11주차 결과보고서

전공: 신문방송학과 학년: 4학년 학번: 20191150 이름: 전현길

1. RS Flip-Flop의 결과 및 Simulation 과정에 대해서 설명하시오.

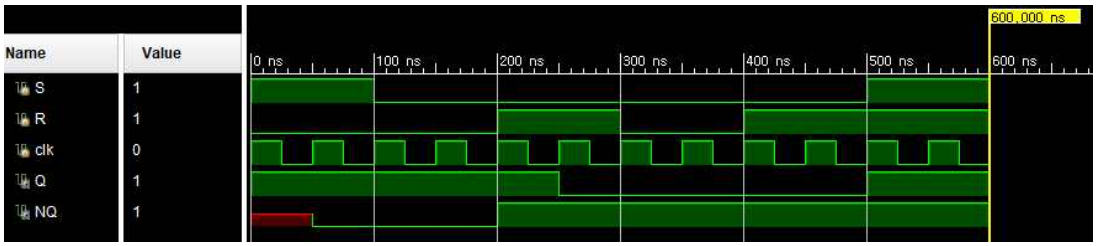
input			output	
input order	R	S	Q	~Q
1	0	1	1	0
2	0	0	1	0
3	1	0	0	1
4	0	0	0	1
5	1	0	0	1
6	1	1	X	X

present state	next state Q(t+1)			
	00	01	10	11
Q(t)	Q(t)	0	1	X
	Q(t)	0	1	X
	Q(t)	0	1	X
	Q(t)	0	1	X
	Q(t)	0	1	X
	Q(t)	0	1	X
	Q(t)	0	1	X
	Q(t)	0	1	X
	Q(t)	0	1	X
	Q(t)	0	1	X
	Q(t)	0	1	X
	Q(t)	0	1	X
	Q(t)	0	1	X
	Q(t)	0	1	X
	Q(t)	0	1	X
	Q(t)	0	1	X
	Q(t)	0	1	X

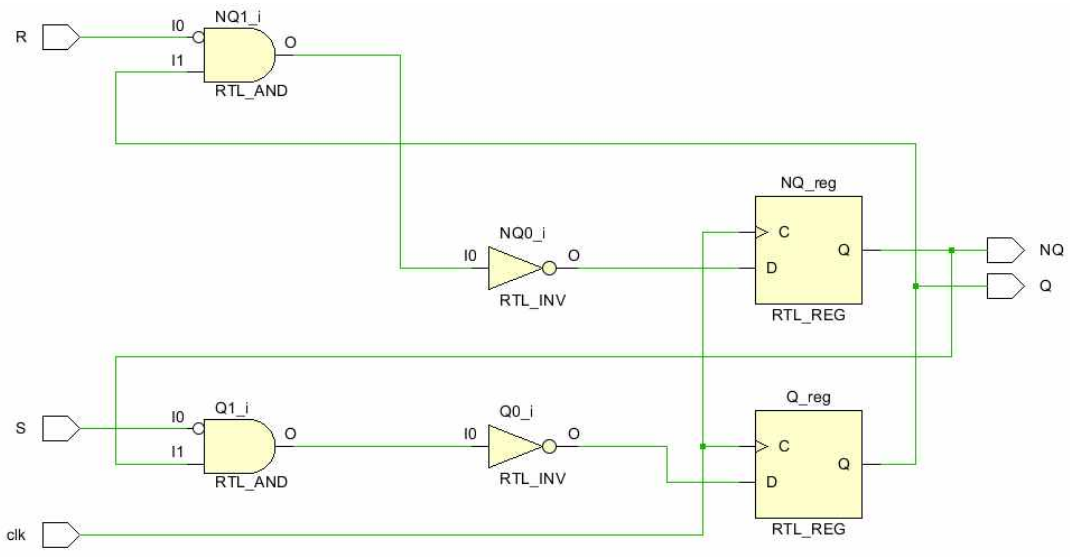
source code	testbench code

`timescale 1ns / 1ps
 	
module SRFF_NAND (
  input S, R, clk,
  output reg Q, NQ
);
 	
always@ (posedge clk) begin
  if (!clk) begin
  Q <= Q;
  NQ <= Q:
  end
  else begin
  Q <= ~(R | NQ);
  NQ <= ~(S | Q);
  end
end
 	
endmodule
	`timescale 1ns / 1ps
	
	module SRFF_NAND_tb;
	
	reg S, R, clk;
	wire Q, NQ;
	
