1. 실습 목표

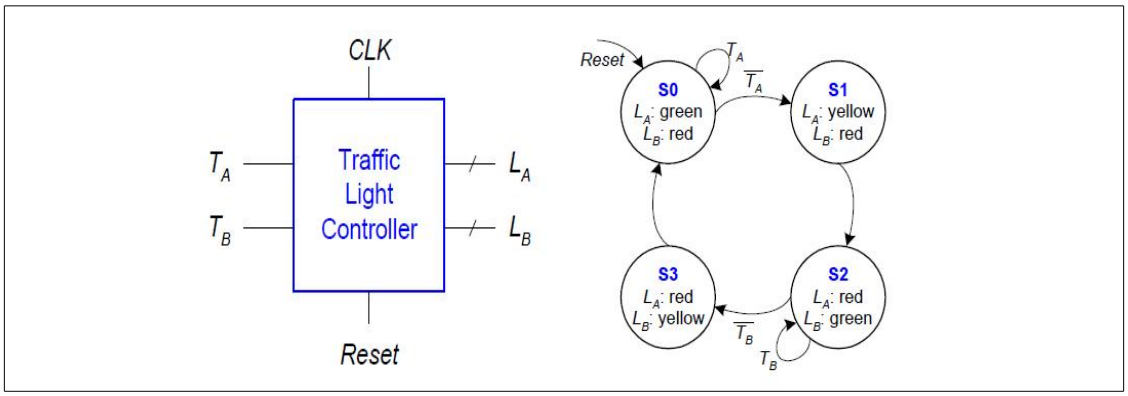
- 사람이 있으면 켜지고 없으면 꺼지는 사거리 신호등을 만든다.

2. Specification

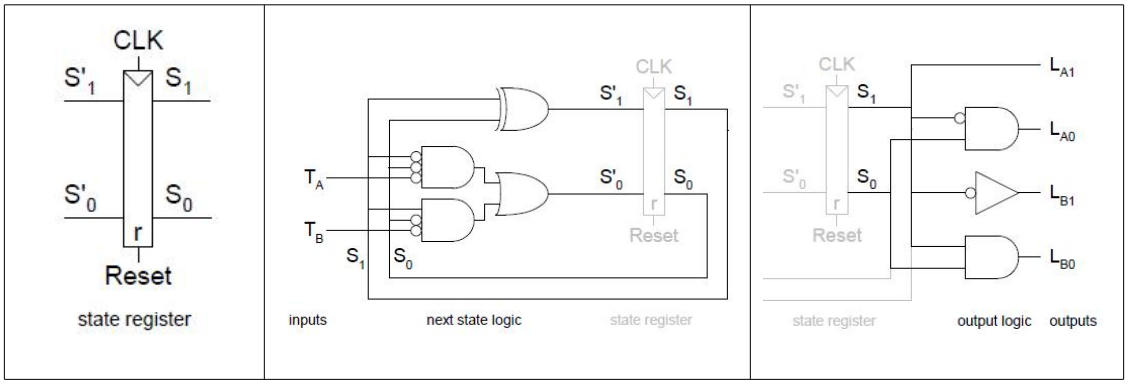
1) I/O port

- Input port : 4개 입력 (T_A, T_B, clock, reset)
- Output port : 2개 출력 (L_A, L_B)

2) 예상 RTL 블록도



3) 예상 내부 회로도



4) 예상 진리표

Current State	Inputs		Next State
S	T_A	T_B	S'
S0	0	X	S1
S0	1	X	S0
S1	X	X	S2
S2	X	0	S3
S2	X	1	S2
S3	X	X	S0

Current State		Inputs		Next State	
S_1	S_0	T_A	T_B	S'_1	S'_0
0	0	0	X	0	1
0	0	1	X	0	0
0	1	X	X	1	0
1	0	X	0	1	1
1	0	X	1	1	0
1	1	X	X	0	0

State	Encoding
S0	00
S1	01
S2	10
S3	11

Current State		Inputs		Next State		D FF input	
S_1	S_0	T_A	T_B	S'_1	S'_0	D_A	D_B
0	0	0	X	0	1	0	1
0	0	1	X	0	0	0	0
0	1	X	X	1	0	1	0
1	0	X	0	1	1	1	1
1	0	X	1	1	0	1	0
1	1	X	X	0	0	0	0

