2018년 2학기 컴퓨터공학실험Ⅱ
CSE3016-05반 11주차 결과 보고서

학번: 20171665
이름: 이 선 호

2018. 11. 30

목 차

Ⅰ RS Flip-Flop

1. NOR 게이트 사용

1) Verilog 코딩

inv.v

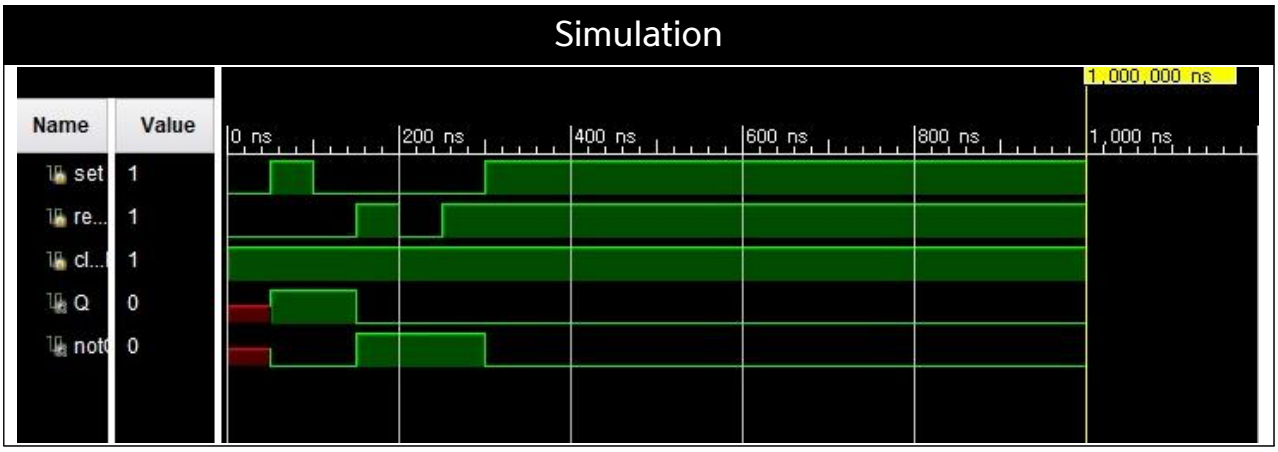
```
`timescale 1ns / 1ps

module inv(s,r, cp, q, nq);

input s, r, cp;
wire set, reset, clockpulse, set2, reset2;
output q, nq;

nor(reset, r, r);
nor(set, s, s);
nor(clockpulse, cp, cp);

nor(reset2, reset, clockpulse);
nor(set2, set, clockpulse);

nor(q, reset2, nq);
nor(nq, set2, q);

endmodule
```

inv_tb.v

```
`timescale 1ns / 1ps

module inv_sim;

reg set,reset, clock;
wire  Q, notQ;

inv rs_inv(
      .s(set), .r(reset), .cp(clock), .q(Q), .nq(notQ)
      );
```

```
initial begin
        set = 1'b0;
        reset = 1'b0;
        clock = 1'b1;
        #50 set = ~set;
        #50 set = ~set;
        #50 reset = ~reset;
        #50 reset = ~reset;
        #50 reset = ~reset;
        #50 set = ~set;
end

initial begin
                #1800
                $finish; end

endmodule
```

3) Simulation 결과 (Timing Diagram)



4) 과정

예비 보고서에서 조사한 바와 같이 RS 플립플롭은 S와 R이 모두 0일 때 Q와 Q'가 0이 되고, S가 1이고 R이 0일 때 Q가 1, Q'는 0으로 되어야 한다. S가 0이고 R이 1이면 Q가 0, Q'가 1로 설정되어야 하고, S와 R이 1일 때의 Q와 Q'의 값은 고려하지 않는다. 출력되는 Q와 Q'값이 각각 S와 R과 피드백이 되도록 하는 것이 RS 플립플롭의 특징이었다. NOR 게이트를 사용한 RS 플립플롭의 회로도를 확인하면 S는 Q와 NOR 연산이 되어 Q'를 결과로 출력하고, R은 Q'와 NOR 연산이 되어 Q를 결과로 출력한다.

