컴퓨터공학실험2 11주차 결과 보고서

전공: 컴퓨터공학 학년: 2학년 학번: 20201635 이름: 전찬

**0. 목차**

1. RS Flip-Flop의 결과 및 simulation 과정 설명

2. D Flip-Flop의 결과 및 simulation 과정 설명

3. 결과 검토 및 논의 사항

4. 추가 이론 조사 및 작성

**1. RS Flip-Flop의 결과 및 simulation 과정**

RS flip-flop은 2개의 input(Reset, Set)을 받아 input 3가지 경우에 맞추어 1bit data를 저장할 수 있는 flip-flop 이다. 이때 R = 1, S = 0 이면 output Q = 0, R = 0, S = 1 이면 Q = 1, R = 0, S = 0 이면 Q = 이전 상태를 유지 하는 형태이다. 또한 RS flip-flop은 RS latch의 Enable input에 clock pulse를 연결한 형태라고 할 수 있다. 이러한 RS flip-flop은 두 가지 형태로 구현할 수 있는데, NAND로 구현한 형태, NOR로 구현한 형태가 존재한다. 이때 NAND를 사용한다면 active HIGH input을 사용하게 되며, NOR gate를 사용한다면 active LOW input을 사용하게 된다. 이를 회로로 구현할 수 있는데 이를 구현하는 verilog 코드는 아래와 같다.

`timescale 1ns / 1ps

module RSlatch\_NAND(

input R, S, E,

output Q, Q\_com

);

assign Q = ~(Q\_com&~(S&E));

assign Q\_com = ~(Q&~(R&E));

endmodule

module RSlatch\_NOR(

input R, S, E,

output Q, Q\_com

);

assign Q = ~(Q\_com|~(~R|~E));

assign Q\_com = ~(Q|~(~S|~E));

endmodule

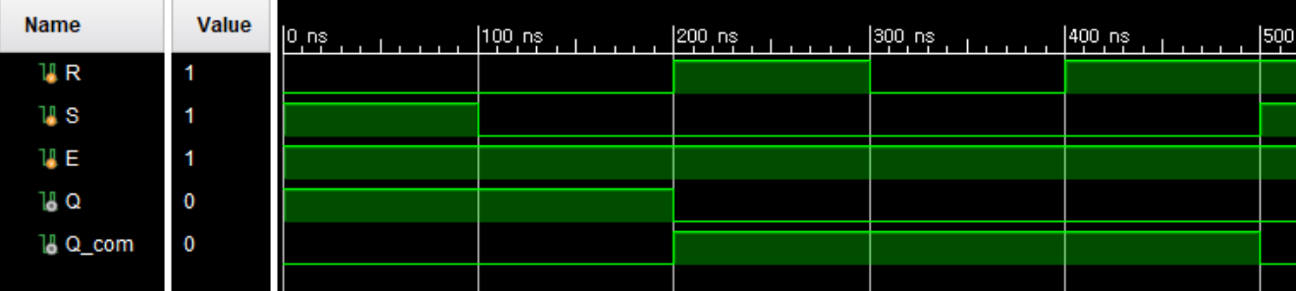
<RS flip-flop의 verilog 코드>

위 코드를 통해서 NOR gate로 구현한 module RSlatch\_NOR 이 active LOW input을 사용했다는 것을 파악할 수 있다.

또한 simulation 결과는 아래와 같다.



<RS flip-flop(NAND)의 simulation 결과>



<RS flip-flop(NOR)의 simulation 결과>

두 simulation 결과에서 약간의 차이점이 있다면, 실제 상황에서는 고려되지 않지만 NAND gate로 구현한 RS flip-flop은 R = 1, S = 1,일 때 Q, Q\_com = 1이며, NOR gate로 구현한 flip-flop은 Q, Q\_com = 0이라는 차이점이 존재하기도 한다.

**2. D Flip-Flop의 결과 및 simulation 과정**

D Flip-Flop은 RS flip-flop에 S = D, R = D’ 형태를 대입하는 flip-flop이다. 만약 D = 1 이라면, Q = 1(set)을 수행하며, D = 0이면 Q = 0(reset)을 수행한다. 또한 input으로 1, 1이 들어가지 않음이 보장된다. 이에 대한 Verilog 코드는 아래와 같다.

`timescale 1ns / 1ps

module RSlatch(

input R, S, E,

output Q, Q\_com

);

assign Q = ~(Q\_com&~(S&E));

assign Q\_com = ~(Q&~(R&E));

endmodule

module Dlatch(

input D, E,

output Q\_out, Q\_com\_out

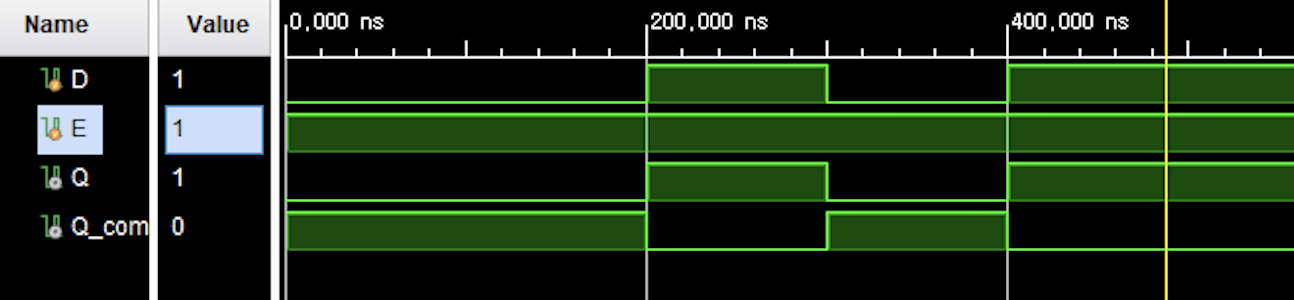
);

