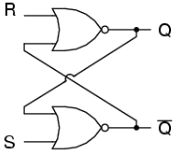
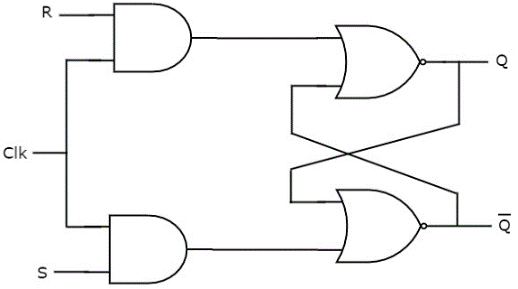
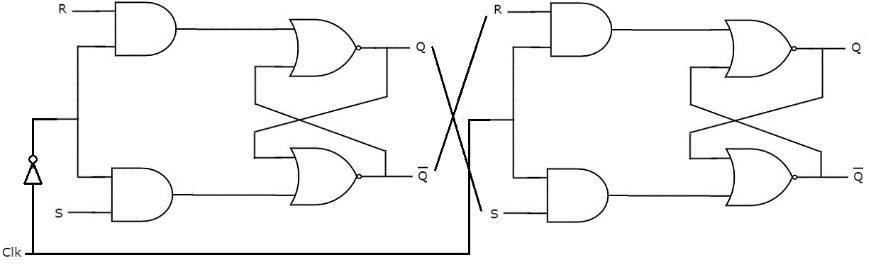
1. 11주차 예비보고서 20141196 김성희

**RS 플립플롭**

**- RS Latch:   
 이미 들어온 1또는 0에 대한 정보를 기록하는 장치다. 왼쪽 그림에서 RS가 10 일 때는 Q에 0이, 01 일 때는 1이 기록된다. 그리고 00일 때는 그 상태가 보존된다. 따라서 평상시에는 00을 유지하다가 Q에 1을 기록하고 싶다면 S에 1을, 0을 기록하고 싶다면 R에 1을 부여하면 된다. 1을 부여한 후에는 다시 0을 재차 부여하여 기록된 상태를 보존을 해야 한다. 즉 Q에 1을 부여한다면 RS에 순서대로 00 -> 01 -> 00을 부여하면 된다. 11은 invalid 값으로 쓰레기 값으로 튀게 되어 사용하지 않는다.**

**- RS Flip Flop:  
클럭을 달아 놓은 RS Latch를 2개 이어 놓은 2진 저장장치다. 역할은 RS Latch와 같지만 클럭을 넣음으로써 훨씬 안정적으로 사용이 가능하다. Clock을 통해 Q값이 바뀔 수도 바뀌지 않을 수도 있기 때문에 의미 없는 비트패턴에 의해 Q값이 튀는 것을 예방할 수 있다.**

****

**오른쪽 그림이 RS 플립플롭이다. 클럭(Clk)이 0일 때 첫번째 Q값이 나오고 클럭이 1이 될 때 두번째 Q값, 최종 기록될 비트가 산출된다. 이를 rising(leading) edge triggered라고 하며 클럭값이 0에서 1로 rising할 때 결과 값이 산출 된다. (반대로 첫번째 Latch의 클럭에 인버터를 빼고 두번째 Latch에 단 다면 이를 falling(trailing) edge triggered라고 한다. 클럭이 1에서 0으로 떨어질 때 기록될 결과가 산출된다.)**

**- RS Flip Flop의 truth table:  
(Q\*를 다음 Q값이라 하고 Q를 바꾸기 이전의 Q값이라 하자.)**

|  |  |  |
| --- | --- | --- |
| **S** | **R** | **Q\*** |
| **1** | **0** | **1** |
| **0** | **1** | **0** |
| **0** | **0** | **Q** |
| **1** | **1** | **X** |

**Q\* = S+R’Q  
R과 S는 각각 Reset과 Set을 의미한다. 따라서 Reset=1일 때 Q\*=0, Set=1일 때 Q\*=1 둘 다 0이면 보존이라고 생각하면 외우기 쉽다.**

**JK 플립플롭**

**- JK 플립플롭:  
RS 플립플롭이 RS latch 보다는 안정적이다. 그러나 아직 해결되지 않은 문제점이 있다. RS의 입력값이 11인 경우다. JK 플립플롭은 전체적으로 RS와 동일하나 입력값 11에 대해서 기능을 하나 추가한 플립플롭이다. 추가한 기능은 토글기능이다. 즉 입력값 11에 대해서 Q\* =Q’, 즉 기존의 Q값의 반전 값을 지니게 된다. RS 플립플롭과 마찬가지로 rising edge trigger 방식과 falling edge trigger 방식이 있다.**

**- JK 플립플롭 truth table:**

|  |  |  |
| --- | --- | --- |
| **J** | **K** | **Q\*** |
| **1** | **0** | **1** |
| **0** | **1** | **0** |
| **0** | **0** | **Q** |
| **1** | **1** | **Q’** |

**Q\* = JQ’ + K’Q  
J와 K를 각각 RS 플립플롭의 S와 R이라고 생각하면 된다. 즉 J = Set을, K = Reset을 의미한다. Set과 Reset이 모두 0인 경우는 이전의 Q값을 보존하고 모두 1인 경우는 이전의 Q값의 반전 값인 Q’을 가진다, 즉 토글한다.**