디자인 소스 코드를 짜기 위해 우선 input과 output을 정해야 한다. Clock을 사용한

RS 플립플롭을 디자인해야 하므로 R, S, Clock을 input으로 설정해야 한다. 각각 r, s, 그리고 clockpulse로 정했다. 그리고 Q와 Q'를 출력하는 output으로 각각 q와 nq를 설정했다. RS 플립플롭 내 모든 회로에서 NOR 게이트를 사용해야 하므로 clockpulse를 r, s와 각각 nor 게이트로 연결하여 각각 reset, set wire로 출력되도록 한다. 그런데 NOR 게이트로 연결이 되므로 예상했던 값의 보수가 되어서 이를 다시 보수 처리 해주는 작업이 필요하다. 그래서 reset과 set을 각각 자기 자신과 NOR 게이트로 연결해준 값을 reset2와 set2 wire에 결과로 출력되도록 한다. reset2와 set2가 Clock이 활성화 되었을 때 우리가 입력하고자 한 정상적인 R과 S의 값을 가지므로 앞에서 설명한 것처럼 reset2와 set2를 각각 nq, q와 NOR 게이트 연산을 하여 다음 상태의 q, nq값을 출력하도록 한다.

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	0	0

Simulation을 통해 RS 플립플롭이 정상적으로 구동되는지 확인하기 위해 R과 S 값을 달리한 입력 경우마다 순서를 부여하여 각 입력들에 따라 출력이 어떻게 되는지 확인했다. 이를 정리한 진리표는 왼쪽 표와 같다.

왼쪽 표의 순서처럼 initial 블록에 50ns가 지날 때마다 그에 맞게 R 또는 S 값을 적절히 설정하여 그에 따른 Q와 Q'값이 어떻게 출력되는지 확인했다. Simulation 결과는 위의 Timing Diagram에서 보여준다.

Timing Diagram의 첫 번째 구간(0~50ns)에서 Q와 Q'값이 X로 나오는 이유는 이전 구간이 존재하지 않기 때문에 저장하고 있던 Q와 Q'값이 존재하지 않아 다음 상태의 Q와 Q'를 계산할 때 참고하지 못하므로 알 수 없다고 표시가 뜨는 것이다.

2. NAND 게이트 사용

1) Verilog 코딩

inv.v

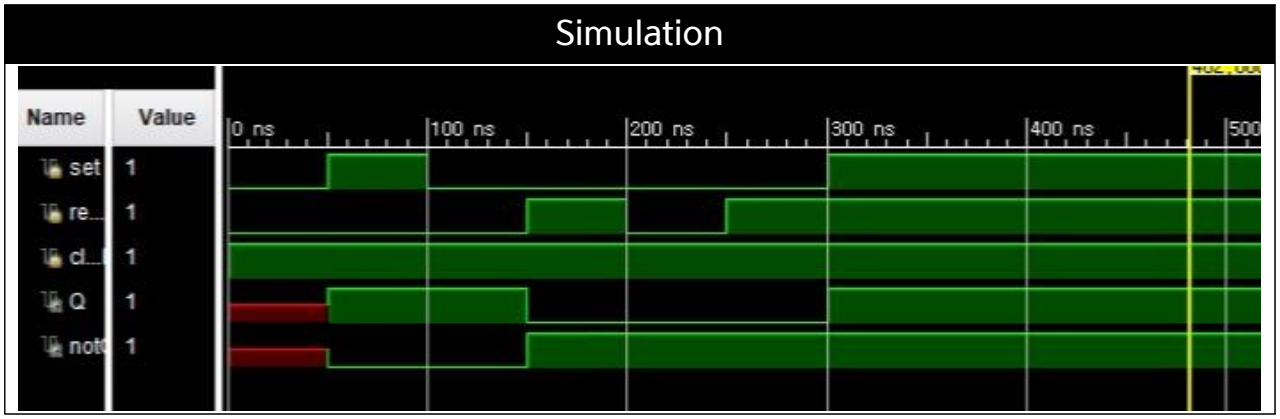
```
`timescale 1ns / 1ps

module inv(s,r, cp, q, nq);

input s, r, cp;
wire set, reset;
```

```
output q, nq;

nand(reset, r, cp);
nand(set, s, cp);

nand(nq, reset, q);
nand(q, set, nq);

endmodule
```

inv_tb.v

```
`timescale 1ns / 1ps

module inv_sim;

reg set,reset, clock;
wire  Q, notQ;

inv rs_inv(
    .s(set), .r(reset), .cp(clock), .q(Q), .nq(notQ)
    );

initial begin
        set = 1'b0;
        reset = 1'b0;
        clock = 1'b1;
        #50 set = ~set;
        #50 set = ~set;
        #50 reset = ~reset;
        #50 reset = ~reset;
        #50 reset = ~reset;
        #50 set = ~set;
end

initial begin
            #800
            $finish; end

endmodule
```

