11주차 예비보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20221549 이름: 김효림

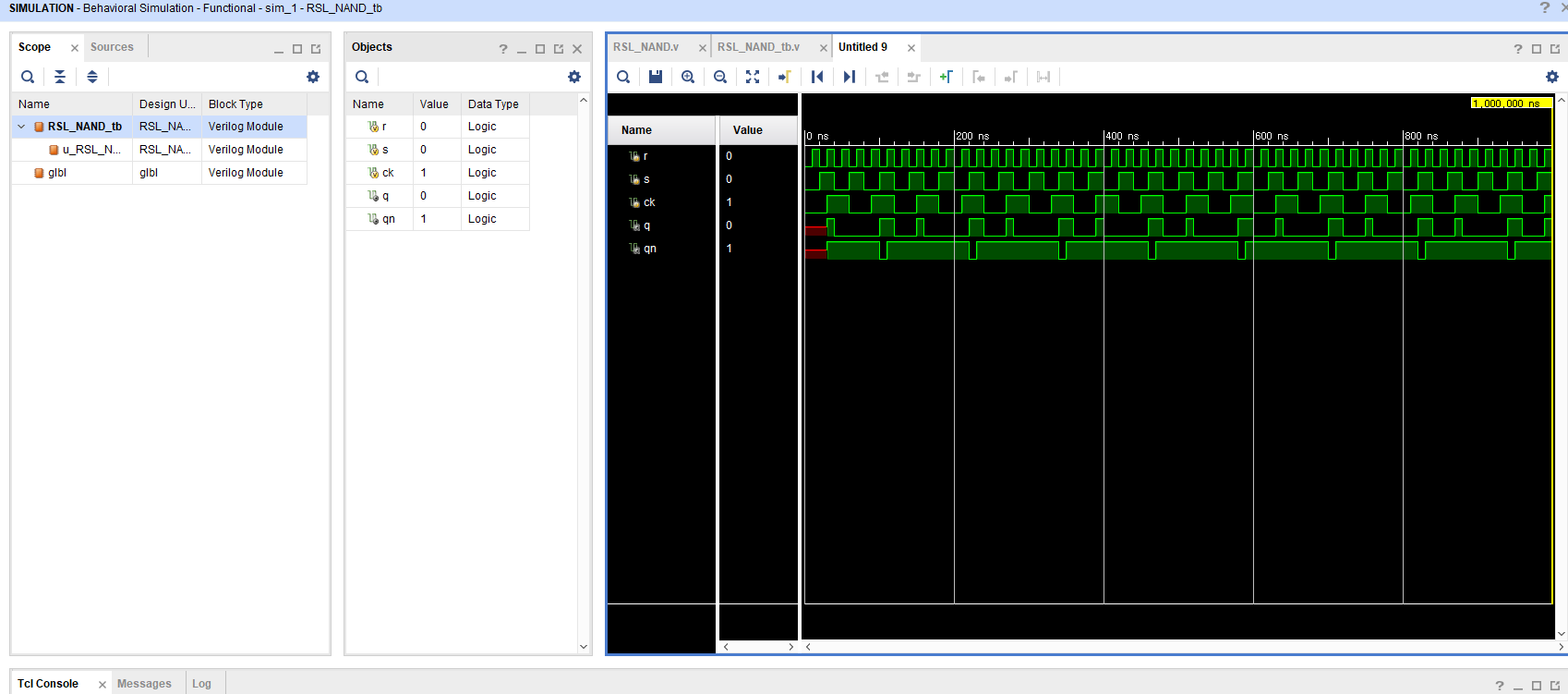
**1.**

.................

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

      (verilog source, simulation 출력 예시, table 등의 과정 상세히 적을것)

