 외에도 여러가지 종류의 시프트 레지스터가 존재하는데, 위 그림은 양방향으로 데이터를 입력받거나 옮길 수 있는 Bidirectional Shift register(양방향 시프트 레지스터)이다. 양방향 시프트 레지스터는 왼쪽 위의 Mode bit(Right/left)에 따라 데이터를 이동시킬 방향을 결정한다. Mode bit가 1인 경우 데이터를 오른쪽으로 옮기며, 0인 경우 왼쪽으로 옮긴다.

**2.**





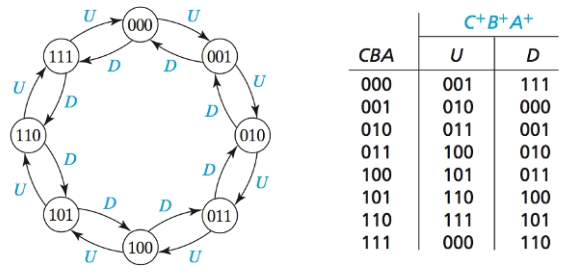
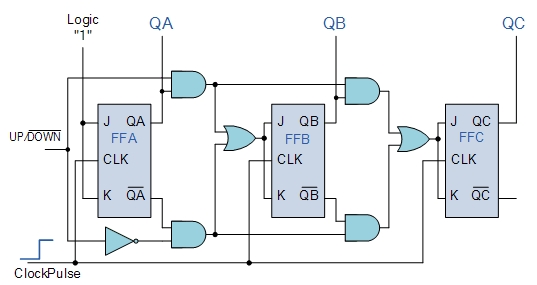
링 카운터(ring counter)는 전체적으로 데이터가 회전하는 시프트 레지스터를 말하며, 마지막 플립플롭의 출력이 첫 번째 플립플롭의 입력에 연결된 구조로 되어있다. 입력된 데이터는 클럭이 발생할 때 마다 한 칸씩 이동하여 순환하게 된다.

링 카운터는 임의의 시간에서 한 개의 플립플롭만 출력이 1이 되고, 나머지 플립플롭들의 출력은 0이 되는데, 위 그림의 타이밍도처럼 클럭이 발생할 때마다 데이터가 전체적으로 한 칸씩 회전하므로 출력1의 위치가 한 칸씩 이동하는 것을 확인할 수 있다.

클럭의 펄스마다 이동하는 방식을 응용하여 링카운터는 직렬통신 회로의 기초가 된다.

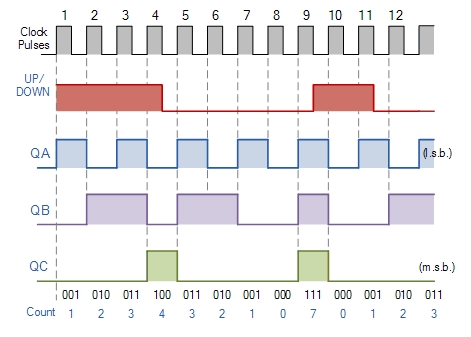
**3.**

Up Down Counter는 Counter의 값이 1씩 증가하는 Up counter의 기능과 counter의 값이 1씩 감소하는 Down counter의 기능을 입력값에 따라 선택할 수 있는 동기식 counter이다.



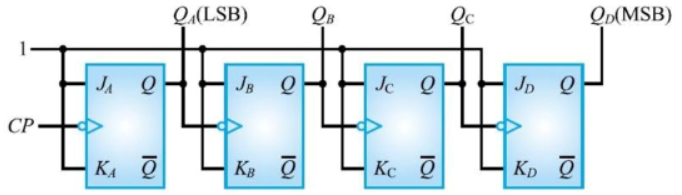
위는 JK 플립플롭을 사용해 구현한 3bit Up Down Counter이다. 맨 왼쪽의 입력값 Up/Down에 따라 Up Counter의 기능을 할지, Down Counter의 기능을 할지가 결정되는데, Up/Down의 값이 1인 경우 3bit Up Counter의 기능을 해 클럭이 발생할 때 마다 현재 상태에서 다음 상태로 이동하고(state diagram에서 시계방향), Up/Down의 값이 0인 경우 3bit Down Counter의 기능을 해 클럭이 발생할 때마다 현재 상태에서 이전 상태로 이동한다(state diagram에서 반시계방향).

