LDLAB 6

수리과학부 2019-16022 박채연

0. Discussion

(1) Concepts that I learned during the lab session

간단한 Sequential Logic Circuit인 R-S latch, gated R-S latch, simple oscillator, R-S flipflop을 구현해보면서 해당 회로들의 동작 원리를 이해할 수 있었다. Sequential Logic Circuit이란 출력이 입력뿐만 아니라 이전 상태에 의해 결정되는 회로이다. 따라서 입력값이 같더라도 이전 상태가 다르다면 출력결과가 다르다는 특징이 있다.

본 랩 실습 이전에는 R-S latch와 R-S flipflop의 개념이 헷갈렸는데, 직접 구현해보면서 두 회로의 차이를 깊이 이해할 수 있었다. 간단하게 설명하자면, R-S latch에 clock이 추가된 것이 R-S flipflop이다. gate delay를 고려하지 않았을 때, R-S latch의 경우 R 또는 S 값이 바뀌면 바로 Q와 Q'의 값에 영향을 미치지만 positive edge triggered R-S flipflop의 경우 clock이 0에서 1로 바뀌는 순간의 (R, S) 값에 따라 Q와 Q'의 값이 결정되고, negative edge triggered R-S flipflop의 경우 clock이 1에서 0로 바뀌는 순간의 (R, S) 값에 따라 Q와 Q'의 값이 결정된다는 차이가 있다는 것을 배웠다.

이론 실습시간에 교수님께서 다양한 flipflop에 대해 설명해주셨지만 그 종류가 많아 기억하기가 어려웠다. 그러나 Homework part를 수행하면서 R-S flipflop, D flipflop, JK flipflop, T flipflop의 동작 방식을 다시 한 번 정리해볼 수 있었다. 간단히 정리하자면 R-S flipflop은 (R, S) = (0, 0)이면 Q 값이 유지되고, (R, S) = (1, 0)이면 Q 값이 0으로 설정된다. 또한, (R, S) = (0, 1)이면 Q 값이 1로 설정되며 (R, S) = (1, 1)일 때 Q 값이 unstable하다. JK flipflop은 (R, S) = (1, 1)일 때 Q 값이 unstable 해지는 문제점을 보완하는 flipflop으로 (J, K) = (1, 1)이면 Q 값이 반전된다는 특징이 있다. 그 외의 경우에는 R-S flipflop과 동일한 방식으로 동작한다. D flipflop은 입력값인 D와 출력값인 Q가 같은 flipflop이다. 마지막으로 T flipflop은 T = 0일 때 Q 값이 유지되지만 T = 1일 때 Q값이 toggle 된다는 특징이 있다. 또한, 4가지 flipflop 모두 신호를 받을 때만 동작하고 신호를 받지 않는다면 계속 상태가 유지되는 Sequential Logic Circuit이다.

(2) Any errors that I made

a gated R-S latch를 구현할 때 always 문을 통해 E, R, S 중 하나의 값이 바뀔 때마다 R AND E와 S AND E를 다시 계산하고 이를 이미 구현한 R-S latch의 input으로 전달함으로써 a gated R-S latch를 구현하고자 하였다. a gated R-S latch를 구현하고 나서 test code를 작성하여 시뮬레이션 결과를 확인해보았는데, E = 1, R = 1, S = 0에서 E = 1, R = 0, S = 1로 바뀔 때 Q 값이 0에서 1로 바뀌어야 함에도 불구하고 계속 0으로 값이 유지되는 등의 문제가 발견되었다. 논리상으로 해당 구현이 올바르다고 생각해 시뮬레이션 결과가 잘못된 이유를 찾아내기 어려웠다. 코드를 천천히 다시 살펴보는 과정에서 오타로 인해 이미 구현한 R-S latch의 input으로 R AND E 와 S AND E가

전달되는 것이 아니라 R AND E만 두 번 전달되는 것을 발견하였다. (레지스터 r 에 R & E, 레지스터 s에 S & E를 저장하고 이를 이미 구현한 R-S latch에 input으로 전달하고자 하였으나 오타로 인해 레지스터 s에도 R & E가 저장되었다.)

