컴퓨터공학실험2 13주차 예비 보고서

전공: 컴퓨터공학 학년: 2학년 학번: 20201635 이름: 전찬

**0. 목차**

1. Shift register에 대해 조사한다.

2. Ring Counter에 대해 조사한다.

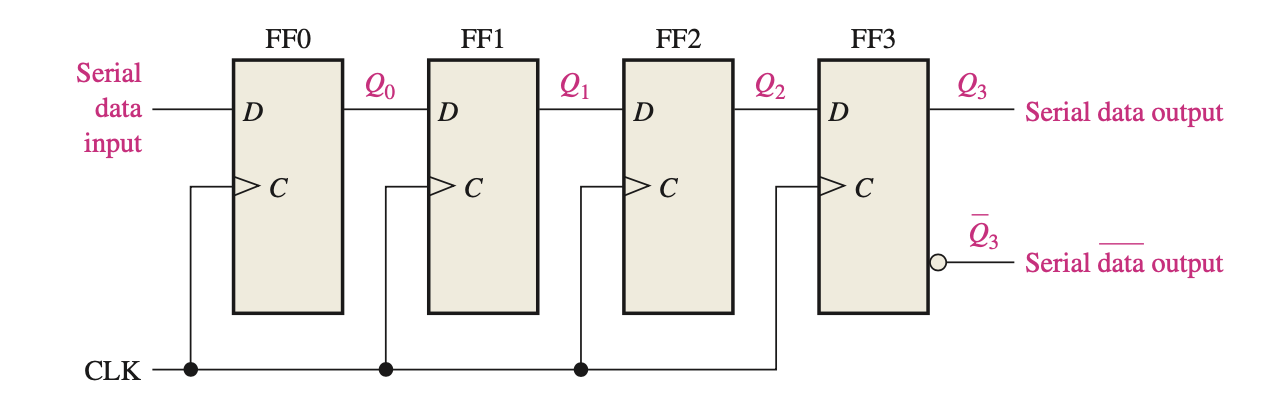
3. UP DOWN Counter에 대해 조사한다.

4. Ripple Counter에 대해 조사한다.

5. 기타 이론

**1. Shift register**

이전 실습에서 D flip-flop을 활용해 event(clock pulse)를 count할 수 있는 counter을 구현했다. 이번에는 D flip-flop의 또 다른 응용 방법인 Shift register에 대해 알아보자. 우선 기본적으로 Shift register의 형태는 아래와 같다.

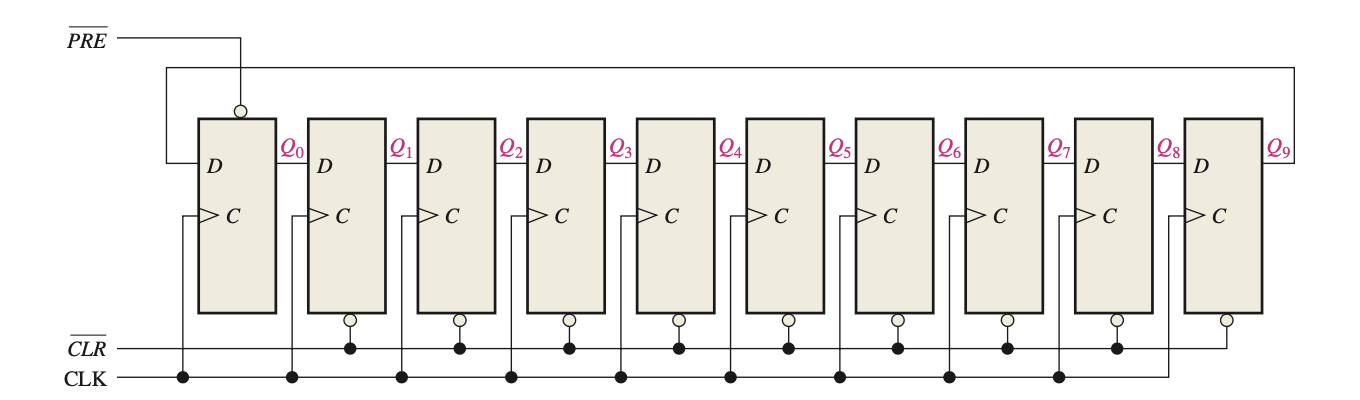


<shift register의 구조>1

위 형태를 보면, 4개의 D flip-flop 연속적으로 연결해, shift register을 구현한 것을 파악할수 있다. 여러 개의 input이 serial data input 방향으로 들어오며, Q3을 통해서 들어온 input과 동일한 형태가 나가게 된다. 만약 data input = 1011 이 들어온다면 Q0Q1Q2Q3은 순서대로 아래와 같을 것이다. 1000 -> 1100 -> 0110 -> 1011 -> 0101 -> ,,, -> 0000 따라서 SISO(serial in serial out) shift register은 input을 bit 단위로 delay 시킬 수 있는 형태로 활용되기도 한다.

