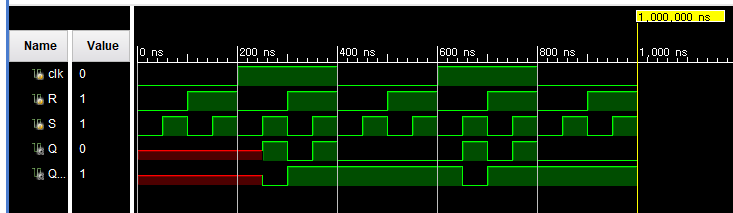
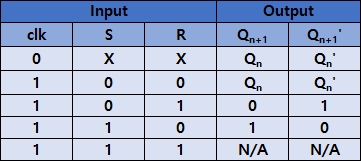
11주차 결과보고서

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

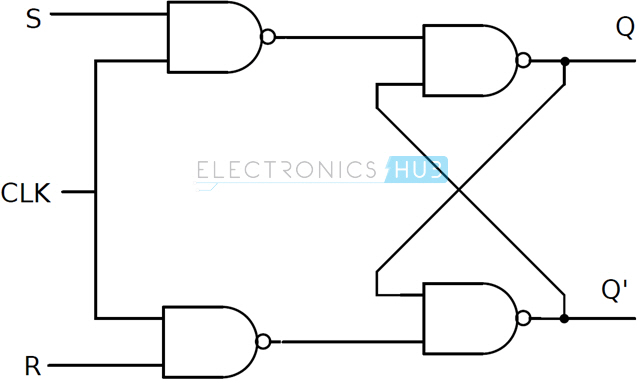
**1.**



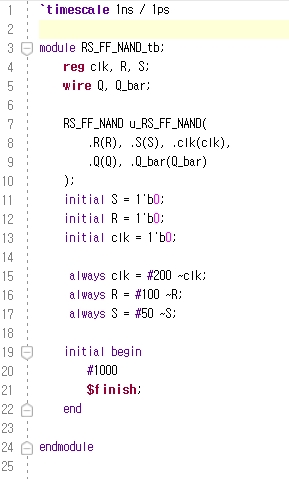
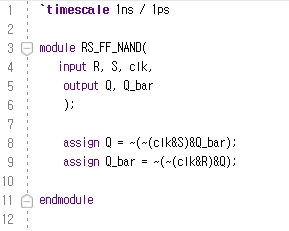
[NAND Gate RS Flip-Flop의 Simulation 결과]



[NAND Gate RS Flip-Flop의 진리표]



[NAND Gate RS Flip-Flop의 회로도]



[NAND Gate RS Flip-Flop의 Verilog Code]

RS 플립플롭은 위 회로도와 같이 4개의 NAND 게이트로 구현이 가능하며, Gated RS Latch에 Gate 신호로 clk(클럭 신호)를 입력한 구조로 되어있다.

출력 Q는 clk와 S를 NAND 연산한 값을 Q’와 NAND 연산한 값으로, 출력 Q’는 clk와 R을 NAND 연산한 값을 Q와 NAND 연산하게 구현했다.

위 진리표를 보면 clk가 0이면 현재 Q의 값을 다음 state의 Q의 값으로 저장하는 것을 볼 수 있는데, clk가 0인 경우, 이 되고, 이 되므로, Gate 뒤쪽의 NAND Gate에서 , 가 되므로 입력 R, S의 값과 무관하게 현재 Q의 값이 다음 state의 Q로 그대로 저장되는 것을 알 수 있다.

이는 위쪽의 시뮬레이션 결과에서도 clk가 0이면 R, S의 값과 관계없이 항상 Q와 Q’의 값이 일정한 것을 확인할 수 있다.

clk의 값이 1인 경우 다음 state의 Q값은 입력값 R, S에 영향을 받게 되는데, 이 경우, Gate의 NAND 게이트들은 각각 , 이 되므로, Gate 뒤의 NAND 게이트들은 , 를 연산하게 된다.

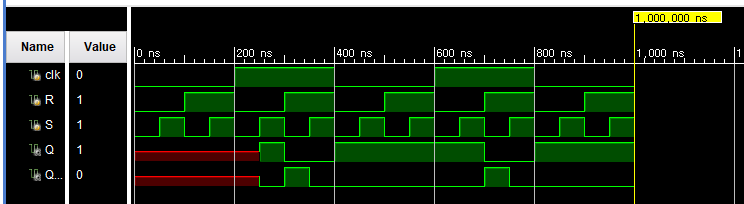
먼저 S=R=0인 경우, , 으로, 진리표와 같이 현재 Q값이 다음 state의 Q로 그대로 저장되는 것을 알 수 있다.