**D 플립플롭**

**- D 플립플롭:  
RS플립플롭부터 언급하였지만 사실 플립플롭 중 가장 단순한 형태와 기능을 가진 회로다. RS방식은 Set Rest 보존 3가지 기능을, JK 방식은 Set Rest 보존 토글 4가지 기능을 가진 것에 반해 D 플립플롭은 Set과 Rest 두가지 기능밖에 없다. 마찬가지로 rising/falling edge trigger 두가지 방식이 있다.**

**- D 플립플롭 truth table:**

|  |  |
| --- | --- |
| **D** | **Q\*** |
| **1** | **1** |
| **0** | **0** |

**Q\*=D  
D=1은 Set을, D=0은 Reset을 의미한다.**

**T 플립플롭**

**- T 플립플롭:**

**다른 플립플롭과는 다르게 Set과 Rest기능이 존재하지 않는다. 대신 JK 플립플롭의 4가지 방식 중 토글과 보존 기능을 가진 단순한 형태의 T플립플롭이다. 마찬가지로 rising/falling edge trigger 2가지 방식이 있다.**

**- T 플립플롭 truth table:**

|  |  |
| --- | --- |
| **T** | **Q\*** |
| **1** | **Q’** |
| **0** | **Q** |

**Q\* = TQ’+T’Q = T⊕Q  
T=1일 때 토글을, T=0일 때 보존을 의미한다.**

**Latch의 기능**

**앞서 RS 플립플롭에서 언급했던 RS Latch를 떠올리면 쉽다. 플립플롭 보다 조금 불안정한 2진 저장장치이며 NOR 또는 NAND 게이트 2개로 구현이 가능하다. RS Latch에 나오는 그림에서 NOR 게이트를 NAND 게이트로 치환하기만 하면 NAND gate RS Latch이다. 단, 이 때는 S=1일 때 Set을 활성화 시키는 것이 아닌 S=0일 때 Set을 활성화 시킨다. 마찬가지로 R=0일 때 Reset이 활성화 된다. 즉 SR이 01, 10, 11 일 때 각각 Q가 1, 0, Q 값을 지니게 된다. 00은 invalid값이다. Gated Latch의 형태도 있는데 이는 Clock 신호와 같은 입력을 Latch의 입력에 각각 AND 게이트로 연산함으로써 기존의 Latch의 불안정함을 보안한 형태다.**

**Clock**

**클럭은 말 그대로 시계에 따온 단어로 타이머 역할을 한다. 이 클럭 신호에 맞춰서 값을 쓰고 지운다. 만약 클럭이 없다면 내가 원하지 않는 값이 중간에 생겼다가 사라지는 경우가 있어서 버그가 발생하기 쉽다. 이것 때문에 플립플롭이 Latch보다 안정적이다.**

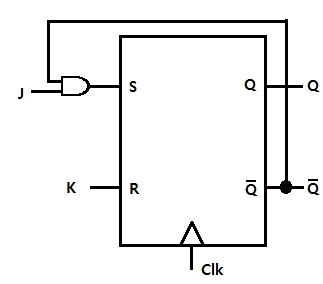
**Edge-Trigger**

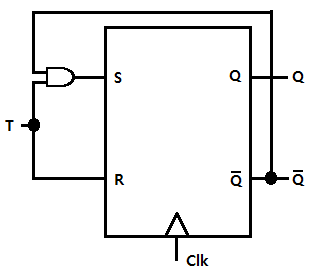
**Clock 신호를 응용해서 회로를 안정하게 만들기 위한 방식으로 clock의 값이 바뀌는 순간에 결과가 출력되도록 하는 방식이다. 예를 들어 0에서 1로 바뀌거나 1에서 0으로 바뀌는 순간에 출력값이 변경되는 형식이다. RS 플립플롭 설명을 참조하자.**

**Master-Slave**

**기본적인 의미는 마스터라는 주체가 다른 모든 슬레이브들의 행동을 지시하는 형태의 시스템을 말한다. 다른 말로는 Primary-Secondary라고 할 수 있다. 플립플롭은 Master Latch와 Slave Latch 두개로 구성된다. 즉 RS플립플롭의 두번째 그림과 그림에서 첫번째 Latch가 Master(or Primary) Latch이고 두번째 Latch가 Slave(or Secondary) Latch이다. 첫번째 Latch에서 나온 결과값으로 인해 두번째 Latch값이 변경되는 것을 통해 Master가 Slave에게 명령을 내리는 것의 개념을 이해할 수 있다.**

**기타 이론**

**JK와 T플립플롭은 모두 RS 플립플롭을 기본으로 구현되는데 어떻게 구현되는지는 Q\*의 식을 통해 쉽게 알 수 있다. Q\*=S + R’Q = JQ’ + K’Q = TQ’ + T’Q. RS 플립플롭에 S대신 JQ’을 R대신 K를 대입하면 JK 플립플롭이 되며 S대신 TQ’을 R대신 T를 대입하면 T 플립플롭이 된다. 즉 다음과 같은 형태의 그림이 나온다.**

****