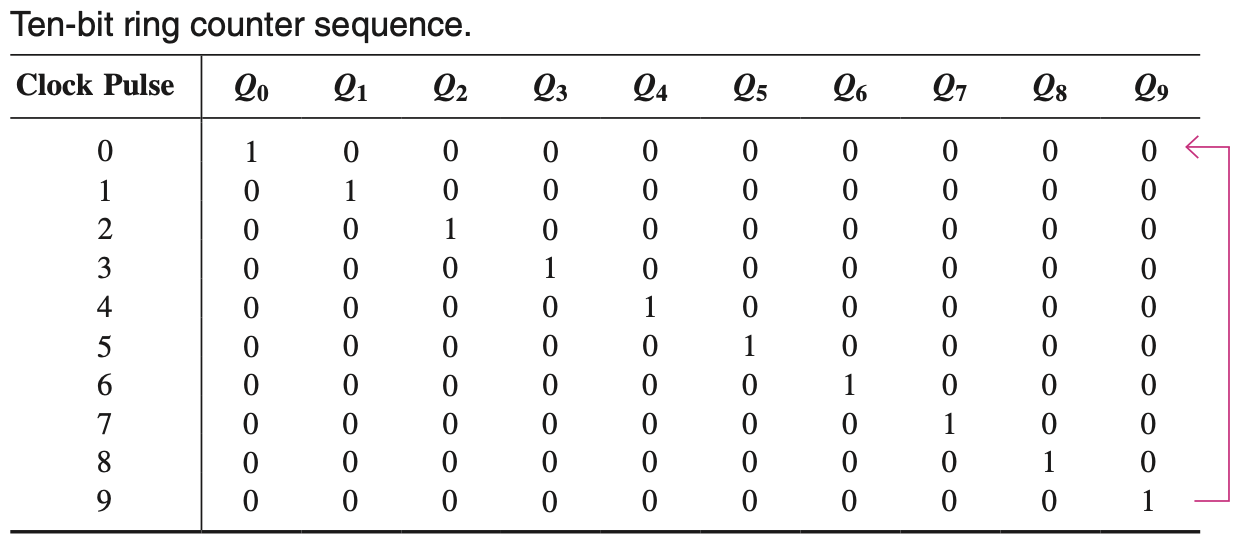
**2. Ring counter**

위 형태처럼 D flip-flop을 연속적으로 연결해 shift register을 구현하면, shift register은 clock pulse마다 D flip-flop을 한 칸씩 움직이는 형태라는 점을 쉽게 파악할 수 있다. 이를 다시 활용해서 counter을 구현할 수 있는데, 바로 Ring counter 이다. ring counter 또한 이전 counter들과 동일하게 clock pulse를 count하는 형태이다. 하지만 이전의 counter들과 다른 점이 존재한다. 이전까지는 D flip-flop의 toggle을 활용하며, clock pulse의 count는 binary, BCD, 2421 code 형태로 저장되었지만, ring counter은 toggle이 아닌 shift register 형태를 활용해 decimal 형태(n까지 count 할 때 n개의 output line을 갖는 형태)로 구현된다. 이러한 ring counter의 형태는 아래와 같다.



<Ring Counter의 형태>2

위 형태를 보면 count를 수행하기 전에, PRE를 통해서 Q0를 먼저 set 해 놓으며, 이후에 clock pulse(rising edge)가 발생했을 때, Q0 -> Q1 -> Q2 -> …. 형태로 shift를 수행하는 회로라는 것을 파악할 수 있다. 이를 통해서 clock pulse가 1번 발생하면 Q1 = 1 이며 나머지는 0, 5번 발생하면 Q5 = 1 이며 나머지는 0 이 되는 형태임을 파악할 수 있다. clock pulse에 따른 각 flip-flop의 output은 아래와 같다.