S=0, R=1인 경우, , 으로, 진리표와 같이 다음 state의 Q가 0으로 저장되는 것을 알 수 있다.

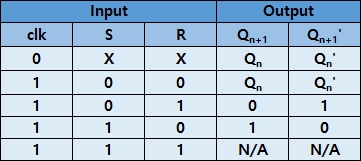
S=1, R=0인 경우, , 으로, 진리표와 같이 다음 state의 Q가 1로 저장되는 것을 알 수 있다.

마지막으로 S=R=1인 경우, , 으로, 다음 state의 Q와 Q’가 모두 1로 저장되는 것을 알 수 있는데, 이와 같은 경우는 invalid input으로 취급받는다.

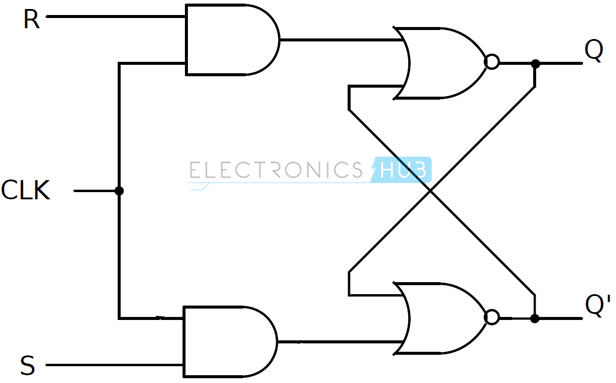
clk가 1일 때 S와 R에 값에 따른 Q, Q’ 값의 변화는 위의 시뮬레이션 결과와 모두 일치하는 것을 확인 할 수 있다. 시뮬레이션의 앞부분(0ns ~ 250ns)에서 Q와 Q’의 처음 상태가 지정되지 않았으므로 Q와 Q’의 값을 특정 값으로 설정하는 입력(250ns에서 clk=1, S=1, R=0)이 들어올 때까지는 don’t care로 출력된다.



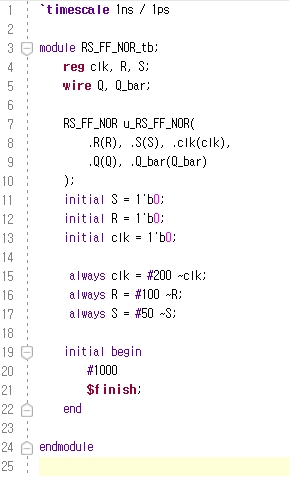
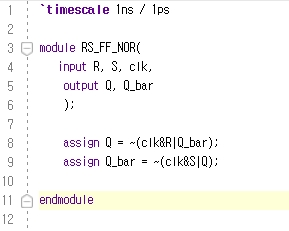
[NOR Gate RS Flip-Flop의 Simulation 결과]



[NOR Gate RS Flip-Flop의 진리표]



[NOR Gate RS Flip-Flop의 회로도]



[NOR Gate RS Flip-Flop의 Verilog Code]

RS 플립플롭은 위 회로도와 같이 2개의 AND 게이트와 2개의 NOR 게이트로 구현이 가능하며, Gated RS Latch에 Gate 신호로 clk(클럭 신호)를 입력한 구조로 되어있다.

출력 Q는 clk와 R를 AND 연산한 값을 Q’와 NOR 연산한 값으로, 출력 Q’는 clk와 S을 AND 연산한 값을 Q와 NOR 연산하게 구현했다.

위 진리표를 보면 clk가 0이면 현재 Q의 값을 다음 state의 Q의 값으로 저장하는 것을 볼 수 있는데, clk가 0인 경우, 이 되고, 이 되므로, Gate 뒤쪽의 NOR Gate에서 , 가 되므로 입력 R, S의 값과 무관하게 현재 Q의 값이 다음 state의 Q로 그대로 저장되는 것을 알 수 있다.

이는 위쪽의 시뮬레이션 결과에서도 clk가 0이면 R, S의 값과 관계없이 항상 Q와 Q’의 값이 일정한 것을 확인할 수 있다.

clk의 값이 1인 경우 다음 state의 Q값은 입력값 R, S에 영향을 받게 되는데, 이 경우, Gate의 AND 게이트들은 각각 , 가 되므로, Gate 뒤의 NOR 게이트들은 , 를 연산하게 된다.

먼저 S=R=0인 경우, , 으로, 진리표와 같이 현재 Q값이 다음 state의 Q로 그대로 저장되는 것을 알 수 있다.