R-S flipflop을 구현할 때 앞에서 구현한 모듈을 이용하여 구현하라는 지시사항을 보지 못하고 Structural Description으로 회로도에 따라 구현하는 실수를 저질렀다. 또한, Structural Description 방법으로 구현한 모듈은 (R, S) = (1, 1)일 때 Q 값과 Q' 값이 모두 1로 나타났는데, 이전에 구현했던 R-S latch는 (R, S) = (1, 1)일 때 Q 값과 Q' 값을 모두 임의로 0으로 지정했었기 때문에 값이 일관적이지 않다는 문제점도 있었다.

(3) How to correct my errors

구현한 a gated R-S latch 코드를 다시 살펴보는 과정에서 오타로 인해 R-S latch의 input으로 R AND E와 S AND E가 각각 전달되는 것이 아니라 R AND E 만 두 번 전달되는 것을 확인하였다. 레지스터 s 의 값을 'R & E'가 아니라 'S & E'로 수정해주었고 다시 시뮬레이션을 수행해보니 정상적인 결과가 출력되는 것을 확인하였다.

R-S flipflop을 Structural Description 방법으로 구현한 코드를 모두 주석처리하고 Behavioral Description 방법으로 구현한 R-S latch를 이용하여 Behavioral Description 방법으로 구현하였다. always @(posedge Clk) 문법을 사용하여 Clock이 0에서 1로 바뀔 때마다 always 문 내부 코드가 실행되도록 구현하였고, 이로써 (R, S) = (1, 1)일 때 앞에서 구현한 R-S latch와 값이 일관적이지 않다는 문제 역시 해결할 수 있었다.

1. Implement the following in Verilog and simulate the behavior.

(1) a R-S latch

R-S latch의 동작을 표로 나타내면 아래와 같다. 즉, S = R = 0 이면 이전 값이 유지되어야 하고, S = 0, R = 1이면 Q 값이 0으로 설정되어야 한다. 또한, S = 1, R = 0이면 Q 값이 1로 설정되어야 하며 S = R = 1이면 Q 값이 불안정해진다.

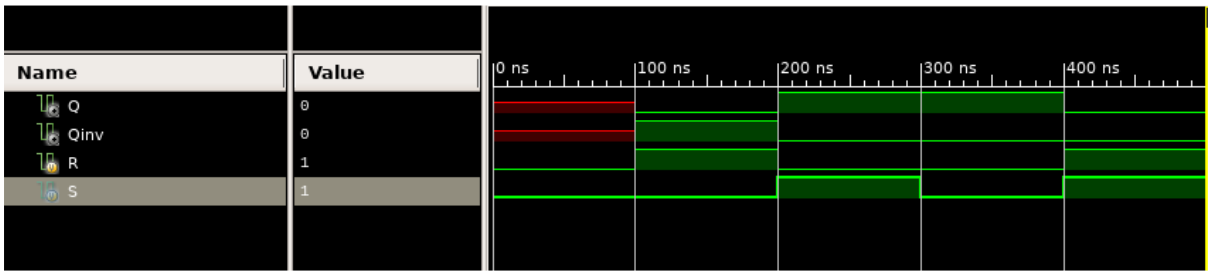
S	R	Q
0	0	Hold
0	1	0
1	0	1
1	1	Unstable

이를 고려하여 R-S latch를 Behavioral Description 방법으로 'RS_latch.v' 파일에 구현하였다. 이를 살펴보면 아래와 같다.

```
module RS_latch(
    input R,
    input S,
    output Q,
    output Qinv
    );

        reg q, qinv;

        assign Q = q;
        assign Qinv = qinv;

        always @(R or S)
                begin
                        if (R==1&&S==0)
                                begin
                                        q = 0;
                                        qinv = 1;
                                end
                        if (R==0&&S==1)
                                begin
                                        q=1;
                                        qinv =0;
                                end
                        if (R==1&&S==1)
                                begin
                                        q=0;
                                        qinv =0;
                                end
                end
endmodule
```