	SRFF_NAND u_test(
	  .S (S), .R (R), .clk (clk),
	  .Q (Q), .NQ (NQ)
	);
	
	initial begin
	  S = 1'b0; R = 1'b0; clk = 1'b1;
	  forever #25 clk = ~clk;
	end
	
	initial begin
	  R = 1'b0; S = 1'b1; #100
	  R = 1'b0; S = 1'b0; #100
	  R = 1'b1; S = 1'b0; #100
	  R = 1'b0; S = 1'b0; #100
	  R = 1'b1; S = 1'b0; #100
	  R = 1'b1; S = 1'b1; #100
	  $finish;
	end
	
	endmodule

RS flip-flop NAND simultation



RS flip-flop NAND circuit design



SR flip-flop(=RS flip-flop)은 두 개의 입력과 두 개의 출력을 갖는 flip-flop으로, set과 reset 입력을 갖는다. SR flip-flop은 flip-flop의 현재 상태와 무관하게 set 입력만 활성화되면 다음 clock edge에서 flip-flop에 1이 저장되며, reset 입력만 활성화되면 다음 flip-flop에 0이 저장된다. 반대로 set, reset 입력이 모두 0이면 현재 상태가 유지된다. 단 두 입력이 동시에 1이 되면 flip-flop의 출력값은 정의할 수 없는 상태가 된다.

첫 번째 회로의 경우 NAND 회로를 사용하여 SR flip-flop을 구현하였고, 클록 주기를 50ns로 설정한 뒤 강의자료에 제시된 순서대로 simulation을 구성하였다. 또, always@ (posedge clk) 문으로 RS flip-flop을 정의하였으므로 클록의 rising-edge에서 상태가 변화한다.

simulation을 살펴보면, combinational logic circuit에서 그랬던 것처럼 입력값의 변화에 따라 출력값이 즉각적으로 변화하고 있지 않은 것을 알 수 있다. 이는 edge-triggered 구현의 특징 때문이다.

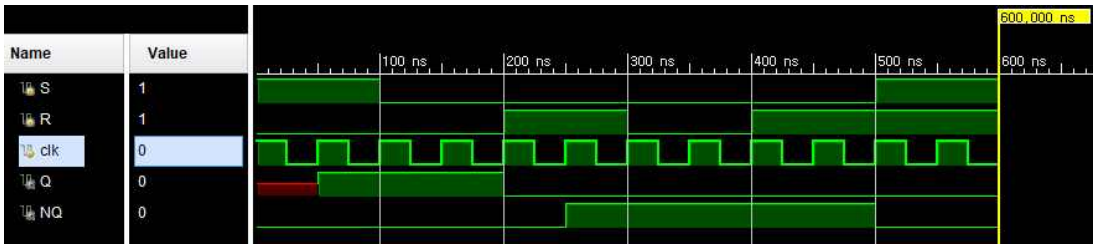
edge-triggered는 이상적으로는 클럭의 값이 변한 그 순간(rising-edge 또는 falling-edge)의 상태에 따라 출력을 생성하지만, 실제로 **입력을 반영하기 위해서는 edge 이전에 어느 정도 입력이 유지되는 시간이 필요**하다. 이러한 지연 시간을 **setup time delay**라고 부른다. 반대로 **edge 후 출력을 유지하기 위해 필요한 최소 시간**을 **hold time delay**라고 부른다.