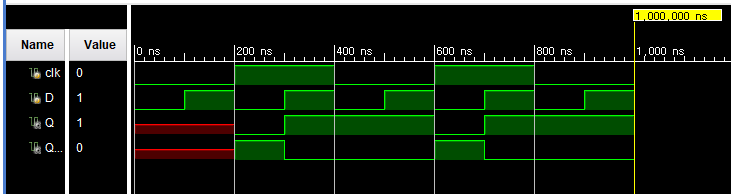
S=0, R=1인 경우, , 으로, 진리표와 같이 다음 state의 Q가 0으로 저장되는 것을 알 수 있다.

S=1, R=0인 경우, , 으로, 진리표와 같이 다음 state의 Q가 1로 저장되는 것을 알 수 있다.

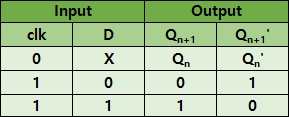
마지막으로 S=R=1인 경우, , 으로, 다음 state의 Q와 Q’가 모두 0으로 저장되는 것을 알 수 있는데, 이와 같은 경우는 invalid input으로 취급받는다.

clk가 1일 때 S와 R에 값에 따른 Q, Q’ 값의 변화는 위의 시뮬레이션 결과와 모두 일치하는 것을 확인 할 수 있다. 시뮬레이션의 앞부분(0ns ~ 250ns)에서 Q와 Q’의 처음 상태가 지정되지 않았으므로 Q와 Q’의 값을 특정 값으로 설정하는 입력(250ns에서 clk=1, S=1, R=0)이 들어올 때까지는 don’t care로 출력된다.

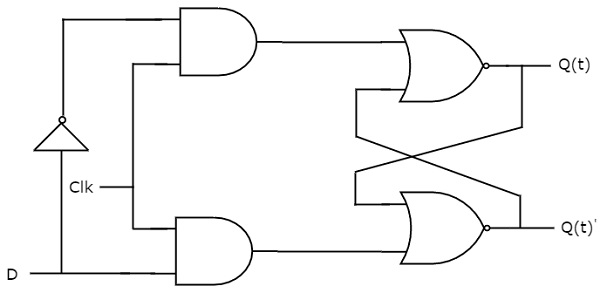
**2.**

****

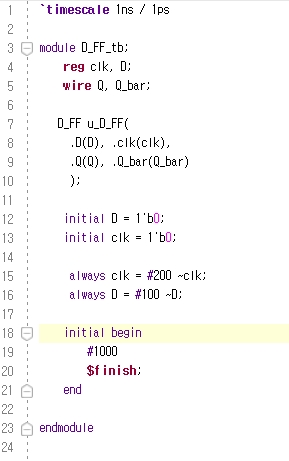
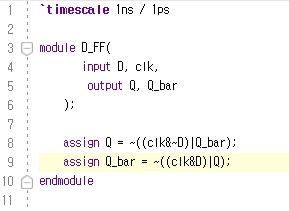
[D Flip-Flop의 Simulation 결과]

****

[D Flip-Flop의 진리표]

****

[D Flip-Flop의 회로도]



[D Flip-Flop의 Verilog Code]

D 플립플롭은 위 회로도와 같이 NOR Gate RS Flip-Flop에서 입력 S를 D로, 입력 R을 D’로 입력함으로써 구현할 수 있으며, 연결된 RS Flip-Flop는 D가 0일 때 S=0, R=1을, D가 1일 때 S=1,R=0을 입력받게 된다.

출력 Q는 clk와 D’를 AND 연산한 값을 Q’와 NOR 연산한 값으로, 출력 Q’는 clk와 D을 AND 연산한 값을 Q와 NOR 연산하게 구현했다.

위 진리표를 보면 clk가 0이면 현재 Q의 값을 다음 state의 Q의 값으로 저장하는 것을 볼 수 있는데, clk가 0인 경우, 이 되고, 이 되므로, Gate 뒤쪽의 NOR Gate에서 , 가 되므로 입력 D의 값과 무관하게 현재 Q의 값이 다음 state의 Q로 그대로 저장되는 것을 알 수 있다.

이는 위쪽의 시뮬레이션 결과에서도 clk가 0이면 D의 값과 관계없이 항상 Q와 Q’의 값이 일정한 것을 확인할 수 있다.

clk의 값이 1인 경우 다음 state의 Q값은 입력값 D에 영향을 받게 되는데, 이 경우, Gate의 AND 게이트들은 각각 , 가 되므로, Gate 뒤의 NOR 게이트들은 , 를 연산하게 된다.

D=0인 경우, , 으로, 진리표와 같이 다음 state의 Q가 0으로 저장되는 것을 알 수 있다.

