# 2018년 2학기 컴퓨터공학실험Ⅱ
# CSE3016-05반 11주차 예비 보고서

학번: 20171665
이름: 이 선 호

2018. 11. 30

# 목 차

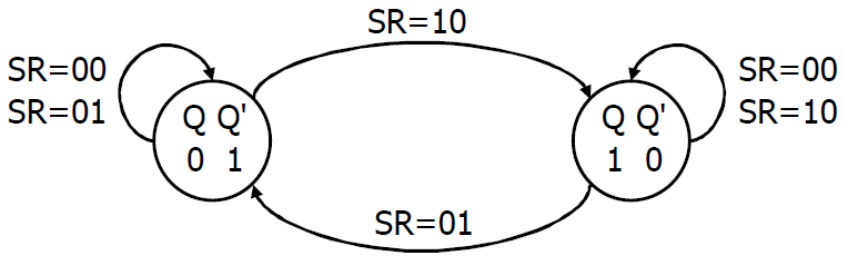
# Ⅰ Flip-Flop과 Latch

## 1. RS 플립플롭

플립플롭(Flip-Flop)은 0 또는 1을 저장할 수 있고 clock이 있는 binary 저장 장치이며, 정상적인 동작 하에서 플립플롭에 저장되는 값은 오직 clock의 값이 바뀌는 시점에서 변경된다. Clock 값이 변경된 이후에 플립플롭에 어떠한 값이 저장되는지는 플립플롭에 들어가는 입력 데이터와 clock 값 변경 이전의 플립플롭에 어떠한 값이 저장되어 있었는가에 따라 결정된다.

플립플롭은 일반적으로 두 개의 출력을 가지는데, 하나는 현재 플립플롭에 저장된 값이고, 다른 하나는 그 값의 보수이다.

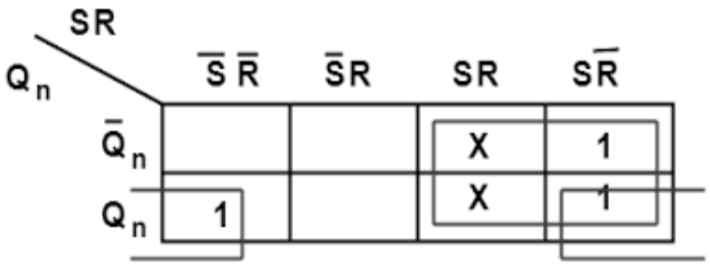
RS 플립플롭은 S와 R이라는 두 개의 입력을 가지는데, S는 플립플롭에 저장된 값을 1로 set한다는 의미를, R은 값을 0으로 reset한다는 의미를 뜻한다. S와 R의 입력에 따라 플립플롭이 출력하는 값이 그에 맞게 변경되며, $q$를 현재 플립플롭에 저장된 상태, $q_{next}$를 플립플롭에 저장되는 다음 상태를 의미한다고 할 때 S와 R에 따른 $q$와 $q_{next}$의 상태도와 진리표를 그리면 아래와 같다.




| S | R | $q$ | $q_{next}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | not allowed |
| 1 | 1 | 1 | not allowed |

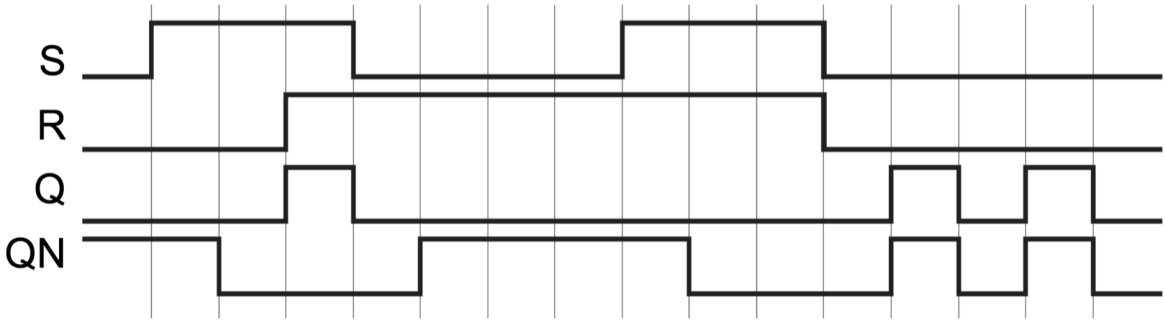
S와 R이 동시에 1이 될 수 없으며, S와 R이 동시에 0이면 이전 상태를 그대로 유지