시뮬레이션의 경우 입력이 변하는 그 순간 클럭이 변화하므로 delay가 전혀 주어지지 않았다. 이 때문에 출력값의 변화가 다음 번째 rising-edge에서 일어나는 것을 볼 수 있다. setup time delay 문제를 해결하는 것은 SR flip-flop과 같은 간단한 회로에서는 그리 어렵지 않다. **적절히 타이밍을 조절하여 setup time delay와 hold time delay의 margin 내에 데이터가 입력되도록 하면** 된다. 단 회로가 복잡해지면 복잡해질수록 이러한 타이밍 계산은 훨씬 더 어려워질 수 있다.

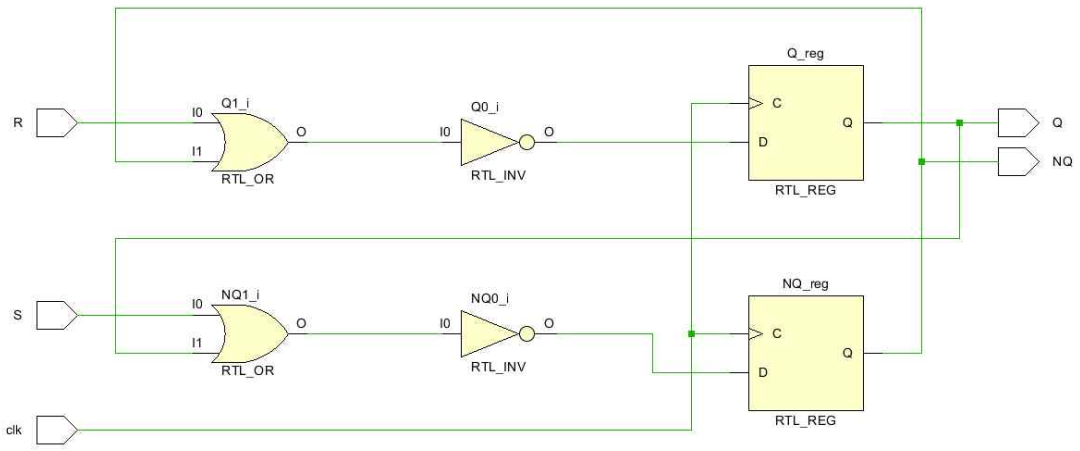
delay에 관련된 문제를 제외하면, set이 활성화될 때 Q가 1로, reset이 활성화될 때 Q가 0으로 변화하고 set, reset이 모두 0이라면 값이 유지되므로 SR flip-flop의 구현이 성공적으로 이루어졌음을 볼 수 있다.

source code	testbench code
`timescale 1ns / 1ps module SRFF_NOR (input S, R, clk, output reg Q, NQ); always@ (posedge clk) begin if (!clk) begin Q <= Q; NQ <= NQ: end else begin Q <= ~(S | NQ); NQ <= ~(S | Q); end end endmodule	`timescale 1ns / 1ps module SRFF_NOR_tb; reg S, R, clk; wire Q, NQ; SRFF_NOR u_test(.S (S), .R (R), .clk (clk), .Q (Q), .NQ (NQ)); initial begin S = 1'b0; R = 1'b0; clk = 1'b1; forever #25 clk = ~clk; end initial begin R = 1'b0; S = 1'b1; #100 R = 1'b0; S = 1'b0; #100 R = 1'b1; S = 1'b0; #100 R = 1'b0; S = 1'b0; #100 R = 1'b1; S = 1'b0; #100 R = 1'b1; S = 1'b1; #100 $finish; end endmodule

RS flip-flop NOR simultation



RS flip-flop NOR circuit design



두 번째 회로의 경우 AND-NOR 방식으로 SR flip-flop을 구성하였다. 이

외에는 NAND gate에서 발생했던 rising-edge 문제가 반대 방향으로(NAND gate로 구현한 회로가 한 clock cycle만큼 Q가 느리게 동작했던 것과 달리, NOR gate는 한 clock cycle만큼 NQ가 느리게 동작) 발생하는 것을 볼 수 있다. 하지만 정상적인 입력 내에서 SR flip-flop의 동작을 잘 구현하고 있으므로 성공적으로 구현되었다고 볼 수 있다.