D=1인 경우, , 으로, 진리표와 같이 다음 state의 Q가 1로 저장되는 것을 알 수 있다.

clk가 1일 때 D의 값에 따른 Q, Q’ 값의 변화는 위의 시뮬레이션 결과와 모두 일치하는 것을 확인 할 수 있다. 시뮬레이션의 앞부분(0ns ~ 200ns)에서 Q와 Q’의 처음 상태가 지정되지 않았으므로 Q와 Q’의 값을 특정 값으로 설정하는 입력(200ns에서 clk=1, D=0)이 들어올 때까지는 don’t care로 출력된다.

**3.**

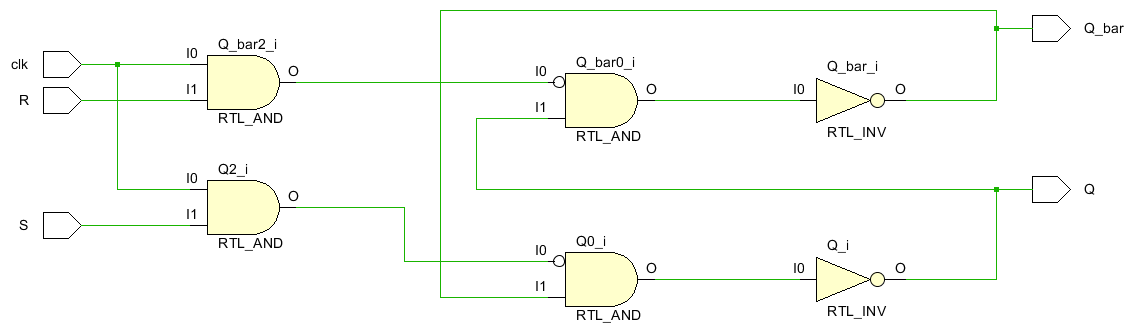
이번 실험을 통해 RS Flip-Flop, D Flip-Flop의 구조와 작동방식에 대해서 이해할 수 있었다. RS Flip-Flop은 NAND Gate를 통한 구현과 NOR Gate를 통한 구현이 가능했다.

D Flip-Flop는 RS Flip-Flop에서 입력 S에 D를, 입력 R에 D’를 입력시킴으로써 구현이 가능한데, RS Flip-Flop이 NAND Gate와 NOR Gate 2개의 방식으로 구현할 수 있으므로 D Flip-Flop 또한 각 종류의 RS Flip-Flop을 사용하여 2종류의 D Flip Flop을 구현할 수 있다. RS Flip-Flop은 clk가 1인 경우 활성화 되며, 이 때는 S, R의 값에 따라 Q의 값이 결정되는데, S, R이 모두 0일때는 현재 Q값을 유지하고, S=0, R=1이면 Q를 0으로 Reset, S=1, R=0이면 Q를 1로 Set한다. S=R=1인 경우에는 유효한 input으로 취급하지 않으며, Q와 Q’가 서로 보수관계가 아닌 값을 가지게 되는 유효하지 않은 output을 출력하는 것을 시뮬레이션으로 확인할 수 있었다.

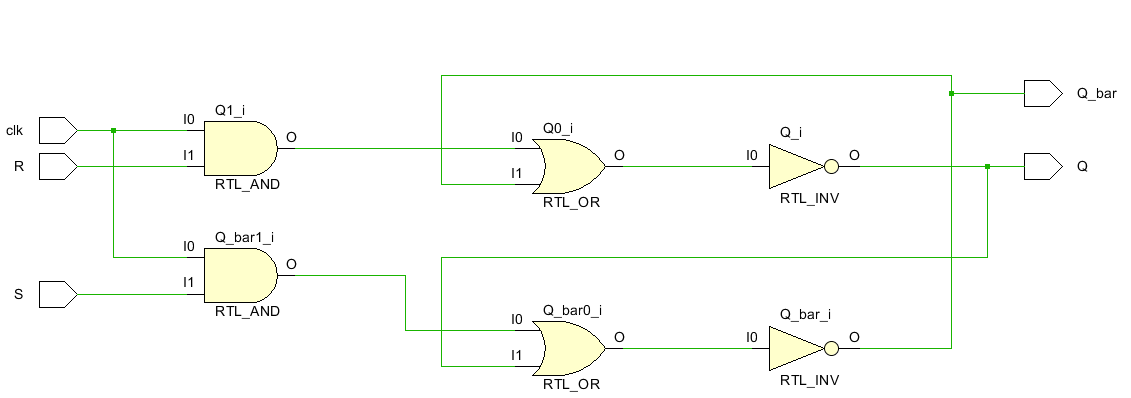
D Flip-Flop은 RS Flip-Flop에서 D에 값에 따라 S=0, R=1 (D=0), S=1, R=0 (D=1) 두 가지 경우만 출력하게 만들었으므로 유효하지 않은 input은 존재하지 않고, Q와 Q’는 항상 보수관계를 만족하는 값을 가지게 된다.