3) Simulation 결과 (Timing Diagram)



4) 과정

NOR 게이트를 사용한 RS 플립플롭과 마찬가지로 AND 게이트만을 사용한 플립플롭에서도 S와 R이 모두 0일 때 Q와 Q'가 0이 되고, S가 1이고 R이 0일 때 Q가 1, Q'가 0으로 되어야 한다. S가 0이고 R이 1이면 Q가 0, Q'가 1로 설정되어야 하고, S와 R이 1일 때의 Q와 Q'의 값은 고려하지 않는다. 그러나 Q와 Q'값을 S와 R과 피드백 해주는 데에서 차이가 발생한다. NAND 게이트를 사용한 RS 플립플롭의 회로도를 확인하면 S는 Q'와 연산이 되어 Q를 결과로 출력하고, R은 Q와 NOR 연산이 되어 Q'를 결과로 출력한다. 이는 Clock과 NAND 게이트 연산을 하면서 R의 값이 원하고자 하는 값의 보수가 되기 때문이다.

디자인 소스 코드를 짜기 위해 우선 input과 output을 정해야 한다. Clock을 사용한 RS 플립플롭을 디자인해야 하므로 R, S, Clock을 input으로 설정해야 한다. 각각 r, s, 그리고 clockpulse로 정했다. 그리고 Q와 Q'를 출력하는 output으로 각각 q와 nq를 설정했다. RS 플립플롭 내 모든 회로에서 NAND 게이트를 사용해야 하므로 clockpulse를 r, s와 각각 nor 게이트로 연결하여 각각 reset, set wire로 출력되도록 한다. reset과 set은 Clock이 활성화되었을 때 우리가 입력하고자 한 정상적인 R과 S의 값의 보수를 가지므로 앞에서 설명한 것처럼 reset와 set를 각각 q, nq와 NAND 게이트 연산을 하여 다음 상태의 q, nq값을 출력하도록 한다.

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	1	1

Simulation을 통해 RS 플립플롭이 정상적으로 구동되는지 확인하기 위해 R과 S 값을 달리한 입력 경우마다 순서를 부여하여 각 입력들에 따라 출력이 어떻게 되는지 확인했다. 이를 정리한 진리표는 왼쪽 표와 같다.

왼쪽 표의 순서처럼 initial 블록에 50ns가 지날 때마다 그에 맞게 R 또는 S 값을 적절히 설정하여 그에 따른 Q와 Q'값이 어떻게 출력되는지 확인했다. Simulation 결과는 위의 Timing Diagram에

서 보여준다.

NOR 게이트만을 사용할 때와 NAND 게이트만을 사용할 때 출력되는 결과가 다른 부분이 존재하는데, 이는 R과 S가 모두 1로 설정되었을 때이다. RS 플립플롭에서 이 경우의 출력은 don't care로 고려하지 않지만, 실제로 NOR 게이트만을 사용했을 때는 Q와 Q'가 모두 0을, NAND 게이트만을 사용했을 때는 모두 1을 출력한다는 사실을 알 수 있다.

