11주차 결과보고서

전공: 컴퓨터공학 학년: 2학년 학번: 20191629 이름: 이주헌

1. RS Latch의 결과 및 simulation 과정에 대해서 설명하시오.

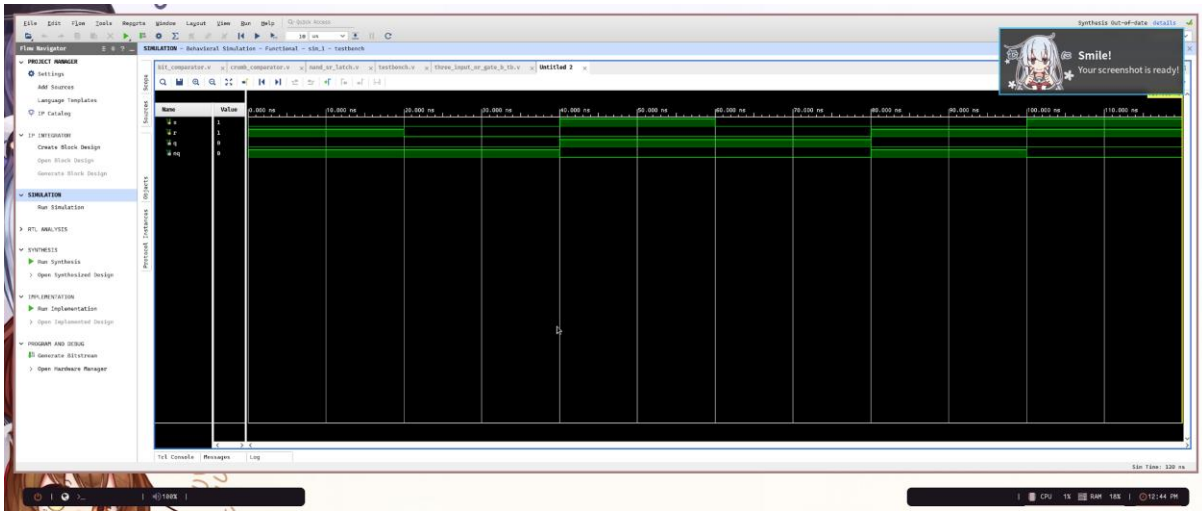
RS latch는 래치라는 이름에 맞게 클럭 입력을 받지 않고, 이전 상태를 계속 유지하는 회로이다. 즉, 다음과 같은 진리표를 따르는 회로를 만들어야 한다.

set	reset	q	not q
0	0	q	not q
0	1	0	1
1	0	1	0
1	1	metastable	metastable

이 회로는 set 핀과 reset 핀을 입력으로 받고, 각 출력이 입력에 다시 들어가서 영향을 주는 피드백 루프로 구현이 되어 있다. 이 회로는 NOR 또는 NAND 게이트를 사용해서 구현할 수 있는데, 두 가지 종류의 래치 모두를 구현하였다. 우선 NOR 게이트를 사용하는 래치이다.

```
`timescale 1ns / 1ps

module nor_sr_latch(s, r, q, nq);
    input s, r;
    output q, nq;

    nor(q, r, nq);
    nor(nq, s, q);
endmodule
```

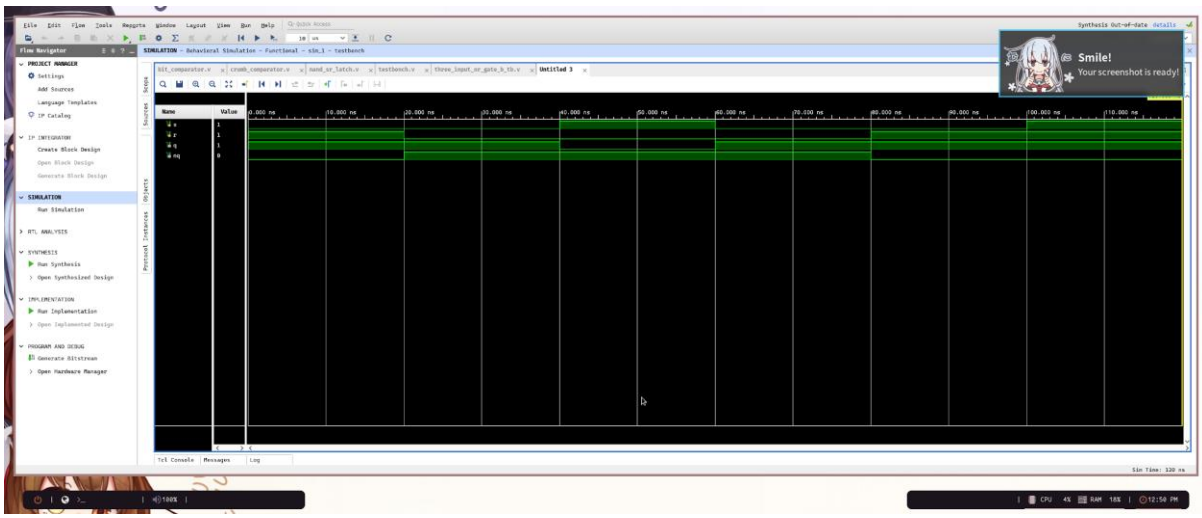
여기서 q는 출력, 그리고 nq는 출력을 반전한 결과이다. 위 코드를 시뮬레이션하면 다음과 같은 결과를 얻을 수 있다.