RSlatch in\_latch (.R(~D), .S(D), .E(E), .Q(Q\_out), .Q\_com(Q\_com\_out));

endmodule

<RS flip-flop의 verilog 코드>

또한 simulation 결과는 아래와 같다.



<D flip-flop의 simulation 결과>

**3. 결과 검토 및 논의 사항**

이번 실험에서는 직접 Enable input(clock pulse에 연결될 input)이 포함된 RS latch, D latch를 직접 구현하며, 실제로 data를 store하는 과정과, 1bit를 저장하는 flip-flop을 구현했다. 또한 FPGA로 simulation을 수행할 때, 처음에는 generate bitstream이 수행되지 않았는데, XDC 참고 파일을 넣어서 수행되게 만드는 과정을 거치기도 했다. 추가로 이번 실습에서는 JK flip-flop의 구현을 진행하지는 않았는데, 이를 구현해보는 것도 좋을 것 같다.

**4. 추가 이론 조사 및 작성**

위에서 말한 것처럼, JK Flip-flop을 실제 Verilog에서 구현해보자. 우선 JK flip-flop은 RS flip-flop에서 input J = 1, K = 1 이 추가적인 기능을 수행할 수 있도록 하는 회로이다. 이는 아래와 같다.

(1) J = 1, K = 0 이면 Q = 1

(1) J = 0, K = 1 이면 Q = 0

(1) J = 0, K = 0 이면 Q = Q

(1) J = 1, K = 1 이면 Q = Q’

따라서 J = 1, K = 1일 때만 RS flip-flop에서 추가해주면 되는데, 이는 아래 형태를 따른 것이다.

(1) 만약 Q = 1 이었다면, Q = 0을 만들어주기 위해서 RS flip-flop에 R = 1, S = 0을 대입해주어야 한다.

(2) 만약 Q = 0 이었다면, Q = 1을 만들어주기 위해 RS flip-flop에 R = 0, S = 1을 대입해주어야 한다.

따라서 J&Q = R, K&Q\_com = S 형태로 연결한다면, 간단하게 JK flip-flop을 구현할 수 있으며, 그 verilog 코드는 아래와 같다.

`timescale 1ns / 1ps

module JKlatch(

input J, K, E,

output Q, Q\_com

);

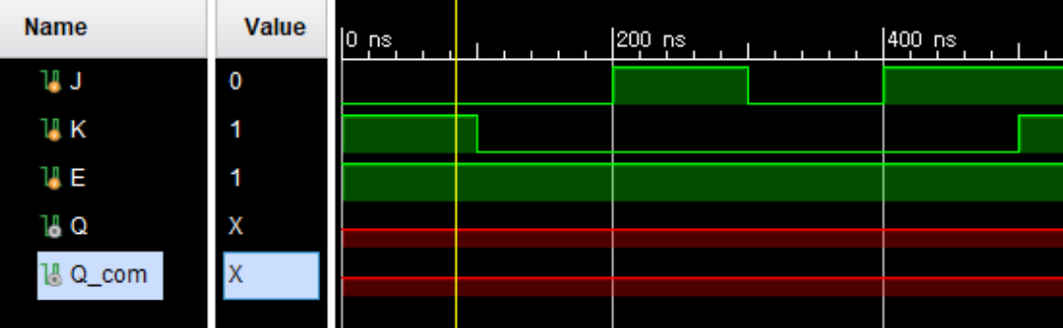
assign Q = ~(Q\_com|(K&Q&E));

assign Q\_com = ~(Q&|(J&Q\_com&E));

endmodule

<JK latch의 verilog 코드>

하지만 위 코드를 토대로 simulation을 진행했을 때 아래와 같이, Q, Q\_com 모두 항상 X가 나왔다.



<JK latch의 simulation 결과>

개인적인 생각으로는 Q, Q\_com이 input으로 들어가는 만큼, Q, Q\_com이 이전에 어떤 값을 갖고 있어야 한다는 생각을 했다. 따라서 simulation ~~.v 파일에 wire Q, Q\_com, assign Q = 1, Q\_com = 0; 형태를 시도해보기도 했으나, 이 문제를 해결할 수 없었다. 이를 해결하기 위해서 처음에 Q, Q\_com에 HIGH(1) / LOW(0)의 대입이 필요할 것 같은데, 이를 해결하는 방법을 아직 생각해내지 못했다.