Ⅱ D Flip-Flop

1. D Flip-Flop

1) Verilog 코딩

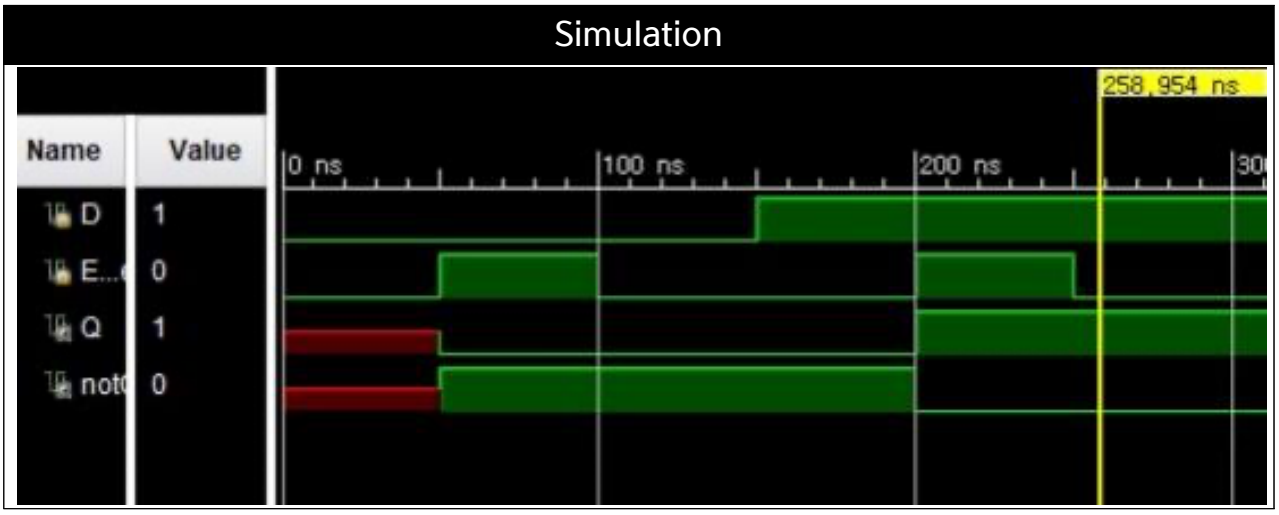
inv.v

```
1  `timescale 1ns / 1ps
2
3  module inv(d,e, q, nq);
4
5  input d, e;
6  output q, nq;
7
8  assign q = ~(((~d) && e)|| nq);
9  assign nq = ~((d && e) || q);
10 endmodule
11
12
```

inv_tb.v

```
1  `timescale 1ns / 1ps
2
3
4  module inv_sim;
5
6  reg D,Enable;
7  wire Q, notQ;
8
9  inv d_inv(
10     .d(D), .e(Enable), .q(Q), .nq(notQ)
11   );
12
13 initial begin
14       D = 1'b0;
15       Enable = 1'b0;
16       #50 Enable = ~Enable;
17       #50 Enable = ~Enable;
18       #50 D = ~D;
19       #50 Enable = ~Enable;
20       #50 Enable = ~Enable;
21 end
22
23 initial begin
24       #800
25       $finish;
26  end
27 endmodule
28
```

3) Simulation 결과



4) 과정

예비 보고서에서 조사한 바와 같이 D 플립플롭에서는 Enable 신호(Clock과 같다고 생각해도 무관함)가 1이 되었을 때 입력한 D의 값이 그대로 Q로 출력되는 것을 알 수 있다. Q'는 항상 출력되는 Q의 보수로 출력된다. 그런데 구현 방법과 회로도 면에서 RS 플립플롭과 유사하다는 것을 알 수 있는데, 이는 D 플립플롭에서 D'를 R로, D를 S로 치환하면 RS 플립플롭과 같아지기 때문이다. 그러나 D 플립플롭에서는 D와 D' 입력이 항상 보수 관계이므로 Enable 신호에 의하지 않는 이상 S와 R이 모두 1일 때처럼 don't care 출력이 나타나는 현상은 발생하지 않는다.

디자인 소스 코드를 짜기 위해 우선 input과 output을 정해야 한다. Enable 신호(Clock)을 사용한 D 플립플롭을 디자인해야 하므로 D와 Enable을 input으로 설정해야 한다. 이를 각각 d 그리고 e로 정했다. 그리고 Q와 Q'를 출력하는 output으로 각각 q와 nq를 설정했다. D를 두 신호로 갈라야 하는데, 한 신호에는 D'가 들어가도록 inverter를 거치도록 한다. e가 1이 되었을 때 d와 d' 둘 중 하나의 신호만을 활성화되도록 해야 하므로 e를 d, d'와 각각 AND 게이트로 연산시킨다. 연산한 결과를 앞에서 설명한 것처럼 각각 q, nq와 NOR 게이트 연산을 하여 다음 상태의 q, nq값을 출력하도록 한다.

Input		Output	
D	E	Q	~Q
0	0	X	X
0	1	0	1
0	0	0	1
1	0	0	1
1	1	1	0
1	0	1	0

Simulation을 통해 D 플립플롭이 정상적으로 구동되는지 확인하기 위해 D와 E 값을 달리한 입력 경우마다 순서를 부여하여 각 입력들에 따라 출력이 어떻게 되는지 확인했다. 이를 정리한 진리표는 왼쪽 표와 같다.