기본적으로 NAND와 NOR 모두 universal logic gate이므로 모든 회로를 NAND 또는 NOR 게이트만을 사용해서 구성할 수 있다는 사실이 알려져 있다. 즉, NOR 게이트를 사용하는 위 회로를 NAND 게이트를 사용해서도 구성할 수 있다.

```
`timescale 1ns / 1ps

module nand_sr_latch(s, r, q, nq);
    input s, r;
    output q, nq;

    nand(q, s, nq);
    nand(nq, r, q);
endmodule
```

위 코드를 시뮬레이션하면 다음과 같은 결과를 얻을 수 있다.



여기서 s와 r 입력이 모두 1인 경우 값이 다르게 나오는 사실을 확인할 수 있는데, 이는 해당 입력이 metastable하기 때문으로, RS 래치의 특성상 두 입력이 모두 1인 경우 신뢰할 수 없는 출력이 도출되게 된다는 특징이 있다. RS 래치는 이후 실험에서 사용할 다른 모든 플립플롭 또는 래치의 기본형이 되고, NAND 게이트를 사용한 구현체와 NOR 게이트를 사용한 구현체의 동작에 차이가 없으므로 이후 실험에서는 NOR만을 사용해서 구현하는 것을 원칙으로 한다.

2. RS flip-flop의 결과 및 simulation 과정에 대해서 설명하시오.

RS 플립플롭은 RS 래치에 클럭 입력을 추가하여 클럭에 맞춰 출력을 바꿀 수 있도록 하는 전기 회로를 말한다. 즉, 클럭 신호가 0일 때에 바뀐 입력은 다음 클럭 입력이 1이 될 때까지 적용되지 않게 된다. 따라서, 이전 실험에서 작성했던 RS 래치의 기본형을 모듈으로 사용하면 쉽게 클럭 입력을 적용한 RS 플립플롭을 만들 수 있다.

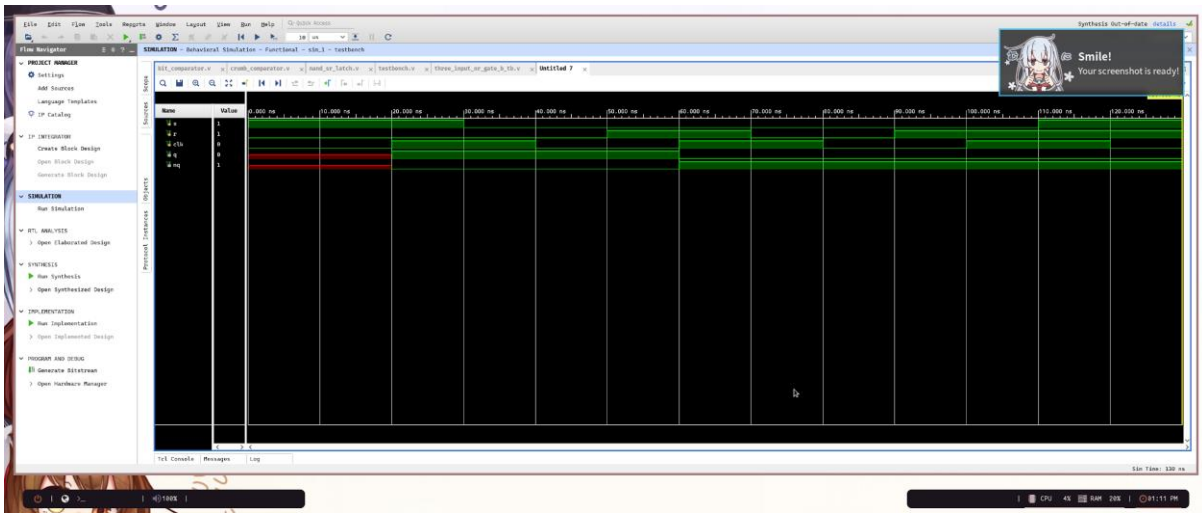
```
`timescale 1ns / 1ps

module sr_flip_flop(s, r, clk, q, nq);
    input s, r, clk;
    output q, nq;
    reg s_clk, r_clk;

    always @(posedge clk) begin
        s_clk <= s;
        r_clk <= r;
    end

    nor_sr_latch device(.s(s_clk), .r(r_clk), .q(q), .nq(nq));
endmodule
```

