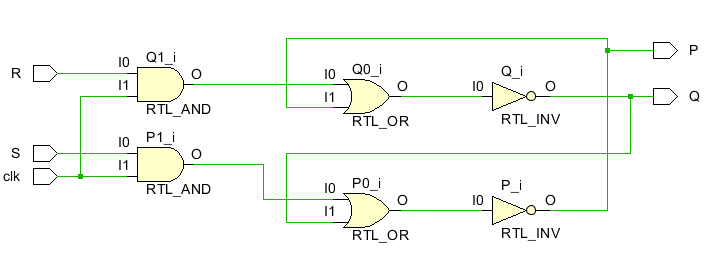
11주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20191619 이름: 이동석

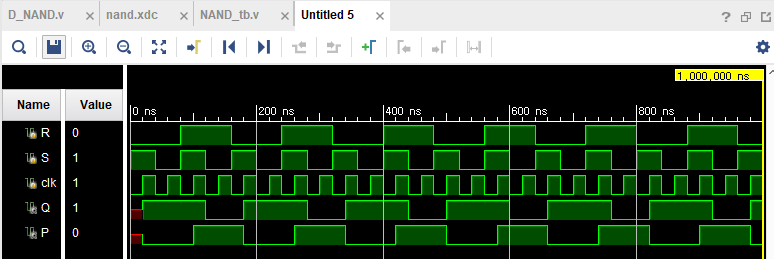
**1. RS-FlipFlop**

1.1 NOR

텍스트이(가) 표시된 사진

자동 생성된 설명

RS플리플롭은 2개의 AND게이트와 2개의 NOR게이트로 회로를 구현할 수 있다. 코드를 확인해보면 2개의 &&와 2개의 NOR게이트를 사용해 표현한 것을 볼 수 있다. RS플리플롭은 RS 래치에서 Clock이 추가된 형태이다. 따라서, clk의 입력이 1일 때 동작한다.



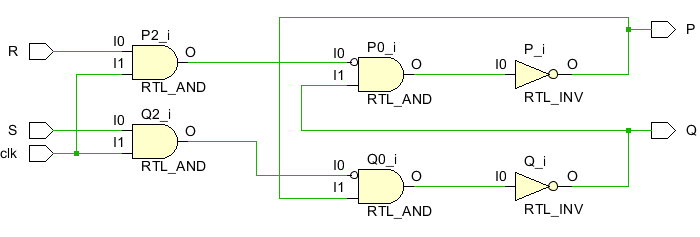
시뮬레이션을 확인해보면, Q와 P의 값은 초기 상태에 값이 없었지만, clk가 1이되면서 값이 생겼다. Clk가 1이 되는 순간 S는 1이고 R은 0이다. 따라서, Q는 1이되고, P는 0이된다. 이후 다음 clk가 1이되는 경우 S와 R모두가 0이므로 현재 상태를 유지한다. R과 S가 모두 1이되는 경우 Q와 P가 모두 1이되는데 이는 RS플리플롭에서 잘못된 입력임을 기억해야한다. (시뮬레이션 상에서는 나타나고 있다.) 이후, R은 1이되고 S가 0이되면 Q가 0으로 변하고 P가 1이되는 것을 확인할 수 있다. 따라서, 진리표를 다음과 같이 작성할 수 있다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 입력순서 | R | S | Q | ~Q(P) |
| (1) | 0 | 1 | 1 | 0 |
| (2) | 0 | 0 | 1 | 0 |
| (3) | 1 | 0 | 0 | 1 |
| (4) | 0 | 0 | 0 | 1 |
| (5) | 1 | 0 | 0 | 1 |
| (6) | 1 | 1 | X | X |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q(present) | Next State Q+ | | | |
|  | Input RS | | | |
|  | 00 | 01 | 10 | 11 |
| 0 | 0 | 1 | 0 | X |
| 1 | 1 | 1 | 0 | X |

1.2 NAND

텍스트이(가) 표시된 사진

자동 생성된 설명

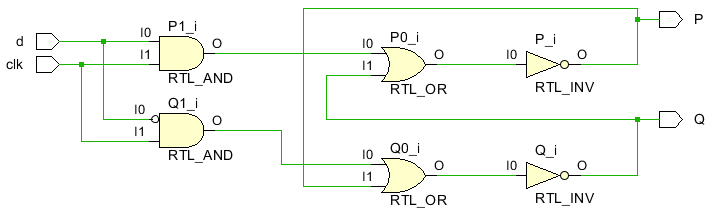
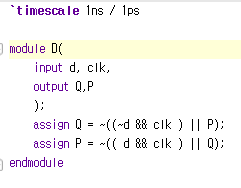
NAND게이트도 NOR와 동작은 동일하다. 4개의 NAND게이트로 구현했다. 마찬가지로, Clk가 1일 때 동작하도록 구현했다.