왼쪽 표의 순서처럼 initial 블록에 50ns가 지날 때마다 그에 맞게 D 또는 E 값을 적절히 설정하여 그에 따른 Q

와 Q'값이 어떻게 출력되는지 확인했다. Simulation 결과는 위의 Timing Diagram에서 보여준다.

처음 D와 E 신호가 0으로 입력될 때 RS 플립플롭 때와는 다르게 Q와 ~Q가 모두 don't care로 출력이 되는데, 이는 RS 플립플롭을 Simulation할 때는 처음부터 clock을 1로 설정하고 실험을 했지만, 여기서는 처음부터 enable 신호를 0으로 입력하고 실험했기 때문에 결과 값이 정해지지 않아 출력이 don't care로 나온다.

Ⅲ 논의

1. 결과 검토 및 논의사항

실험 결과에서 예비보고서에서 조사한 내용과 일치한다는 사실을 알 수 있다. 단, 한 가지 의문인 점은 원래 플립플롭은 clock이 1에서 0으로 신호가 떨어지거나 0에서 1로 값이 올라갈 때 delay를 두고 앞의 주기의 입력 값과 현재 저장되어 있는 Q와 Q'의 값을 참고하여 뒤의 주기가 시작할 때 출력 값을 변경시키는 것으로 알고 있었는데, 실제로 실습을 진행했을 때 나타나는 Time Diagram에서는 입력을 주고 나서 바로 출력 값이 변화하는 양상을 보인다. 예비 보고서에서 조사한 내용과 같이 clock 또는 enable 신호의 level에 따라서 출력 값이 변하는 동작을 한 것과 마찬가지이다. 실제로 RS 래치와 D 래치의 timing diagram을 조사했을 때 실험에서 simulation 결과로 나온 것과 일치한다는 사실을 알아냈다. 그래서 플립플롭이 아니라 래치를 구현한 것에 더 가깝지 않은가 하는 생각이 들었다.

RS 플립플롭에서 NOR 게이트와 NAND 게이트를 사용했을 때 R, S 값에 따라서 다음 상태의 Q가 어떻게 변하는지를 실습 시간에 제공한 표에 정리했다.

Present State Q(t)	Q(t+1) Next State			
	INPUT RS			
	00	01	10	11
0	0	1	0	0
1	1	1	0	0

Present State Q(t)	Q(t+1) Next State			
	INPUT RS			
	00	01	10	11
0	0	1	0	1
1	1	1	0	1

2. 추가 이론 조사 및 작성

실험에서는 Truth Table 또는 Characteristic Table을 사용하여 입력 값에 따른 출력을 정리했지만, 순서 논리 회로를 제작할 때는 Excitation Table을 자주 사용한다.

Excitation Table이란 플립플롭에 저장된 Q와 다음 상태의 Q만을 참고하여 플립플롭의 종류에 따라 입력이 어떠한 값으로 들어갔는지를 알아낼 수 있는 표이다. D, RS, T, JK 플립플롭의 Excitation Table을 그리면 아래와 같다.

q	q_{next}	D
0	0	0
0	1	1
1	0	0
1	1	1

q	q_{next}	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

q	q_{next}	T
0	0	0
0	1	1
1	0	1
1	1	0

q	q_{next}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

위의 표에서 X는 0 또는 1 중 어느 것도 들어올 수 있다는 뜻이다. RS 플립플롭 Excitation 표에서 형광색으로 칠해진 부분을 예를 들어, RS 플립플롭에서 현재 저장된 Q의 값이 0일 때 S와 R이 모두 0이면 다음 상태의 Q는 0이고, S가 0이고 R이 1이어도 다음 상태의 Q는 0이 된다. 따라서 S와 R이 각각 0과 0 또는 0과 1일 때 현재 상태의 Q가 0이면 다음 상태의 Q가 0이라는 것이다. JK 플립플롭에서는 S와 R이 모두 1일 때 Q가 다음 상태에서는 보수로 출력되므로 모든 Q와 다음 Q 상태에 관해 don't care가 J와 K에 한 번씩 생긴다. 다음 12주차 실습에서 조사하는 counter와 같은 순차 회로를 제작할 때 excitation table이 굉장히 유용하다.