위 모듈에 특정 순서로 입력을 집어넣으면 다음과 같은 형태의 표를 얻을 수 있다. 이는 플립플롭 및 래치의 정의에 따라 작성한 것이다.

순서	set	reset	q	not q
1	1	0	1	0
2	0	0	1	0
3	0	1	0	1

4	0	0	0	1
5	0	1	0	1
6	1	1	metastable	metastable

여기서 얻은 순서표를 실제로 시뮬레이션으로 확인해보면 다음과 같은 그래프를 얻을 수 있다. 시뮬레이션을 돌릴 때, 일부러 클럭 입력이 바뀔 때와 입력 신호가 바뀔 때 사이에 10ns 정도의 지연 시간을 집어넣어 플립플롭의 특성도 함께 관찰할 수 있도록 하였다.



여기서 알 수 있듯이, RS 플립플롭은 래치와 다르게 클럭 입력의 상승 에지에서면 값을 업데이트하지만, RS 래치의 문제점인 metastable 상태를 보정하는 논리가 하나도 존재하지 않기 때문에 RS 래치의 문제점을 함께 가지고 있다는 특징이 있다. 즉, s 입력과 r 입력이 모두 1인 경우에 대한 q 및 not q 출력은 정의되어 있지 않다.

3. D latch의 결과 및 simulation 과정에 대해서 설명하시오.

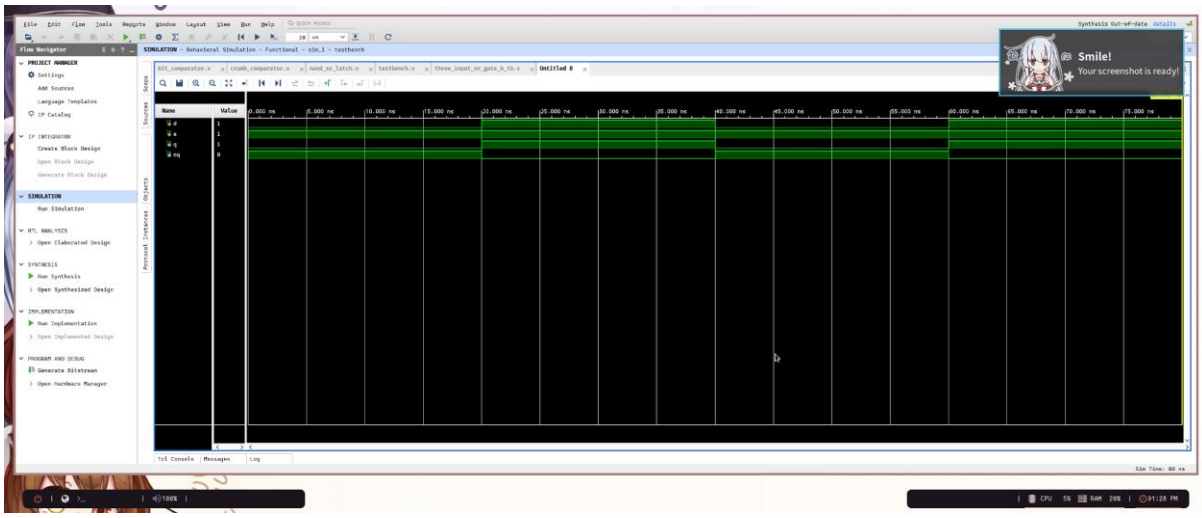
D 래치는 RS 래치의 입력을 하나로 묶은 형태로서, RS 래치의 set 핀은 D 입력에 바로 연결되고 reset 핀은 D 입력을 반전한 결과에 연결된다. 따라서, 이론적으로는 set 핀과 reset 핀이 동시에 1이 되는 경우는 존재할 수 없기 때문에 RS 래치의 metastable 오류를 원천적으로 차단할 수 있는 래치이다. 이 역시 위에서 구현한 NOR RS 래치 모듈을 사용하여 쉽게 구현할 수 있다.