하게 된다. 만일 S와 R이 동시에 1이었다가 동시에 다시 0으로 되었을 때는 glitch 현상이 일어난 후 어떤 상황이 일어날지 알 수 없게 되므로 되도록이면 S와 R이 동시에 1이 되는 상황을 피해야 한다. 위의 진리표를 참고하여 $q_{next}$의 카르노 맵을 나타내어서 SOP 식을 찾아내면 아래와 같다.



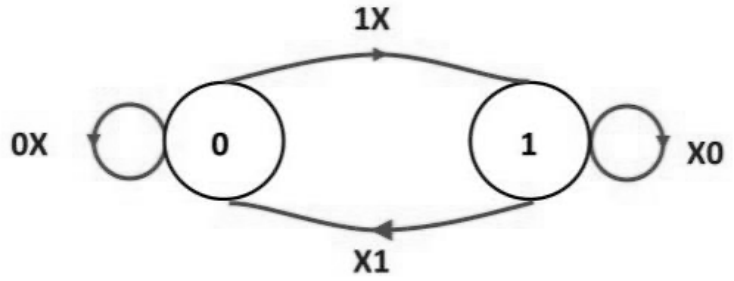

$$q_{next} = S + R'q$$

S와 R, 그리고 $q$와 $q_{next}$의 timing diagram을 그리면 아래와 같다. 실제로 clock이 1에서 0으로 바뀌거나 0에서 1로 바뀌는 시점에서 $q$와 $q'$의 값이 약간의 delay를 두고 바뀌며. S와 R은 clock이 1이 되기 전에 참고하고자 하는 특정한 값으로 설정되어 있어야 한다. (밑의 그림에서 QN은 $q$의 보수인 $q'$ 출력을 의미한다.)



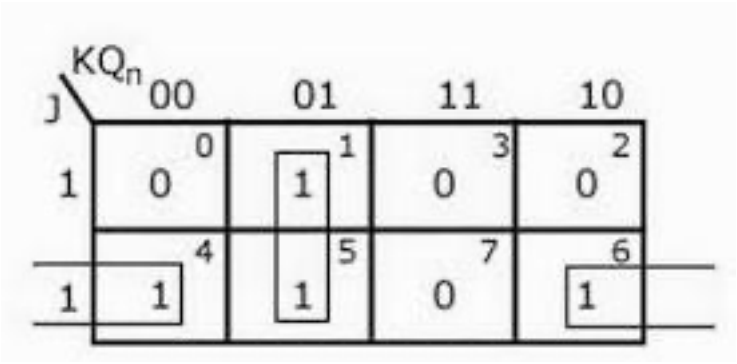

# 2. JK 플립플롭

JK 플립플롭은 RS 플립플롭과 T 플립플롭의 가능을 모두 아우르며, RS 플립플롭에서의 S가 J, R이 K에 해당한다고 볼 수 있다. 그래서 J가 1이면 JK 플립플롭의 값이 1로 set되는 것을 의미하고, K가 1이면 값이 0으로 reset되는 것을 뜻한다. RS 플립플롭에서 S와 R이 모두 1일 때를 고려하지 않는 것과는 달리, JK 플립플롭에서는 J와 K가 모두 1이면 플립플롭의 값이 보수로 바뀐다는 특징이 있다. RS 플립플롭에서와 마찬가지로 진리표와 상태도를 그리면 다음과 같다. (밑의 상태도에서 X의 의미는 0 또는 1이 올 수 있으며, don't care를 의미한다.)




| J | K | $q$ | $q_{next}$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 |

위의 진리표를 참고하여 $q_{next}$의 카르노 맵을 나타내어서 SOP 식을 찾아내면 아래와 같다. (밑의 K map에서의 Qn은 기존 플립플롭에 저장된 $q$를 의미한다.)