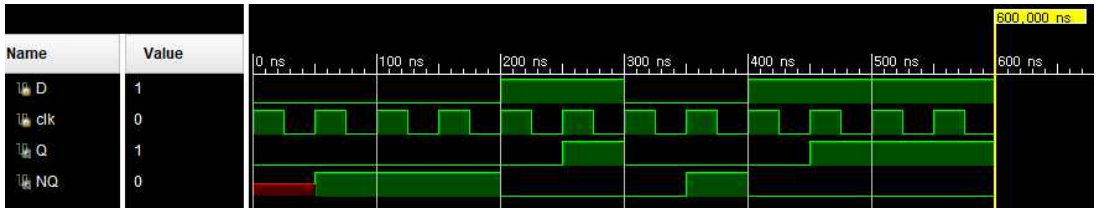
2. D Flip-Flop의 결과 및 Simulation 과정에 대해서 설명하시오.

input		output	
input order	D	Q	~Q
1	0	0	1
2	0	0	1
3	1	1	0
4	0	0	1
5	1	1	0
6	1	1	0

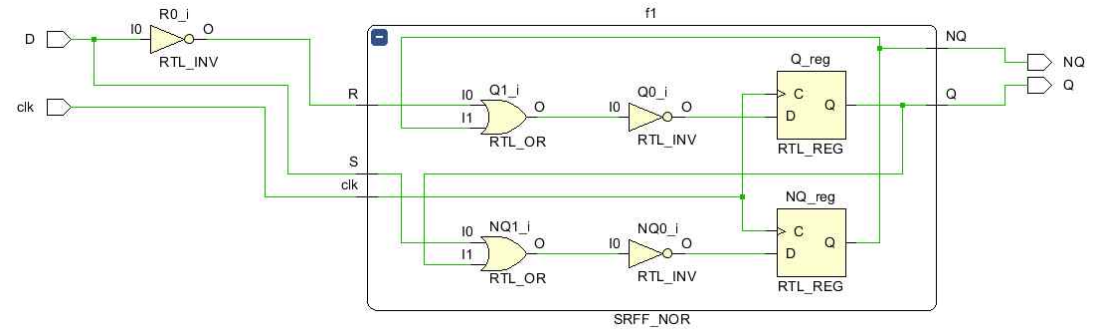
Q(t)	Q(t+1)	
	input D	
	0	1
Q(t)	0	1

source code	testbench code
`timescale 1ns / 1ps module SRFF_NOR (input S, R, clk, output reg Q, NQ); always@ (posedge clk) begin Q <= ~(R | NQ); NQ <= ~(S | Q); end endmodule module DFF (input D, clk, output Q, NQ); SRFF_NOR f1(.S (D), .R (~D), .clk (clk), .Q (Q) , .NQ (NQ)); endmodule	`timescale 1ns / 1ps module DFF_tb; reg D, clk; wire Q, NQ; DFF u_test(.D (D), .clk (clk), .Q (Q), .NQ (NQ)); initial begin D = 1'b0; clk = 1'b1; forever #25 clk = ~clk; end initial begin D = 1'b0; #100 D = 1'b0; #100 D = 1'b1; #100 D = 1'b0; #100 D = 1'b1; #100 D = 1'b1; #100 $finish; end endmodule

D flip-flop simultation



D flip-flop circuit design



간단한 SR flip-flop은 S, R 입력을 통해 상태를 활성화하거나 비활성화한다. 이 **SR flip-flop의 Set, Reset 입력에 각각 데이터와 데이터의 역수를 입력하면** 클럭이 활성화되었을 때는 데이터를 저장하고, 클럭이 비활성화되었을 때는 저장된 데이터를 유지하는 **D flip-flop을 구현할 수 있다.**

verilog 회로 구현의 경우 SR flip-flop 모듈을 재활용하여 D, D'를 각각 S, R 입력에 입력함으로써 구현했다. testbench code 역시 SR flip-flop에서 그랬듯이 클록 주기를 50ns로 설정한 뒤 강의자료에 제시된 순서대로 simulation을 구성하였다. 내부에 구현된 SR flip-flop이 always@ (posedge clk) 문으로 정의되어 있으므로 D flip-flop 역시 클록의 rising-edge에서 상태가 변화한다.