D	Enable	Set	Reset	Q	not Q
0	0	0	0	Q	not Q

1	0	0	0	Q	not Q
0	1	0	1	0	1
1	1	1	0	1	0

```
`timescale 1ns / 1ps

module d_latch(d, e, q, nq);
    input d, e;
    output q, nq;
    wire s, r;

    assign s = d & e;
    assign r = ~d & e;

    nor_sr_latch device(.s(s), .r(r), .q(q), .nq(nq));
endmodule
```

여기서 d가 래치에 주어지는 입력, 그리고 e는 칩 활성화 핀이다. D 래치의 동작 원리로부터, 입력이 1이면 출력이 1, 입력이 0이면 출력도 0이 되므로 주어진 입력을 릴레이하는 논리회로라고 생각할 수 있다. 따라서, 잇달아서 0, 1, 0, 1을 입력한다면 정출력은 0, 1, 0, 1, 역출력은 1, 0, 1, 0이 될 것이다. 이는 시뮬레이션을 통해 검증할 수 있다.



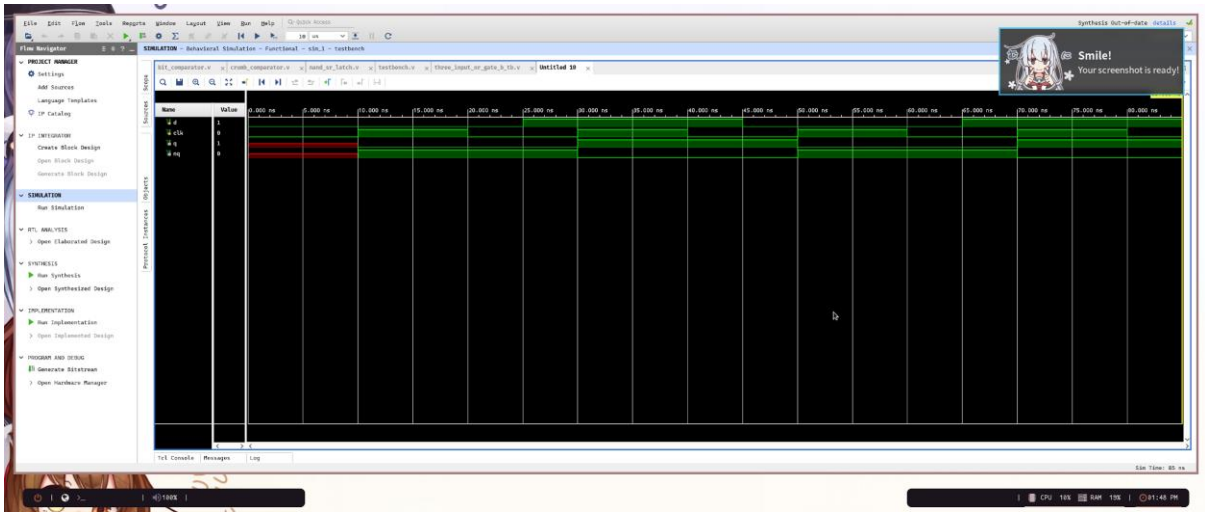
이 시뮬레이션에서는 E(칩 활성화) 핀이 바뀌는 입력을 무시하는 역할을 한다는 사실이 명백하기 때문에 편의상 E 핀은 항상 1으로 두고 시뮬레이션하였다. 그래프를 잘 확인해보면, 위에서 세운 가설과 완전히 같은 결과가 나온다는 것을 알 수 있다.