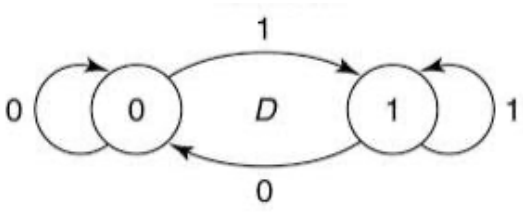



$$q_{next} = Jq' + K'q$$

# 3. D 플립플롭

D 플립플롭은 매우 간단한 플립플롭인데, 플립플롭의 다음 상태는 clock 값의 변경이 일어나기 전의 D 입력 값이다. 그래서 하나의 입력을 필요로 하며, 오직 D의 입력 값에 따라 $q_{next}$와 $q_{next}'$ 값이 바뀐다. D가 0으로 들어오면 $q_{next}$는 0이고 $q_{next}'$는 1이며,

반대로 D가 1로 들어오면 $q_{next}$는 1이고 $q_{next}'$는 0이 된다. $q$를 현재 플립플롭에 저장된 상태, $q_{next}$를 플립플롭에 저장되는 다음 상태를 의미한다고 할 때 D에 따른 $q$와 $q_{next}$의 상태도와 진리표를 그리면 아래와 같다.

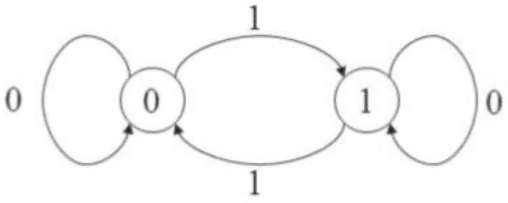



| D | $q$ | $q_{next}$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

간단한 논리식으로 나타내면 $q_{next} = D$이다.

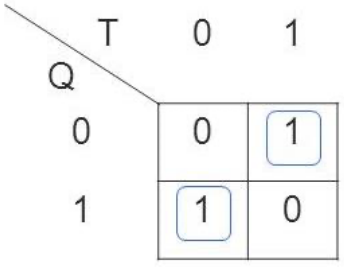
# 4. T 플립플롭

T 플립플롭은 D 플립플롭과 같이 한 개의 입력 T를 필요로 하지만, D 플립플롭에서는 D의 값이 그대로 플립플롭에 저장되는 값으로 된다면 T 플립플롭에서는 T의 값에 따라서 플립플롭에 저장되는 값이 원래 저장되었던 값의 보수로 바뀌거나 그대로 유지된다. 다시 말해서, 만일 T가 1이면 플립플롭의 상태를 변화시키고, T가 0이면 플립플롭의 상태는 유지된다. 진리표와 상태도를 그리면 다음과 같다.




| T | $q$ | $q_{next}$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

위의 진리표를 바탕으로 카르노 맵을 그려서 SOP식을 정리하면 아래와 같다.

