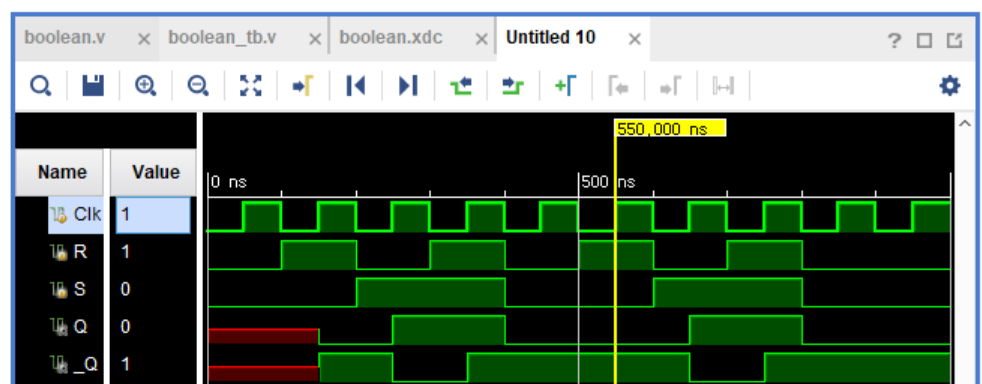
11주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20231609 이름: 정희선

**1. RS flip-flop**

**1.1) NAND gate RS flip-flop**

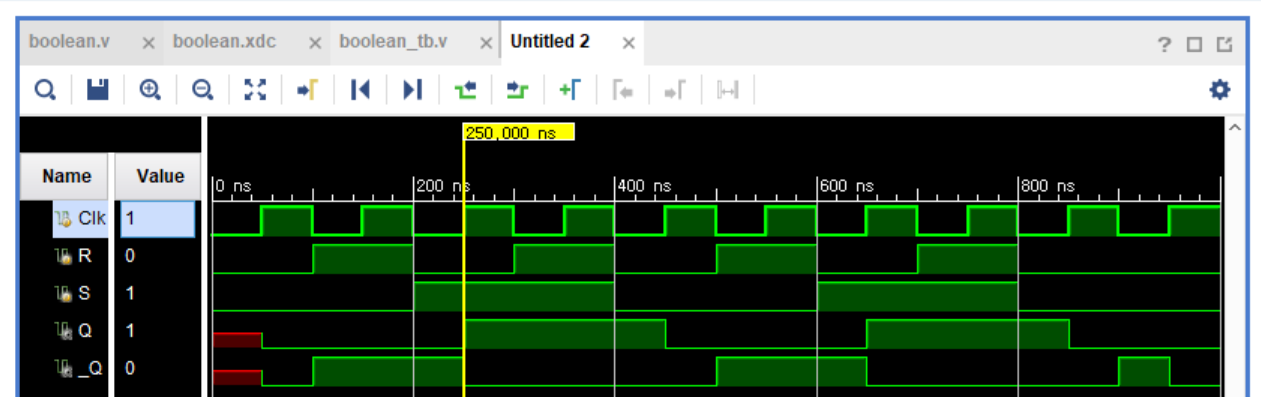
|  |  |
| --- | --- |
| **Design Source** | **Test Bench** |
| *`timescale 1ns / 1ps*  *module boolean(*  *input Clk, R, S,*  *output Q,*  *output \_Q*  *);*  *wire out1;*  *wire out2;*  *assign out1 = ~(R&Clk);*  *assign out2 = ~(S&Clk);*  *assign Q = ~(out2&\_Q);*  *assign \_Q = ~(out1&Q);*  *endmodule* | *`timescale 1ns / 1ps*  *module boolean\_tb;*  *reg Clk,R,S;*  *wire Q, \_Q;*  *boolean u\_boolean(*  *.Clk(Clk),*  *.R(R),*  *.S(S),*  *.Q(Q),*  *.\_Q(\_Q)*  *);*  *initial begin*  *Clk = 1'b0;*  *forever #50 Clk = ~Clk;*  *end*  *initial begin*  *R = 1'b0;*  *S = 1'b0;*  *#100 R = 1'b1; S = 1'b0; // Reset 상태*  *#100 R = 1'b0; S = 1'b1; // Set 상태*  *#100 R = 1'b1; S = 1'b1; // 금지 상태*  *#100 R = 1'b0; S = 1'b0; // 유지 상태*  *#100 R = 1'b1; S = 1'b0; // Reset 상태*  *#100 R = 1'b0; S = 1'b1; // Set 상태*  *#100 R = 1'b1; S = 1'b1; // 금지 상태*  *#100 R = 1'b0; S = 1'b0; // 유지 상태*  *end*  *initial begin*  *#1000*  *$finish;*  *end*  *endmodule* |



우선 RS 플립플롭은 두 개의 입력(Reset과 Set)을 통해 출력을 제어하는 플립플롭이다. 이 경우 NAND 게이트를 기반으로 구현한 RS 플립플롭이다. 두 개의 NAND 게이트를 교차 연결하여 회로를 구성한다.