4. D flip-flop의 결과 및 simulation 과정에 대해서 설명하시오.

RS 래치와 RS 플립플롭 사이의 관계와 마찬가지로, D 래치에 클럭 입력이 추가된 모양을 하고 있다. 따라서, D 플립플롭은 D 입력이 바뀌더라도 클럭 입력이 들어오기 전까지 입력을 처리하지 않는 논리 회로이다. 이 회로는 위에서 구현한 D 래치를 모듈으로 이용하여 다음과 같이 구현할 수 있다.

```
`timescale 1ns / 1ps

module d_flip_flop(d, clk, q, nq);
    input d, clk;
    output q, nq;
    reg d_clk;

    always @(posedge clk) begin
        d_clk <= d;
    end

    d_latch device(.d(d_clk), .e(1), .q(q), .nq(nq));
endmodule
```

D 플립플롭은 기본적으로 D 래치에 클럭 입력을 추가하여 만드는 것이기 때문에 입력 값을 정출력으로 릴레이한다는 점은 래치와 동일하다. 그래서 입력이 처리되는 시간에 대해 조금 더 알아보기 위하여 일부러 클럭 신호와 입력이 바뀌는 시간에 차이를 주어 시뮬레이션하였다.



시뮬레이션 결과에서 알 수 있듯이, D 플립플롭은 일반적인 D 래치와 다르게 클럭 신호의 상승 에지에서만 결과값을 업데이트한다는 것을

5. JK flip-flop의 결과 및 simulation 과정에 대해서 설명하시오.

JK 플립플롭은 RS 플립플롭의 고질병인 metastable 문제를 해결할 수 있는 플립플롭 회로이다. RS 플립플롭의 입력값에 하나의 피드백 루프를 추가하여 J, K 입력이 모두 1인 경우에도 안정적으로 결정론적인 출력을 생성할 수 있도록 개량한 것이라고 볼 수 있는데, 다음과 같은 진리표를 따른다.

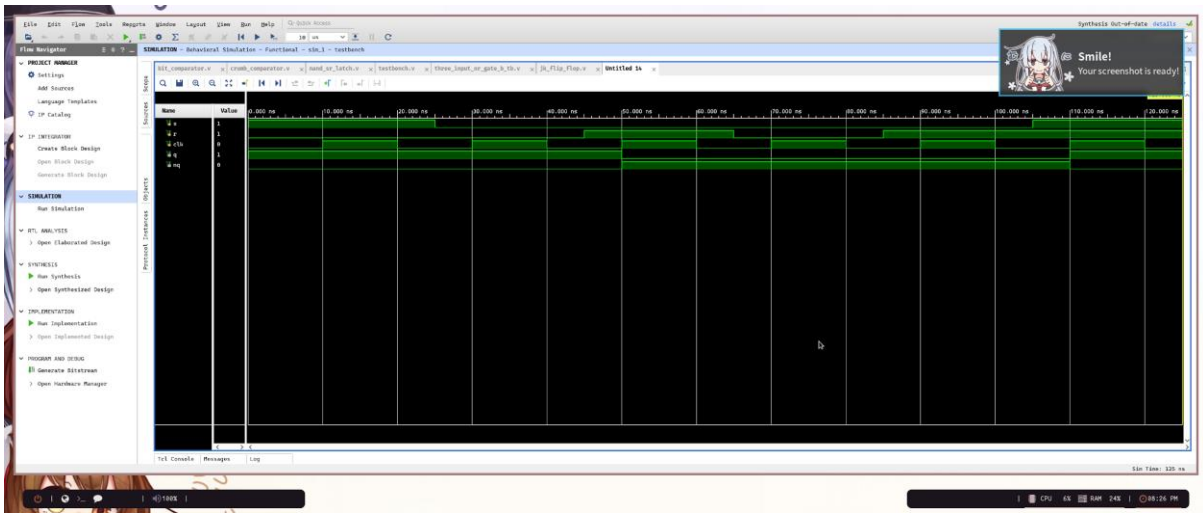
J	K	q	not q
0	0	q	not q
0	1	0	1
1	0	1	0
1	1	not q	q

위 진리표에서 알 수 있듯이, J와 K 입력이 모두 1이면 이전 상태를 반전시킨 결과를 출력한다. 이 회로는 RS 플립플롭의 S 핀에 J와 ~Q 핀을 AND 게이트에 통과시킨 결과를 할당하고, R 핀에 K와 Q 핀을 AND 게이트에 통과시킨 결과를 할당하는 것으로 구현할 수 있다. 이를 Verilog로 구현하면 다음과 같다.

```
`timescale 1ns / 1ps

module jk_flip_flop(j, k, clk, q, nq);
    input j, k, clk;
    output q, nq;
    reg s, r;

    always @(posedge clk) begin
        s <= j & nq;
        r <= k & q;
    end

    nor_sr_latch device(.s(s), .r(r), .q(q), .nq(nq));
endmodule
```

RS 플립플롭에서 사용했던 테스트를 똑같이 적용시켜보면 다음과 같은 결과가 나와야 한다.