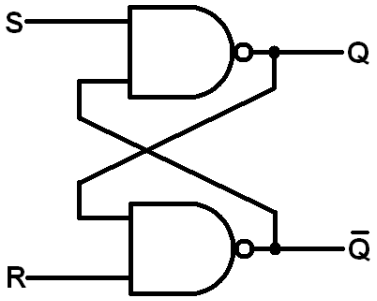



$$q_{next} = Tq' + T'q = T \oplus q$$

# 5. Latch

래치(Latch)도 플립플롭과 유사하게 1비트를 저장하는 장치로서. 2개 또는 그 이상의 논리 게이트가 피드백으로 연결되어 구성되어 있다. 래치와 플립플롭의 가장 큰 차이점은 enable 신호에 있는데, 래치는 enable 신호가 1이거나 0일 때 입력값을 그대로 출력 q로 전달한다. 반면에 플립플롭은 enable 신호인 clock의 값이 0에서 1로 상승하거나 혹은 1에서 0으로 하강하는 구간에서 약간의 delay를 두고 clock이 활성화되었을 때 플립플롭에 저장되어 있던 값과 입력 값을 확인하여 그에 맞게 설정한 값을 q로 전달한다. 래치가 enable 신호의 level에 따라 동작한다면, 플립플롭은 enable 신호의 변화에 주목하여 구간별로 그에 맞게 플립플롭의 값을 조정한다고 이해할 수 있다.

가장 간단한 2게이트 래치인 SR 래치는 각각의 게이트 출력이 다른 게이트의 입력으로 연결되어 있다. 회로도로 표현하면 아래 왼쪽 그림과 같다.




S와 Q'이 서로 NOR 게이트로 연결되어 있고, 마찬가지로 R과 Q가 서로 NOR 게이트로 연결되어 있다. 각각의 논리 연산 결과는 Q와 Q'의 출력이 된다. 만일 enable 신호가 있으면 이 신호를 S와 R과 각각 AND 게이트로 연결해주면 된다. 왼쪽 그림을 바탕으로 각 출력 Q와 Q'에 대한 논리식을 작성하면 아래와 같다.

$$Q_{next}{}' = (S+Q)',\ Q_{next} = (R+Q')'$$

SR 래치에서 S와 R은 플립플롭에서와 유사하게 S가 Q의 값을 1로 set함을, R은 Q의 값을 0으로 reset한다는 의미를 뜻한다. S와 R에 따른 Q의 진리표를 나타내면 아래와 같다. (진리표에서 X는 0 또는 1이 올 수 있으며, don't care를 의미한다.)

| S | R | $q$ | $q_{next}$ | $q_{next}{}'$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | X | 0 | 1 |
| 1 | 0 | X | 1 | 0 |
| 1 | 1 | X | 0 | 0 |

플립플롭에서와 마찬가지로 S와 R이 동시에 1이 되는 경우는 고려하지 않으며, 만일 그렇다고 하더라도 Q와 Q'의 값을 모두 0으로 설정해 버린다.

# Ⅱ Clock과 Edge-Trigger

## 1. Clock의 기능

동기형 시스템의 순차 회로에 가해지는 전기적 진동의 속도 또는 상태를 나타내며, 순차 회로에서는 clock의 변화가 발생했을 때 입력에 따라 출력을 변화시킨다. 앞에서 살펴 본 플립플롭에서는 clock의 신호 변화에 따라 플립플롭의 출력이 변화되는 시점과 그 변화되는 값을 결정짓는다는 것을 확인하면 clock이 순차 회로에서 지휘자 역할을 한다고 볼 수 있다. 입력에 따라 출력을 변화시킬 때 clock이 필요한 이유는 일정한 속도로 연산처리를 함으로써 전체적인 회로의 속도를 맞추어야 다른 작업들이 서로 엉클어지고 뒤죽박죽되는 일을 막을 수 있기 때문이다. 다시 말해 어떤 순차 회로가 전체에 걸쳐서 정확한 동작을 하려면 각 단계의 작업이 질서 있게 모두 동시에 발맞추어 진행되도록 해야 하는데 이러한 기능을 담당하는 것이 바로 clock이며, 정해진 주파수로 모든 순차 논리 회로 소자에 일정한 pulse를 내보낸다. clock은 대개 high 또는 low라는 레벨 상태로 존재할 수 있는데, high는 1로 나타내고 low는 0으로 나타내며 이 두 가지 상태 사이에서 모든 시간에 걸쳐 교대로 변화하는 신호이다.

## 2. Edge-Trigger의 특성

특정 시점에서 특정한 일을 처리하기 위해 트리거(trigger)라는 것을 사용하는데, 트리거는 clock의 타이밍에 따른 사건 처리를 담당한다. 일반적으로 트리거는 Edge Trigger와 Level Trigger가 있는데, 여기서는 전자와 관련해서 조사했다.