<clock pulse에 따른 ring counter flip-flop의 각 output>3

**3. UP DOWN Counter**

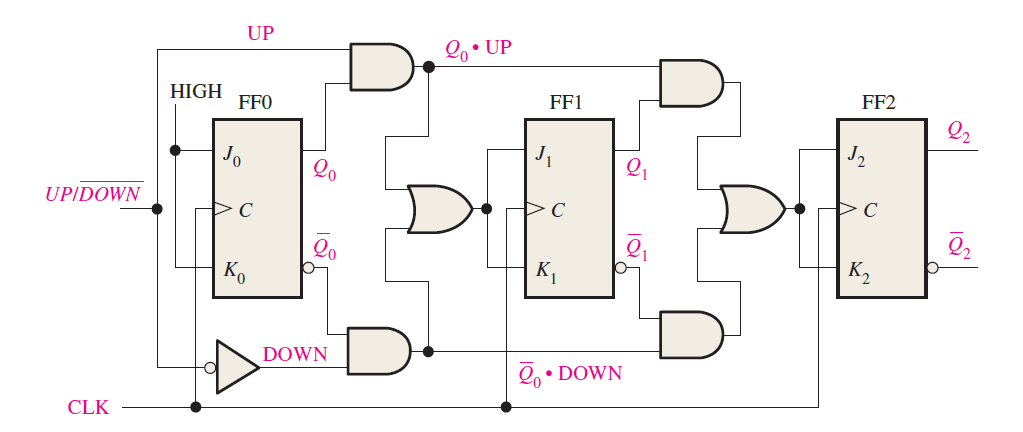
지금까지 알아본 counter은 모두 한쪽 방향(UP, +1이 되는 형태) 로만 count를 수행할 수 있었다. 그 예로 2-bit counter은 00 -> 01 -> 10 -> 11 형태로 계속해서 증가하는 방향으로만 count를 수행했다. 하지만 실제로 양쪽 방향으로의 count가 필요한 여러 경우가 존재한다. 예를 들어 들어오는 input에 대해서 positive = UP count(+1), negative = DOWN count(-1)를 수행해서 positive input과 negative input의 차이가 어떻게 되는지 등의 경우가 있다. 이러한 경우에는 UP DOWN을 둘 다 수행할 수 있는 counter을 구현해야 하며, 이 형태가 UP DOWN counter 이다.

UP DOWN counter의 기본적인 구조는 아래와 같다.

(1) UP/ 에 대해서 UP = 1 이면, clock pulse = 1일 때 UP count를 수행한다.

(2) UP/ 에 대해서 UP = 0 이면, clock pulse = 1일 때 DOWN count를 수행한다.

이에 대한 형태는 아래와 같다.



<3-bit UP DOWN counter의 형태>4

우선 모든 JK flip-flop에서 J = K 형태임을 파악할 수 있다. 따라서 UP DOWN counter을 구현하기 위해서는 JK flip-flop의 not change / toggle 두 가지 형태를 활용한다는 것을 알 수 있다. 우선 첫 번째로 UP count를 수행할 때는, 아래와 같은 구조를 따르게 된다.

(1) Q0는 항상 toggle 된다.(FF0)

(2) 만약 Q0 = 1(HIGH)이며, UP = 1인 경우 JK flip-flop(FF1)의 toggle을 수행한다.

(3) 만약 Q0 = 1, Q1 = 1, UP = 1 인 경우, JK flip-flop(FF2)의 toggle을 수행한다.

또한 DOWN count일 때는 아래와 같은 구조를 따른다.

(1) Q0는 항상 toggle 된다.(FF0)

(2) 만약 Q0 = 0(LOW)이며, = 0인 경우 JK flip-flop(FF1)의 toggle을 수행한다.

(3) 만약 Q0 = 0, Q1 = 0, = 0 인 경우, JK flip-flop(FF2)의 toggle을 수행한다.

이를 통해서 양 방향으로 count를 수행하는 회로를 구현할 수 있으며, 더 나아가 4, 5, …. , n bit UP DOWN counter을 구현하는 것도 계속 동일한 형태로 구현해낼 수 있다.