순서	J	K	q	not q
1	1	0	1	0
2	0	0	1	0
3	0	1	0	1
4	0	0	0	1
5	0	1	0	1
6	1	1	1	0



시뮬레이션 결과를 위 순서표와 비교해 보았을 때, JK 플립플롭의 정의에 따라 잘 작동한다는 것을 알 수 있다.

6. 결과 검토 및 논의 사항

먼저 첫 번째 실험에서는 다른 회로를 구현할 때 기본이 되는 RS 래치를 구현하였다. SR 래치의 시뮬레이션에서는 여러 번 시뮬레이션을 돌려도 S와 R이 모두 1인 경우를 포함해서 같은 결과가 도출되는 것을 볼 수 있었다. 이는 시뮬레이션이 현실의 전자 부품을 완전히 흉내낼 수 없기 때문으로, 현실의 전자 부품에서 발생하는 지연 시간 등이 구현되어 있지 않기 때문이라고 볼 수 있다. 이 회로는 특히 NAND와 NOR 게이트 모두를 사용하여 구현하였는데, 두 회로 모두 기능상 차이가 없다는 것을 다시 확인하였다. 또, RS 래치를 사용하여 클럭 신호의 상승 에지에 값을 업데이트하는 RS 플립플롭도 구현하였다.

다음으로 D 래치와 D 플립플롭을 구현하였다. 기본적으로 D 래치는 RS 래치에서 S 핀과 R 핀을 D라는 하나의 입력 신호로 조절하는 회로이기 때문에, 식을 처음부터 다시 만들기보다는 이미 만들어 두었던 NOR 기반 RS 래치 모듈을 재사용하는 방식으로 구현하였다. D 플립플롭 역시 클럭 신호의 상승에지에 값을 업데이트하는 방식을 구현했는데, 미리 만들어 둔 D 래치를 사용해서 구현하였다.

마지막으로 JK 플립플롭을 구현하였다. JK 플립플롭은 RS 플립플롭의 metastable 상태 문제를 해결하기 위해 고안된 플립플롭 디자인으로, J와 K 핀이 모두 1이라면 이전 출력을 반전하여 출력하는 성질을 가지고 있다. 클럭 신호와 두 개의 피드백 루프를 가지고 있는 JK 플립플롭의 특성상 이 플립플롭은 래치로 구현할 수 없다. 하지만 이 회로 역시 내부적으로는 RS 래치와 같은 구조를 사용하기 때문에, JK 플립플롭도 마찬가지로 NOR 기반 RS 래치 모듈에 입력되는 값을 조절하는 방식으로 구현하였다.

7. 추가 이론 조사 및 작성

T 플립플롭은 toggle 플립플롭이라는 이름에 걸맞게, 입력이 들어올 때마다 출력값을 반전하고, 입력이 없어져도 현재 상태를 유지하는 플립플롭 회로이다. 즉, 다음과 같은 진리표를 만족해야 한다.

T	q	not q
0	q	not q
1	not q	q

이 진리표를 잘 비교해보면, JK 플립플롭에서 2번, 3번 행을 없앤 것과 출력 값이 같다는 사실을 알 수 있다.

J & K	q	not q
0 & 0	q	not q
0 & 1	0	1

1 & 0	1	0
1 & 1	not q	q

따라서, 이미 JK 플립플롭이 구현되어 있다면 이 플립플롭을 재사용하여 아주 간단하게 T 플립플롭을 구성할 수 있다. T 플립플롭의 Verilog 코드는 다음과 같다.

```
`timescale 1ns / 1ps

module t_flip_flop(t, clk, q, nq);
    input t, clk;
    output q, nq;

    jk_flip_flop device(.j(t), .k(t), .clk(clk), .q(q), .nq(nq));
endmodule
```