※ NAND로 구성된 RS Flip-Flop과 NOR로 구성된 RS Flip-Flop 2가지 모두 작성할 것

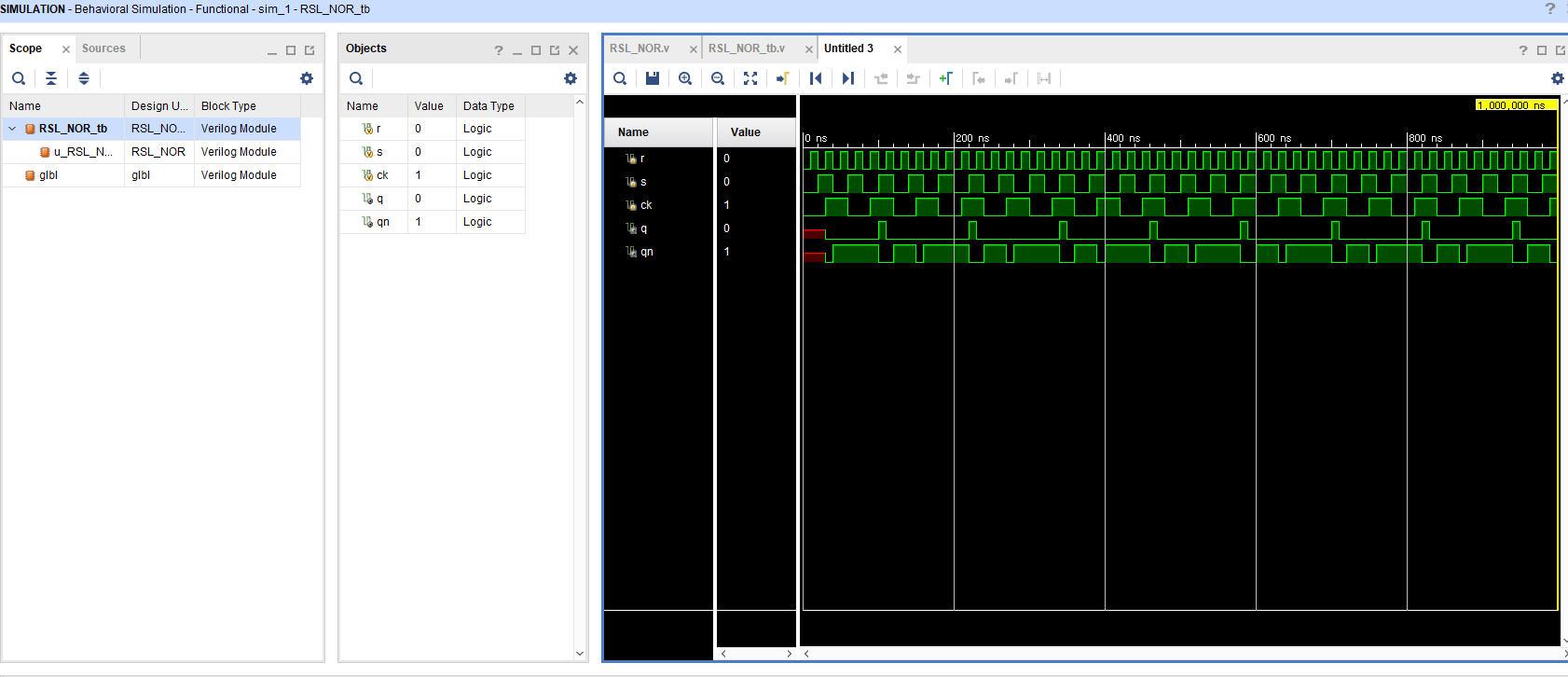


NAND로 구성한 RS Flip-Flop의 simulation 결과

|  |  |
| --- | --- |
| RSL\_NAND.v | RSL\_NAND\_tb.v |
| module RSL\_NAND(  input s, r, ck,  output q, qn  );    wire a,b;    assign a = s & ck;  assign b = r & ck;    assign qn = ~( ~b & q);  assign q = ~( ~a & qn);  endmodule | module RSL\_NAND\_tb;  reg r,s,ck;  wire q, qn;  RSL\_NAND u\_RSL\_NAND(  .r(r),  .s(s),  .ck(ck),  .q(q),  .qn(qn)  );  initial begin  r = 1'b0;  s = 1'b0;  ck = 1'b0;  end  always@(r or s or ck)begin  r <= #10 ~r;  s <= #20 ~s;  ck <= #30 ~ck;  end  endmodule |

NAND로 구성한 RS Flip-Flop의 Verilog Source code

RS Flip flop의 회로도를 확인하며, wire를 통해 중간 계산 값을 선언해 주고, 그 결과값과 입력값을 NAND연산하여 결과값을 도출할 수 있었다.



NOR gate로 구성된 RS Flip Flop의 simulation 결과

|  |  |
| --- | --- |
| RSL\_NOR.v | RSL\_NOR\_tb.v |
| module RSL\_NOR(  input s,r,ck,  output q, qn  );    wire a,b;    assign a = s & ck;  assign b = r & ck;    assign q = ~(b || qn);  assign qn = ~(a || q);    endmodule | module RSL\_NOR\_tb;  reg r,s,ck;  wire q, qn;  RSL\_NOR u\_RSL\_NOR(  .r(r),  .s(s),  .ck(ck),  .q(q),  .qn(qn)  );  initial begin  r = 1'b0;  s = 1'b0;  ck = 1'b0;  end  always@(r or s or ck)begin  r <= #10 ~r;  s <= #20 ~s;  ck <= #30 ~ck;  end  endmodule |

NOR gate로 구성된 RS flip flop의 Verilog source code

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Input | | | Output | |
| 입력 순서 | 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 |