always문 내에서 if문을 사용하여 R과 S값이 바뀔때마다 해당 값들을 확인하고 각 경우에 맞는 Q 값과 Q′ 값을 지정해주었다. 이때, R = S = 1는 not allowed 조건인데, LAB06 ppt 6페이지의 시뮬레이션 결과와 일치하도록 (R, S) = (1, 1)일 때 임의로 Q와 Q′ 값을 모두 0으로 지정하였다.

R-S latch 시뮬레이션 결과는 아래와 같다. 앞에서 제시한 표와 마찬가지로 S = 0, R = 0일 때 이전 값이 hold되고, S = 0, R = 1일 때 Q = 0이 되며, S =1, R = 0일 때 Q가 1이 되는 것을 확인 가능하다.



(2) a gated R-S latch using (1)

a gated R-S latch의 동작을 표로 나타내면 아래와 같다. 즉, E(enable) 값이 0이면 상태가 계속 유지되고, 1이면 (1)에서의 R-S latch와 동일한 방식으로 동작한다.

E	S	R	Q
0	-	-	Hold
1	0	0	Hold

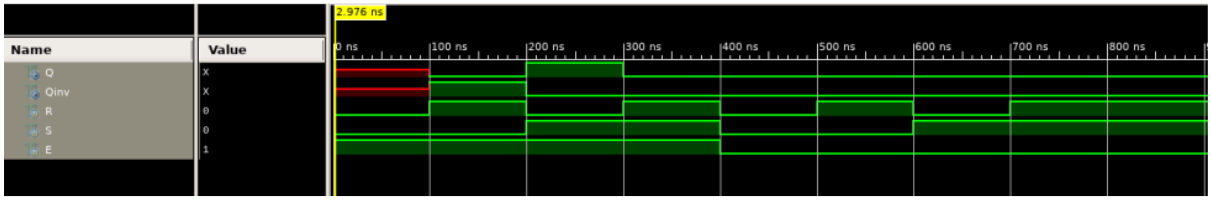
1	0	1	0
1	1	0	1
1	1	1	Unstable

이를 고려하여 a gated R-S latch를 (1)의 R-S latch를 이용하여 Behavioral Description 방법으로 'gated_RS_latch.v' 파일에 구현하였다. 이를 살펴보면 아래와 같다.

```
module gated_RS_latch(
    input R,
    input S,
    input E,
    output Q,
    output Qinv
    );

        reg r, s;

        always @(E or R or S)
            begin
                r = R & E;
                s = S & E;
            end
        RS_latch T1(.R(r), .S(s), .Q(Q), .Qinv(Qinv));

endmodule
```

always 문을 통해 E 혹은 R 혹은 S 값이 바뀔 때마다 R AND E 와 S AND E를 계산하고 이를 (1)에서 구현한 RS_latch 모듈에 input으로 전달하였다.

a gated R-S latch의 시뮬레이션 결과는 아래와 같다. 앞에서 제시한 표와 마찬가지로 E = 0 이면 이전 값이 hold되고, E = 1이면 (1) 에서와 동일한 방식으로 동작하는 것을 확인가능하다.