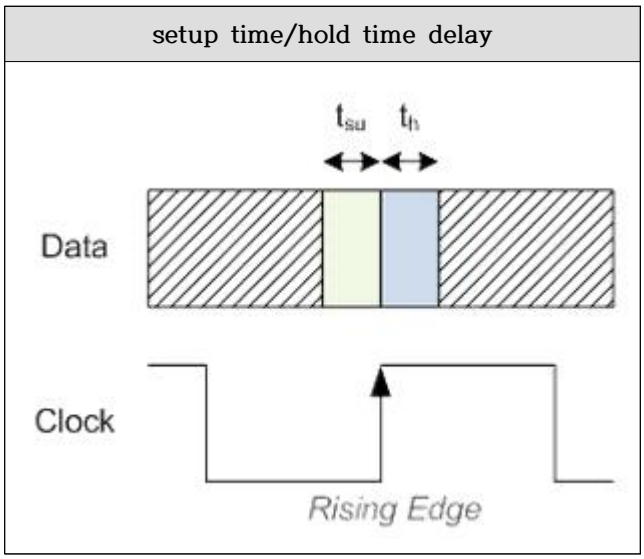
앞서 말한 setup-time delay 문제에 의해 D flip-flop 역시도 한 clock cycle씩 D 입력에 따른 출력값의 반영이 늦어지고 있다. 단 이외에는 D flip-flop의 상태표에 따른 동작이 나타나므로 정상적으로 구현되었다고 볼 수 있다.

3. 결과 검토 및 논의 사항

이번 실험에서는 NAND gate, NOR gate를 이용한 SR flip-flop, SR flip-flop을 이용한 D flip-flop을 Verilog 코드로 구현해 보았으며, FPGA 보드를 활용하여 실제 출력 결과를 확인해 보았다. 실험 결과, setup-time delay에 의한 타이밍 문제를 제외하곤 구현을 통해 기대되었던 동작이 정상적으로 이루어졌음을 확인했다.

이번 실험 때 처음으로 always@ (posedge clk) 방식의 구현을 시도해 보았는데, 생각보다 신경써야 할 지점이 많았고 다른 구현할 수 있는 방식도 많았다. 이번 코드를 작성할 때에는 어느 정도 NAND gate를 통한 실제 구현을 직접 고려하며 코드를 짰지만, 보다 추상적으로 코드를 작성하더라도 잘 구현되는 것을 볼 수 있었다. verilog 코드를 작성하는 다양한 방식에 대해 배울 수 있었다.

4. 추가 이론 조사 및 작성



edge-triggered 방식으로 구현된 디지털 회로가 rising edge가 일어나는 시점에 즉시 입력값을 반영하여 즉시 출력할 수 있다면 좋겠지만, 현실에서 edge-triggered circuit을 구현할 때에는 두 가지 종류의 delay를 고려해야 한다. **setup time delay**와 **hold time delay**이다.

setup time delay란 **edge 전, 입력을 받아들이기 위해 필요한 최소 시간을 의미**하며 hold time delay란 **edge 후, 출력을 유지하기 위해 필요한 최소 시간을 의미**한다. 특정한 데이터에 대해서 원하는 출력(=정상적인 출력)을 얻기

위해서는 clock 전후로 두 delay 동안 데이터가 동일한 상태로 유지되어야 한다.

이번 testbench 코드를 설계할 때는 delay 문제를 논의하기 위해 일부러 setup time violation이 발생하도록 설계했으나, 이번에 설계했던 것과 같은 간단한 flip-flop의 경우 클록 주기를 유지하고 첫 번째 입력 변화 시간을 조금만 앞으로 당기는 것만으로도 문제를 해결할 수 있다.