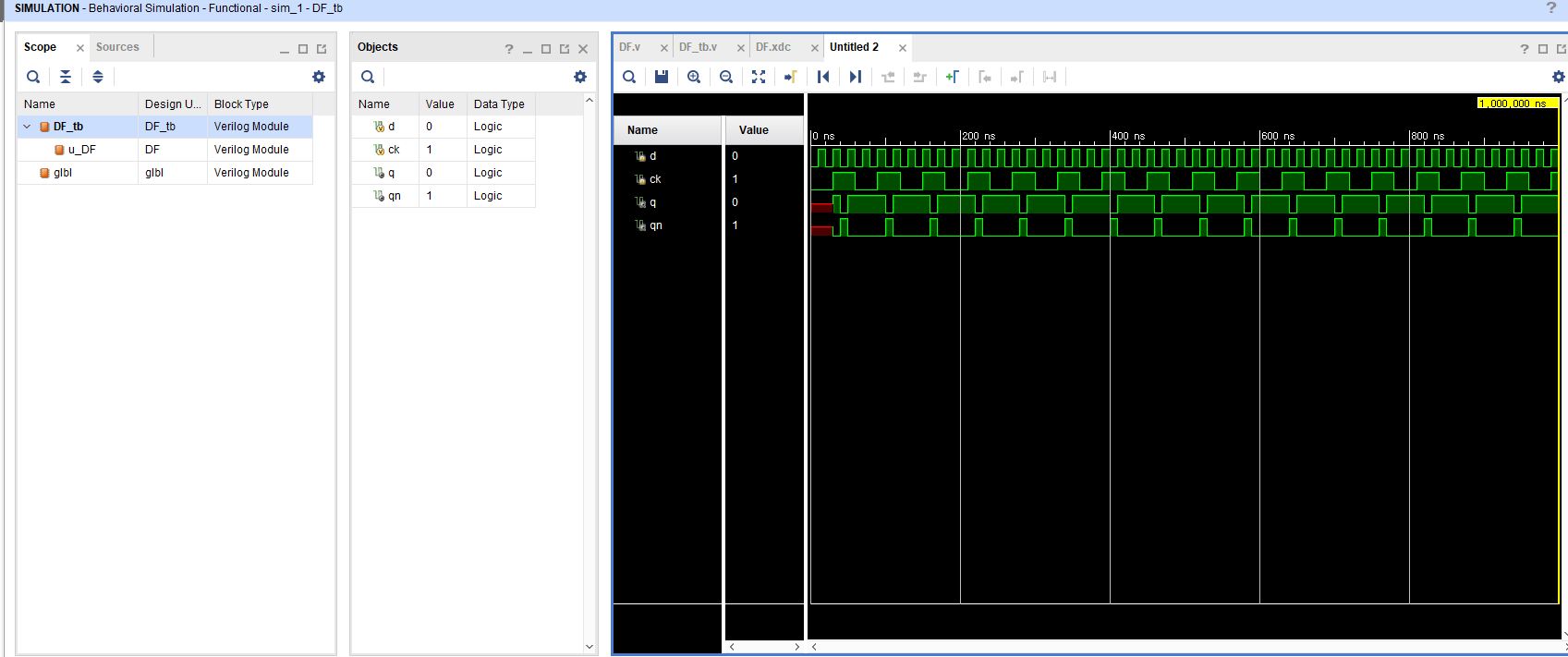
RS Flip Flop의 truth table 결과

RS Flip flop의 회로도를 확인하며, wire를 통해 중간 계산 값을 선언해 주고, 그 결과값과 입력값을 NOR연산하여 결과값을 도출할 수 있었다.

................

**2.**

.......................



 D flip flop의 simulation 결과

|  |  |
| --- | --- |
| DF.v | RSL\_NOR\_tb.v |
| module DF(  input d,ck,  output q, qn  );    wire a,b;    assign a = ~(d&ck);  assign b = ~(ck & ~d);    assign q = ~(a & qn);  assign qn = ~(b & q);  endmodule | module DF\_tb;  reg d,ck;  wire q,qn;  DF u\_DF(  .d(d),  .ck(ck),  .q(q),  .qn(qn)  );  initial begin  d = 1'b0;  ck = 1'b0;  end  always@(d or ck)begin  d <= #10 ~d;  ck <= #30 ~ck;  end  endmodule |

D flip-flop의 Verilog source code

|  |  |  |  |
| --- | --- | --- | --- |
| Input | | Output | |
| 입력 순서 | 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 |

D Flip flop의 truth table

D flip flop의 회로도를 확인하여, wire를 통해 중간 과정을 저장하고 그 중간 결과값을 input과 NAND 연산을 통해 계산하여 결과값을 구할 수 있었다.

.........................

**3.**

.......................

시뮬레이션 결과를 통해 clock에 아무런 값이 할당되지 않았을 때는 output의 값을 확인할 수 없다는 점을 육안으로 확인할 수 있었다. RS flip flop에서 NOR gate로 구현을 하든, NAND gate로 구현을 하든 같은 simulation 결과가 나옴을 확인하여, 두 회로는 같은 기능을 한다는 걸 확인할 수 있었다.

.........................

**4.**

.......................

플립에는 입력값과 clock 외에도, preset과 clear라는 것을 추가할 수 있다. Preset과 clear는 클럭과 상관없이 입력되는 비동기 신호로, preset에 1이 입력되면 1이 출력된다. 일종의 초기화 동작이라고 할 수 있으며, 클럭을 우선하는 비동기식 입력단자로 처음 전원을 켰을 때 플립플롭의 상태를 원하는 데이터로 초기화할 수 있다. 즉, 모든 플립플롭의 출력을 unknown 상태에서 known 상태로 가게 하는데 사용된다. 이 둘은 플립플롭의 다른 어떤 입력보다도 최우선 순위를 가진다.

..........................