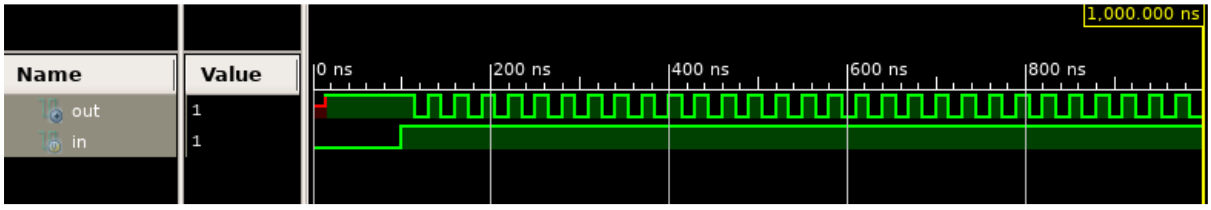
(3) a Simple Oscillator(the clock period is free)

. 구현하고자 하는 Simple Oscillator의 회로는 아래와 같다. Simple Oscillator는 Enable이 0일 때는 출력 상태가 유지되지만 1일 때는 oscillate한다

위의 회로 그림을 고려하여 'Oscillator.v' 파일에 Structural Description 방법으로 Simple Oscillator를 구현하였다.

```
`timescale 1ns / 1ps

module Oscillator(
    input Enable,
    output Feedback
    );

            wire w;

            and T1(w, Enable, Feedback);
            not #5 T2(Feedback, w);


endmodule
```

a simple oscillator 의 시뮬레이션 결과는 아래와 같다. Enable이 0일 때 출력결과가 1로 유지되지만 Enable이 1일 때는 출력결과가 oscillate하는 것을 확인할 수 있다. 한편, 아래의 시뮬레이션 결과를 살펴보면 초기 출력결과가 undefined임을 알 수 있는데, 이는 not gate의 gate delay를 5ns로 지정했기 때문이다.



(4) a R-S Flip-Flop(Edge-triggered) using above modules

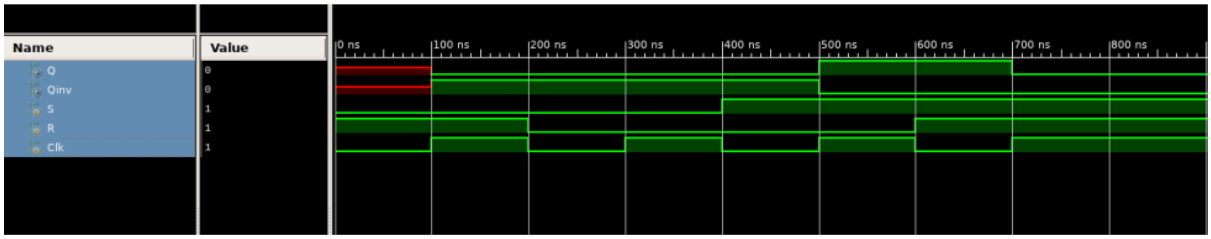
positive edge triggered R-S flipflop은 Clock이 0에서 1로 바뀔 때의 (R, S) 값에 따라 Q 값이 결정되고, negative edge triggered R-S flipflop은 Clock이 1에서 0으로 바뀔 때의 (R, S) 값에 따라 Q 값이 결정된다. (R, S) 값에 따른 Q 값은 R-S latch와 동일하다. 그러나 gate delay를 고려하지 않을 때 R-S latch는 (R, S) 값이 바뀌면 바로 Q 값에 영향을 끼치지만, edge triggered R-S flipflop은 Clock 신호가 0에서 1로 바뀔 때만(혹은 1에서 0으로 바뀔 때만) Q 값에 영향을 끼친다는 차이점이 있다.

이를 고려하여 positive edge triggerd R-S flipflop을 (1)에서 구현한 R-S latch를 이용하여 Behavioral Description 방법으로 'RS_flipflop.v' 파일에 구현하였다. 이를 살펴보면 아래와 같다.

```
module RS_flipflop(
    input S,
    input R,
    input Clk,
    output Q,
    output Qinv
    );

            //wire nand1, nand2;
            //nand T1(nand1, Clk, S);
            //nand T2(nand2, Clk, R);
            //nand T3(Q, nand1, Qinv);
            //nand T4(Qinv, nand2, Q);

            reg r, s;

            always @(posedge Clk)
                    begin
                            r = R;
                            s = S;
                    end
            RS_latch T1(.R(r), .S(s), .Q(Q), .Qinv(Qinv));
endmodule
```

always 문을 통해 Clock 신호가 0에서 1로 바뀌는 순간의 R값과 S 값을 각각 레지스터 r과 s에 저장하고 이를 (1)에서 구현한 R-S latch 모듈의 input으로 전달하였다. 만약 negative edge triggered R-S flipflop을 구현하고 싶다면 always @()에서 괄호 안을 'negedge Clk'로 수정해주면 된다.