텍스트, 실내이(가) 표시된 사진

자동 생성된 설명

역시 시뮬레이션도 거의 동일하다. Q와 P의 값은 초기 상태에 값이 없었지만, clk가 1이되면서 값이 생겼다. Clk가 1이 되는 순간 S는 1이고 R은 0이다. 따라서, Q는 1이되고, P는 0이된다. 이후 다음 clk가 1이되는 경우 S와 R모두가 0이므로 현재 상태를 유지한다. R과 S가 모두 1이되는 경우 Q와 P가 모두 1이되는데 앞과는 다르게 모두 0이 된다. 하지만, 본래 금지된 입력이다. 이후, R은 1이되고 S가 0이되면 Q가 0으로 변하고 P가 1이되는 것을 확인할 수 있다.

**2. D-FlipFlop**

****

D플리플롭은 현재 상태가 입력과 동일하게 결정되는 특징을 가진다. 구현은 1개의 NOT과 2개의 AND 2개의 NOR게이트가 사용된다. 마찬가지로, clk가 있어 1일 때 동작하도록 구현했다.

테이블이(가) 표시된 사진

자동 생성된 설명

Clk가 1이 될 때, 입력이 0으로 들어오므로 Q가 0, P는 1로 바뀌는 것을 확인할 수 있다. 다음으로, clk가 1이되면 d의값이 1이므로, Q는 다시 1로, P가 0으로 바뀐다. 또한, clk가 0이면, 이전 상태를 그대로 유지하는 모습을 보인다. 이를 바탕으로 표를 작성할 수 있다.

|  |  |  |
| --- | --- | --- |
| D | Q | ~Q(P) |
| 0 | 0 | 1 |
| 0 | 0 | 1 |
| 1 | 1 | 0 |
| 0 | 0 | 1 |
| 1 | 1 | 0 |
| 1 | 1 | 0 |

|  |  |  |
| --- | --- | --- |
| State / Next state | 0(D) | 1(D) |
| 0 | 0 | 1 |
| 1 | 0 | 1 |

**3. 결과 및 논의**

이번 실습에서 RS 플리플롭과 D플리플롭을 구현하였다. RS플리플롭은 R이 Reset을 나타내므로 R이 1일 때 상태는 0이되며 S가 1일 떄 상태는 1이된다. 또한 00일때는 이전상태를 유지하는 모습을 보였다. D플리플롭의 경우에는 현재 상태와 관계없이 입력되는 값이 상태가 되었다. 시뮬레이션을 보고 진리표도 작성가능했으며, 원하는 값을 출력하긴 하였다.

하지만, 코드는 정확하게 원하는 Rising edge 플리플롭이 아니다. Clk가 1일 때와 0일 때만 구분된다. 만약, rising edge에서만 구동하기 위해서는 코드를 조금 바꿔 작성해야한다. Always 문과 posedge를 사용하면 가능하다. 따라서, 다음과 같이 always @ (posedge clk) 작성하면, clk의 rising edge를 의미한다. 만약, D플리플롭을 작성한다면 CLR변수를 추가하고,

Initail Q <= 1’b0; // 초기값

always @ (posedge clk) // negedge는 falling edge를 의미

beign

if (CLR) Q <= 1’b0; // 0으로 초기화

else Q <= D;

end

와 같이 작성 가능할 것이다. 마찬가지로, RS플리플롭도 동일하게 작성가능하다.

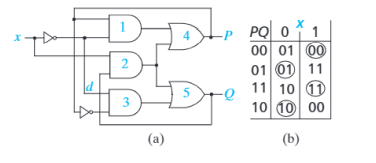
**4. 추가이론**

플리플롭은 추가적인 입력을 더할 수 있다. 이러한 입력은 클럭 신호와는 관계없이 플리플롭을 설정할 수 있기에 보통 asynchronous(비동기식)으로 부른다. 보통 프리셋 / 클리어라고 부른다. 프리셋은 플리플롭을 Set상태로 구동, 클리어는 Reset상태로 구동한다.

테이블이(가) 표시된 사진

자동 생성된 설명

순차회로에서는 보통 피드백 루프를 회로의 상태로 정의하는 반면, 비동기식 순차회로에서는 입력의 변경에 따라 회로의 상태가 변경될 수 있다. 이러한 예로는 래치가 있다. 뿐만 아니라 마스터 슬레이브나 에지 트리거 플리플롭은 클락의 입력에만 바뀌므로 내부적으로 플리플롭은 비동기적 회로라고 볼 수 있다. 아래는 비동기 회로의 예이다.



클락을 사용하지 않기 때문에, 전력의 소비가 더 적고 속도도 더 빠르다는 장점이 있다. 하지만, 오늘날 대부분은 동기식이며 비동기식의 치명적인 단점 때문에 사용하지않는다.