**1.2) NOR gate RS flip-flop**

|  |  |
| --- | --- |
| **Design Source** | **Test Bench** |
| *`timescale 1ns / 1ps*  *module boolean(*  *input Clk, R, S,*  *output Q,*  *output \_Q*  *);*  *wire out1;*  *wire out2;*  *assign out1 = ~(R | ~Clk); // Clk와 R의 입력 조합*  *assign out2 = ~(S | ~Clk); // Clk와 S의 입력 조합*  *assign Q = ~(out2 | \_Q); // NOR 게이트로 Q 계산*  *assign \_Q = ~(out1 | Q); // NOR 게이트로 \_Q 계산*  *endmodule* | *`timescale 1ns / 1ps*  *module boolean\_tb;*  *reg Clk,R,S;*  *wire Q, \_Q;*  *boolean u\_boolean(*  *.Clk(Clk),*  *.R(R),*  *.S(S),*  *.Q(Q),*  *.\_Q(\_Q)*  *);*  *initial begin*  *Clk = 1'b0;*  *forever #50 Clk = ~Clk;*  *end*  *initial begin*  *R = 1'b0;*  *S = 1'b0;*  *#100 R = 1'b1; S = 1'b0; // Reset 상태*  *#100 R = 1'b0; S = 1'b1; // Set 상태*  *#100 R = 1'b1; S = 1'b1; // 금지 상태*  *#100 R = 1'b0; S = 1'b0; // 유지 상태*  *#100 R = 1'b1; S = 1'b0; // Reset 상태*  *#100 R = 1'b0; S = 1'b1; // Set 상태*  *#100 R = 1'b1; S = 1'b1; // 금지 상태*  *#100 R = 1'b0; S = 1'b0; // 유지 상태*  *end*  *initial begin*  *#1000*  *$finish;*  *end*  *endmodule* |



위 코드는 NOR 게이트를 교차 연결하여 구성한 RS 플립플롭이다. NAND 기반 RS 플립플롭과 입력 논리의 차이만 존재하고 동작은 동일하다.

두 개의 RS 플립플롭의 시뮬레이션 결과를 분석해보면,

1. R=1, S=0일 때, reset 상태가 되어 Q=0이 설정된다.
2. R=0, S=1일 때, set 상태가 되어 Q=1이 설정된다.
3. R=1, S=1일 때, 금지 상태가 되어 출력이 정의되지 않는다. (모순 발생)
4. R=0, S=0일 때, 유지 상태가 되어 Q가 이전의 상태를 유지한다.

* Truth table

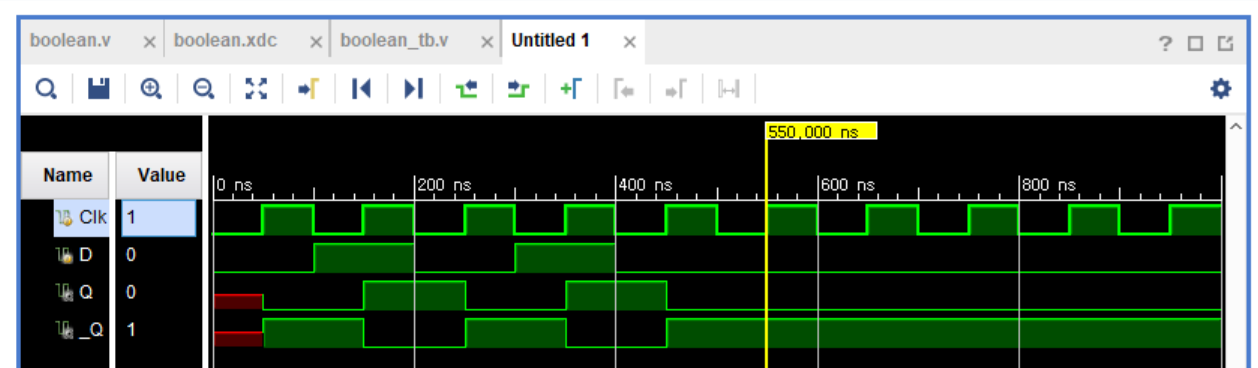
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| INPUT | | | OUTPUT | |
| 입력 | R | S | Q | ~Q |
| 1 | 0 | 1 | 1 | 0 |
| 2 | 0 | 0 | 1 | 0 |
| 3 | 1 | 0 | 0 | 1 |
| 4 | 0 | 0 | 0 | 1 |
| 5 | 1 | 0 | 0 | 1 |
| 6 | 1 | 1 | X | X |

* State table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Present state Q | R | S | Next Q | 동작 설명 |
| 0 | 0 | 0 | 0 | 유지 |
| 0 | 0 | 1 | 1 | Set |
| 0 | 1 | 0 | 0 | Reset |
| 0 | 1 | 1 | X | 금지 상태 |
| 1 | 0 | 0 | 1 | 유지 |
| 1 | 0 | 1 | 1 | Set |
| 1 | 1 | 0 | 0 | Reset |
| 1 | 1 | 0 | X | 금지 상태 |