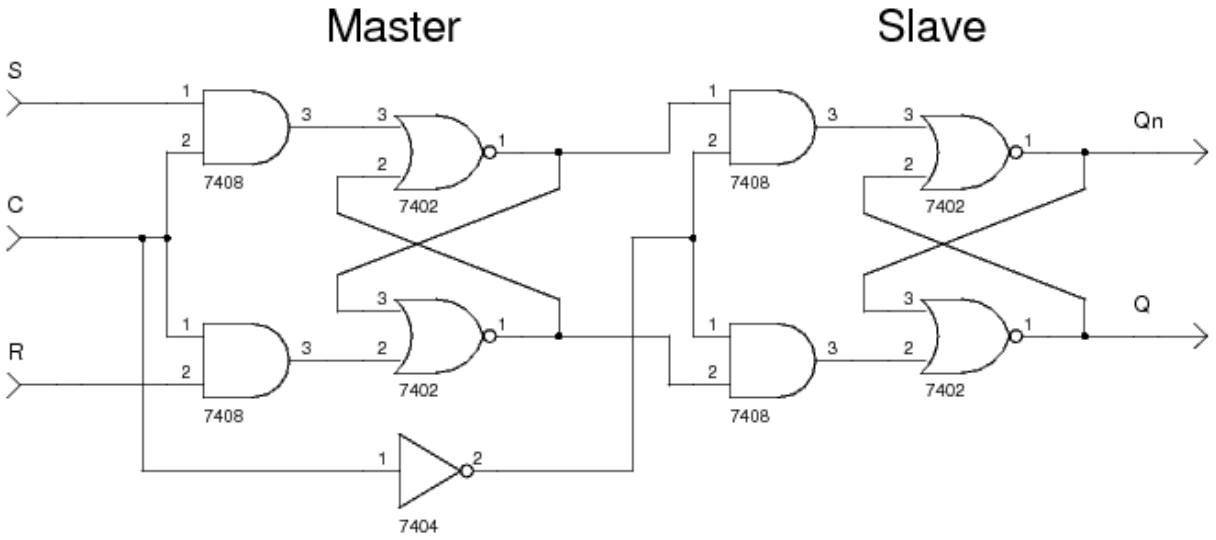
Edge Trigger는 interrupt가 발생하여 clock의 레벨이 상승하거나 하강하는 그 찰나에 어떤 사건을 처리하는 것을 의미한다. 이는 앞서 살펴 본 플립플롭에서 사용된다. 플립플롭은 clock이 0에서 1로 변화하거나 1에서 0으로 변화하고 나서 약간의 delay를 두고 상태가 변화한다는 것을 고려하면 이해할 수 있다.

clock이 1에서 0으로 변화할 때 사건을 처리하는 것을 Trailling Edge Trigger라고 하며, 0에서 1로 변화할 때 처리하는 것을 Leading Edge Trigger라고 한다.

# Ⅲ Master-Slave

# 1. Master-Slave

마스터-슬레이브(Master-Slave) 시스템은 용어의 의미를 해석하면 쉽게 이해할 수 있다. 마스터(Master)는 전체적인 일을 총괄하는 사람, 슬레이브(Slave)는 마스터에 종속되어 있어서 마스터의 일을 따르는 사람을 뜻한다. 마찬가지로 순차 논리 회로에서도 이와 같은 개념을 사용하는데, 마스터는 하나의 일을 수행하는데 있어서 동작의 주체가 되는 역할을 담당하는 것을 의미한다. 그래서 모든 다른 슬레이브들을 제어하고 명령하는 주체가 될 수 있다. 반대로 슬레이브는 마스터에 종속되어 있으며, 마스터의 지시에 따라 행동한다. 순차 논리 회로에서는 주로 두 개의 래치 또는 플립플롭을 연이어 만들어서 마스터-슬레이브 개념을 사용하며, 아래 그림은 clock이 있는 SR 플립플롭 두 개를 사용해서 한 플립플롭의 출력을 다른 플립플롭의 S와 R 입력으로 연결한 것이다.