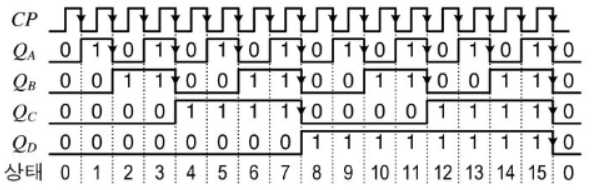
Up/Down 입력값의 변화에 따른 각 Q값의 양상을 그린 타이밍도의 예시는 다음과 같다.



위 타이밍도를 보면 Up/Down의 값이 1일 때 (클럭 펄스가 1~4) Q의 값이 1부터 4까지 증가하다가 Up/Down의 값이 0으로 변해(클럭 펄스가 5~9) Q의 값이 4부터 0으로 감소하고(clk=5~8), 0에서 7로(clk=9) 변하는 것을 확인할 수 있다.

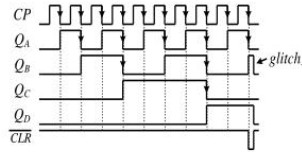
**4.**





Ripple Counter는 공통 클럭 펄스에 모든 플립플롭이 동기화되지 않고 주로 첫번째 플립플롭만 클럭 펄스에 동기화되는 비동기식 카운터(Asynchronous counter)를 의미한다. Ripple counter는 보통 JK 플립플롭 또는 T 플립플롭으로 설계되고, 첫번째 플립플롭을 제외한 모든 플립플롭에서 각각의 플립플롭의 클럭 입력으로 이전 플립플롭의 출력을 입력한다. Ripple Counter는 동기식 카운터와 다르게 모든 플립플롭을 동기화 시킬 필요가 없어 동작 및 논리회로 구성이 단순하고 구현이 용이하지만, 각 플립플롭을 통과할 때 마다 지연시간이 누적되기 때문에 고속동작에는 부적합하다는 단점이 있다. 위 사진은 JK 플립플롭으로 구현한 4bit Ripple Up Counter의 회로도이다. 일반적으로 모든 플립플롭의 JK에는 1을 입력한다.

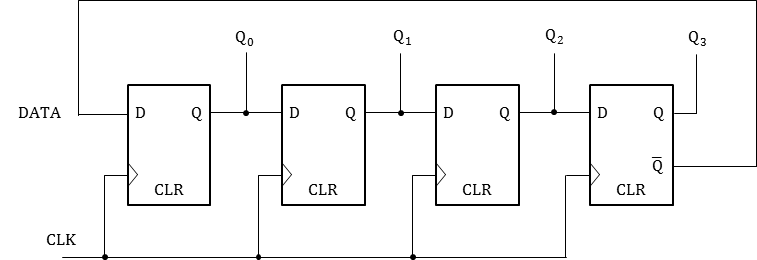
Ripple Counter의 각 플립플롭은 동시에 출력이 발생하는 것이 아니라 이전의 플립플롭의 출력에 의해 다음 플립플롭이 동작하기 때문에 (첫번째 플립플롭만 클럭 펄스에 의해 동작) 전달 지연시간이 각 플립플롭을 통과할 때 마다 점점 누적되는데, 아래 사진과 같이 , , , 4개의 플립플롭의 출력이 동시에 변환되는 것이 아니기 때문에 아주 짧은 시간 동안 중간 과정의 다른 출력값들이 생성 될 수 있다.



이러한 글리치 현상은 정밀한 시간 단위로 작동하는 회로에는 영향을 줄 수 있으므로 Ripple Counter에서는 글리치가 발생할 기간을 예측하여 delay를 줌으로써 글리치를 방지해야 한다.

**5.**

- 존슨 카운터(Johnson Counter)



존슨 카운터(Johnson Counter)는 마지막 플립플롭의 보수 출력()이 첫 번째 플립플롭의 입력에 연결된 구조로 되어있다는 점만 제외하면 Ring counter와 동일한 구조를 갖는다.

존슨 카운터에서도 클럭이 발생할 때 마다 플립플롭에 저장된 데이터가 한 칸씩 이동하여 순환하게 되는데, ring counter와 다르게 N개의 플립플롭으로 구성된 존슨 카운터는 2N개의 서로 다른 상태(state)를 출력한다.

존슨 카운터는 대표적으로 Decade Counter, 8 Counter 등에 사용된다.

아래의 사진은 4bit Johnson Counter의 타이밍도와 state table이다.