positive edge triggered R-S flipflop의 시뮬레이션 결과는 아래와 같다. Clock이 0에서 1로 바뀔 때 앞에서 구현한 R-S latch와 동일하게 동작하고, 그 외의 경우에는 상태가 유지되는 것을 확인 가능하다.



2. Explain the different Flipflops(SR, D, JK, etc.) and how they differ from each other.

Flipflop이란 대표적인 Sequential Logic Circuit으로, 1비트의 정보를 기억할 수 있는 기억 소자이다. 전원이 공급되면, 신호(=Clock)를 받을 때까지 현재의 상태를 유지한다는 특징이 있다. Clock이 0에서 1로 바뀔 때 입력값에 따라 상태가 바뀌는 flipflop을 positive edge triggered flipflop이라고 하고, Clock이 1에서 0으로 바뀔 때 입력값에 따라 상태가 바뀌는 flipflop을 negative edge triggered flipflop이라고 한다.

Flipflop에는 여러가지 종류가 있는데, 그 중에서도 R-S flipflop, JK flipflop, D flipflop, T flipflop이 대표적이다. 각각의 동작방식을 positive edge triggered flipflop을 기준으로 정리하면 다음과 같다.

1) R-S flipflop

Clock이 0에서 1로 바뀔 때 R과 S값에 따른 출력값 Q를 아래와 같이 표로 나타낼 수 있다. 이 때, (R, S) = (1, 1)이면 Q 값이 unstable하기 때문에 이는 허용되지 않은 값이다.

R	S	Q
0	0	Hold
1	0	0
0	1	1
1	1	Unstable

2) JK flipflop

R-S flipflop은 Clock이 0에서 1로 바뀔 때 (R, S) = (1, 1)이면 Q 값이 unstable한데 이러한 단점을 보완한 flipflop이다. Clock이 0에서 1로 바뀔 때 (J, K) = (1, 1)이면 Q 값이 반전된다. Clock이 0에서 1로 바뀔 때 J와 K값에 따른 출력값 Q를 아래와 같이 표로 나타낼 수 있다.

J	K	Q
0	0	Hold
1	0	0
0	1	1
1	1	Toggle

3) D flipflop

D flipflop은 Clock이 0에서 1로 바뀔 때 Q 값이 입력값 D와 같아지는 flipflop이다. 즉, 입력값 D를 Clock Pulse의 시간 간격만큼 지연시켜 출력한다는 특징이 있다. Clock이 0에서 1로 바뀔 때 D 값에 따른 출력값 Q를 아래와 같이 표로 나타낼 수 있다.

D	Q
0	0
1	1

4) T flipflop

T flipflop은 JK flipflop의 특수한 형태로 J와 K를 하나로 묶어 T로 표현한다. 즉, T = 0일 때의 T flipflop의 동작은 (J, K) = (0, 0)일 때의 JK flipflop의 동작과 일치하고, T = 1일 때의 T flipflop의 동작은 (J, K) = (1, 1)일 때의 JK flipflop의 동작과 일치한다. 따라서 T flipflop은 Clock이 0에서 1로 바뀔 때 T = 0이면 Q 값이 유지되고, T = 1이면 Q 값이 반전된다. Clock이 0에서 1로 바뀔 때 T 값에 따른 출력값 Q를 아래와 같이 표로 나타낼 수 있다.

T	Q
0	Hold
1	Toggle