위에서 clock이 1이면 마스터 플립플롭이 동작하고, clock이 1에서 0으로 바뀔 때 마스터 플립플롭의 출력이 슬레이브 플립플롭의 S, R 입력으로 들어가면서 슬레이브 플립플롭이 동작한다. 플립플롭 두 개를 마스터 슬레이브로 구현했을 때의 장점은 입력과 출력이 분리할 수 있다는 것이다. 예를 들어, JK 플립플롭에서 J와 K가 모두 1일 때 clock이 변화하면 출력이 보수가 된다고 했다. 그런데 만일 delay가 짧아서 출력이 변화하고 나서도 clock pulse가 계속 남아 있으면 다시 플립플롭의 값을 보수로 취해서 출력하는 문제가 발생한다. 이를 레이스 현상이라고 하는데, 마스터 슬레이브 플립플롭에서는 시간의 차이를 두고 플립플롭 두 개를 번갈아 작동시키기 때문에 레이스 현상을 방지할 수 있다. 그러나 clock pulse가 가해지고 있는 동안 입력이 변화하면 플립플롭 회로가 원치 않는 결과를 낼 수도 있다. 요약하자면 마스터 슬레이브는 clock이 변화한 이후 출력을 변화시킬 때 발생할 수 있는 오류는 최소화할 수 있지만, 입력을 변화시킬 때 발생하는 오류는 차단하기 어렵다.

# Ⅳ 기타 이론

## 1. Multi-Vibrator

멀티 바이브레이터(Multi-Vibrator)는 디지털 시스템에서 중요하게 사용되는 것 중의 하나로, 기본적으로 두 개의 inverter로 구성되어 있고 각각의 출력을 피드백해서 서로 상대 인버터의 입력으로 연결한다. 멀티 바이브레이터는 종류에 따라 이진수를 저장하기도 하고, pulse를 세기도 하며, 동기화 작업 등 여러 가지 기능을 수행할 수 있다. 멀티 바이브레이터의 종류에는 비안정(astable), 단안정(monostable), 그리고 쌍안정(bistable) 멀티 바이브레이터로 크게 세 가지가 있다.

비안정 멀티 바이브레이터는 두 개의 출력의 동작 상태가 모두 불안정 하여 1과 0 상태를 일정 주기로 번갈아 바뀌기 때문에 주로 디지털 시스템에 대한 clock 신호 또는 구형파 발진기로 사용된다. 예를 들어 입력이 0인 상태이면 inverter를 거치면서 출력이 1이 되고, 다시 피드백 되어서 inverter의 입력으로 들어가기 때문에 입력이 1이 되고 출력이 0이 된다. 이를 게이트 전파 시간만큼을 주기로 하여 전체 회로 시간에 걸쳐 반복한다.

단안정 멀티 바이브레이터는 한쪽 상태에서만 동작 상태가 안정적이고 다른 상태에서는 불안정한 회로이다. 어느 한쪽 상태에서는 멀티 바이브레이터 내 콘덴서에 충전이 되면서 다시 트리거 신호가 들어올 때까지 그 상태를 유지하게 되고, 다른 상태에서는 위의 비안정 멀티 바이브레이터처럼 1과 0을 주기적으로 번갈아 가게 된다. 그래서 비안정 멀티 바이브레이터와는 다르게 폭이 확장된 pulse를 출력할 수 있다.

쌍안정 멀티 바이브레이터는 양쪽 모두 회로가 안정적인 회로이다. 트리거 신호에 의해 동작 상태가 반전되는데, 처음 트리거 신호에 의해 동작하는 것을 set 상태, 원래의 상태로 복귀하는 것을 reset 상태라고 한다. 이는 앞서 살펴 본 플립플롭과 유사하기 때문에 플립플롭 역할을 수행할 수 있으며 1비트의 데이터를 기억하는 최소단위의 회로로 사용된다.