clk가 1일 때 D가 0인 경우 S=D=0, R=D’=1이 되어 Q의 값을 0으로 Reset하며, D가 1인 경우 S=D=1, R=D’=0이 되어 Q의 값을 1로 Set 하는 것을 시뮬레이션으로 확인할 수 있었다.

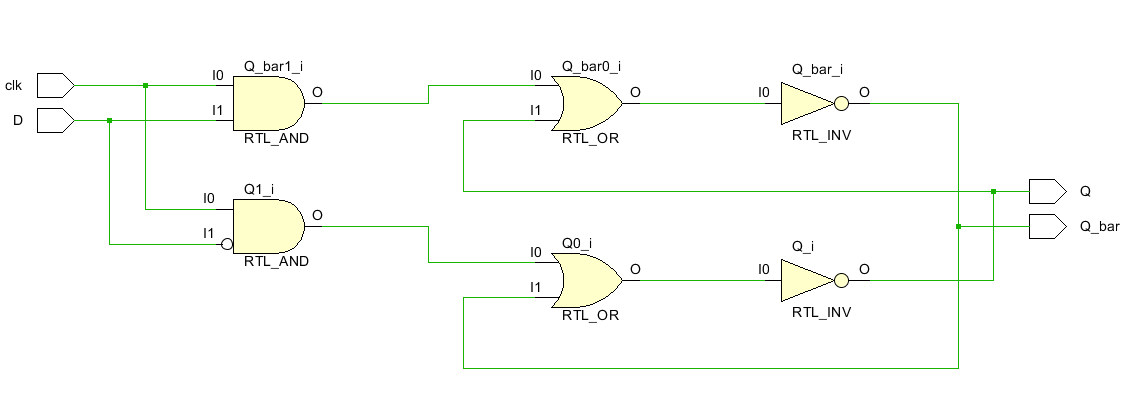
아래의 사진들은 1~2.까지 각각의 Schematic이다.



[NAND Gate RS Flip-Flop의 Schematic]



[NOR Gate RS Flip-Flop의 Schematic]



[D Flip-Flop의 Schematic]

**4.**

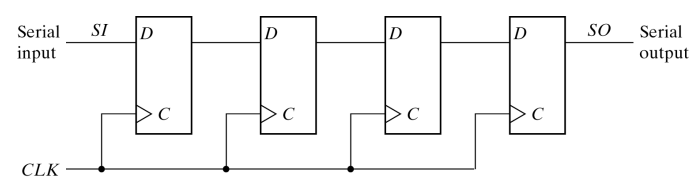
- 레지스터 (Register)

레지스터는 컴퓨터에 CPU 등에 내장되어 있는 빠르고 용량이 작은 저장장치이다. 외부로부터 들어오는 데이터를 저장하거나 이동하는 목적으로 사용되며, 클럭 신호를 공유하는 n개의 플립플롭들을 연결하여 n비트를 저장할 수 있다.

레지스터는 1개의 클럽 입력을 공유하는 다수의 플립플롭들의 묶음을 의미하며 카운터(Counter), 시프트 레지스터(Shift Register) 등의 비트 저장 및 이동이 가능한 소자들을 총칭한다.

레지스터에서 데이터를 전송할 때는 직렬 전송방식과 병렬 전송 방식이 존재하는데, 직렬전송 방식은 이동 레지스터(shift register)라 하며, 데이터를 1비트씩 이웃한 플립플롭 회로로 이동하고 가장 끝의 플립플롭은 또 다른 레지스터의 플립플롭으로 전송시킨다.

직렬전송은 회로가 간단하다는 장점이 있으나, 데이터 전송시간이 오래 걸린다는 단점이 있다.



[4bit Shift Register – Clock Edge마다 다음 플립플롭으로 데이터가 이동함]

병렬전송 방식은 레지스터에 기억된 전체 내용을 하나의 제어 신호를 통해 다른 레지스터로 동시에 전송시키는 방식이다.

병렬전송 방식은 데이터 전송시간이 매우 빠르지만, 레지스터의 비트 수만큼 데이터 전송경로가 필요하므로 직렬전송 방식에 비해 회로도가 복잡해진다는 단점이 있다.