**2. D filp-flop**

|  |  |
| --- | --- |
| **Design Source** | **Test Bench** |
| *`timescale 1ns / 1ps*  *module boolean(*  *input Clk, D,*  *output Q, \_Q*  *);*    *wire out1;*  *wire out2;*    *assign out1 = ~D&Clk;*  *assign out2 = D&Clk;*  *assign Q = ~(out1|\_Q);*  *assign \_Q = ~(out2|Q);*    *endmodule* | *`timescale 1ns / 1ps*  *module boolean\_tb;*  *reg Clk, D;*  *wire Q, \_Q;*  *boolean u\_boolean (*  *.Clk (Clk ),*  *.D (D ),*  *.Q (Q ),*  *.\_Q (\_Q )*  *);*  *initial begin*  *Clk = 1'b0;*  *D = 1'b0;*  *end*  *always Clk = #50 ~Clk;*  *always D = #100 ~D;*  *initial begin*  *#1000*  *$finish;*  *end*  *endmodule* |



D 플립플롭은 클럭 신호의 상승 엣지에서 입력 D 값을 출력으로 전달하는 디지털회로이다. 시뮬레이션 결과, 클럭 신호의 상승 엣지에서만 입력 D가 출력 Q에 반영되는 것을 확인할 수 있다. 클럭의 하강 엣지에서는 출력 Q가 유지된다. \_Q는 Q의 보수 값을 나타낸다. 클럭 상승 엣지 이후에만 상태가 업데이트 되므로, 입력 D의 변화는 결과에 즉각적인 영향을 미치지 않는다.

시뮬레이션 결과,

입력 D=0인 경우, 클럭 상승 엣지에서 Q=0, ~Q=1으로 설정된다.

입력 D=1인 경우, 클럭 상승 엣지에서 Q=1, ~Q=0으로 설정된다.

* Truth table

|  |  |  |  |
| --- | --- | --- | --- |
| INPUT | | OUTPUT | |
| 입력 | 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 |

* State table

|  |  |  |
| --- | --- | --- |
| Present state Q | D | Next Q |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**3.**

RS 플립플롭은 Set과 Reset 동작을 기반으로 출력을 제어하는 기본적인 메모리 요소로, 디지털 회로 설계에서 중요한 역할을 한다. NAND 게이트와 NOR 게이트를 기반으로 구현할 수 있고, 각 구현 방식에서 동작 원리와 결과가 동일하게 유지된다. 시뮬레이션 결과는 truth table과 일치한다. 이러한 RS 플립플롭은 기본적인 상태 저장 장치로서 역할을 수행하지만 금지 상태가 존재한다는 단점이 존재하고, 이를 해결하기 위해 JK 플립플롭이나 D 플립플롭과 같은 구조가 개발되었다.

D 플립플롭은 RS 플립플롭의 금지 상태 단점을 보완해 줄 수 있다. RS 플립플롭의 입력을 단일 입력으로 단순화하고, 클럭 신호를 사용하여 데이터의 저장 시점을 명확히 제어한다. 시뮬레이션 결과는 truth table과 일치했다. 이때 D 입력이 변하더라도 클럭 상승 엣지 이전에는 출력 상태에 영향을 미치지 않아 D 플립플롭이 동기식으로 작동하는 것을 보장하였다. D 플립플롭은 금지 상태가 존재하지 않고, 입력 데이터의 저장과 클럭 동기화를 통해 RS 플립플롭에 비해 안정적인 상태 저장장치로 기능한다.

**4.**

RS 플립플롭과 D 플립플롭의 비교 : RS 플립 플롭은 금지 상태로 인해 단순 상태 저장에 제한적이다. 하지만 D 플립플롭은 이를 개선하여 실질적 디지털 설계에 적합한 방식으로 동작한다. 또, D 플립플롭은 RS 플립플롭을 기반으로 하고 있고, 클럭 신호와 입력 논리를 추가한 구조이므로 RS 플립플롭보다 높은 안정성과 확장성을 제공한다.

플립플롭의 전력 효율성: 전력 효율성은 플립플롭 설계에서 매우 중요한 요소이다. 우선 플립플롭이 동작하면서 출력 상태가 변화할 때 소모되는 전력을 스위칭 전력이라고 부른다. 스위칭 전력은 클럭 주파수와 플립플롭 게이트 수에 의해 결정된다. 이를 줄이기 위해 불필요한 상태 변화를 최소화하도록 설계한다. 다음으로 회로가 동작하지 않고 대기 상태에 있을 때 소모되는 전력을 정적 전력이라고 한다. 이는 플립플롭의 누설 전류에 의해 결정된다. 저전력 기술이나 전력 관리 블록을 통해 이 전력을 죄소화한다.