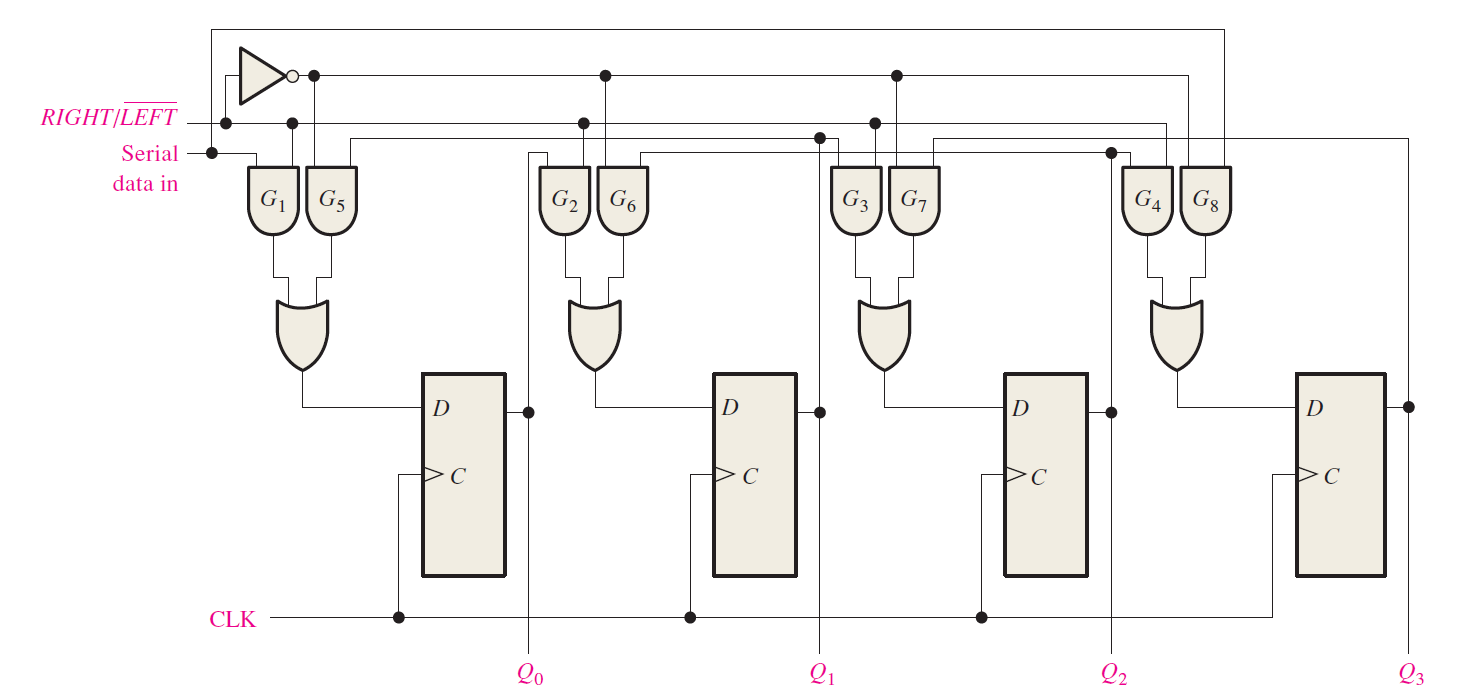
**4. Ripple Counter**

이전 실험에 full adder을 연속적으로 연결해 여러 bit단위의 addition을 수행하는 회로인 4-bit parallel adder을 구현한 적이 있다. 또한 이때 4-bit parallel adder의 문제점으로, 들어오는 input에 대해서 각 full adder을 거치며 전파 지연이 발생한다는 점이 존재했다. 따라서 이를 asynchronous / ripple adder 라고 하기도 했다. 이처럼 counter 또한 clock pulse와 그 뒤의 flip-flop들이 동일한 시간에 수행되지 않을 때(동기화된 형태가 아닐 때) 이를 asynchronous / ripple counter 이라 할 수 있다. 또한 이는 12주차 실습에서 구현한 2-bit counter, decade counter 와 같은 형태임을 간단하게 파악할 수 있다.

추가적으로 synchronous 형태의 counter와 차이점에는 무엇이 존재하는지 간단하게 알아보자. synchronous counter은 clock pulse마다 모든 flip-flop이 동시에 작동해야 되기 때문에, flip-flop의 enable input으로 clock pulse를 받게 된다. 하지만 ripple 형태는 enable에 이전 flip-flop에서 구현된 output(D flip-flop으로 구현한 counter의 경우에는 E = Q\_com)이 들어온다는 점을 차이점이라고 할 수 있다.

**5. 기타 이론**

위에서 가장 기본적인 shift register(SISO form)을 알아보았다. 이는 계속해서 오른쪽으로 shifting이 되는 register이라 할 수 있는데, UP/DOWN counter 처럼 shift 또한 양 방향으로 수행할 수 있는 회로가 존재한다. 이를 bidirectional shift register이라 하며 회로의 형태는 아래와 같다.



<bidirectional shift register의 형태>5

회로의 형태를 살펴봤을 때, 위에서 설명한 UP/DOWN counter과 거의 유사한 형태라는 점을 파악할 수 있다. 약간의 차이점이 존재한다면, UP/DOWN counter은 UP count, DOWN count 두 count에서 모두 오른쪽으로만 가는 형태(FF0은 오른쪽 FF1, FF2, … 의 output이 필요하지 않은 형태)을 보여주었는데, bidirectional shift register 같은 경우에는, 왼쪽으로의 shift 형태를 구현하기 위해서 오른쪽 D flip-flop의 output이 다시 왼쪽 D flip-flop의 input로 사용될 수 있다는 차이점이 존재하기도 한다.

**6. 출처**

1. Thomas L. Floyd, Person, Digital Fundamentals Eleventh edition(global edition), 452p

2. Thomas L. Floyd, Person, Digital Fundamentals Eleventh edition(global edition), 468p

3. Thomas L. Floyd, Person, Digital Fundamentals Eleventh edition(global edition), 469p

4. Thomas L. Floyd, Person, Digital Fundamentals Eleventh edition(global edition), 517p

5. Thomas L. Floyd, Person, Digital Fundamentals Eleventh edition(global edition), 463p