$S'_1 = S_1 \oplus S_0$

$S'_0 = \overline{S_1}\overline{S_0}T_A + S_1\overline{S_0}\overline{T_B}$

$D_A = \Sigma(1,2)$

$D_B = 000X+11X0$

Current State		Outputs			
S_1	S_0	L_{A1}	L_{A0}	L_{B1}	L_{B0}
0	0	0	0	1	0
0	1	0	1	1	0
1	0	1	0	0	0
1	1	1	0	0	1

$L_{A1} = S_1$

$L_{A0} = \overline{S_1}S_0$

$L_{B1} = \overline{S_1}$

$L_{B0} = S_1S_0$

Output	Encoding
green	00
yellow	01
red	10

3. 실험내용

```
module traffic_light(rst, clk, T, S, next_S, La, Lb);
input wire [1:0] T;              // 입력T 선언 (T[0]는 T_A, T[1]는 T_B로 지정)
input wire rst, clk;             // reset, clock 선언
output wire [1:0] La, Lb;        // 출력 L_A,L_B 선언
output wire [3:0] S, next_S;
 // 이전상태와 다음상태 선언
 // S=0001일 때 상태는 S_0, next_S=0001일 때 S_0'
 // S=0010일 때 상태는 S_1, next_S=0010일 때 S_1'
 // S=0100일 때 상태는 S_2, next_S=0100일 때 S_2'
 // S=1000일 때 상태는 S_3, next_S=1000일 때 S_3'

reg [1:0] la, lb;
reg [3:0] next_s, s;
 // 조건문안에 임시로 저장해줄 함수 선언

always@(posedge clk or posedge rst)  // clock이나 reset이 상승edge일 때 작동
begin
 if(rst)                          // reset이 입력되면
  begin
   s = next_s;                    // 이전상태는 reset을 입력하기 전 상태
   next_s = 4'b0001;              // 다음상태는 초기상태 S_0(0001)로 돌아간다.
   la= 2'b00;                     // L_A는 green
   lb= 2'b10;                     // L_B는 red
  end
```

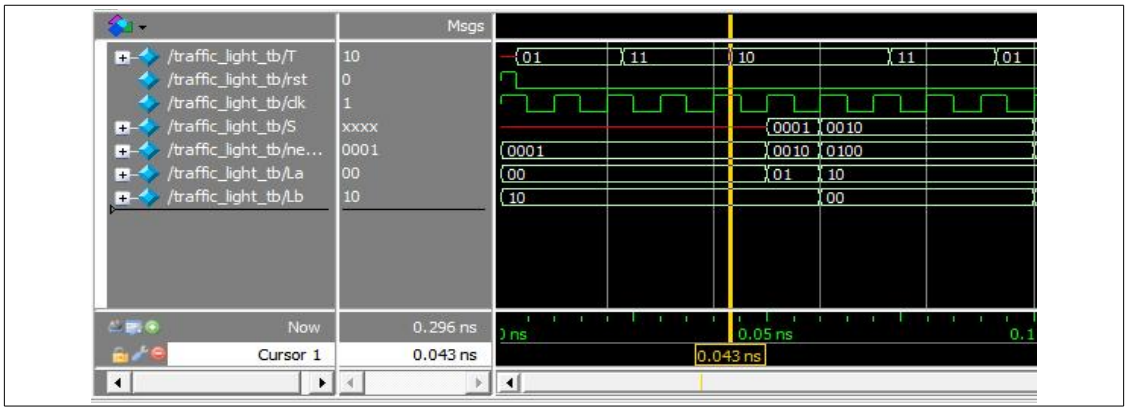
```
 else if(T[0]==0 && next_s==4'b0001)  // T_B와 상관없이 T_A가 0이고 상태가 S_0이면
  begin
   s = next_s;              // 이전상태는 S_0(0001)로 바뀌고
   next_s = next_s<<1;      // 다음상태는 S_1(0010)로 바뀐다. shift 사용
   la= 2'b01;               // L_A는 yellow
   lb= 2'b10;               // L_B는 red
  end

 else if(next_s==4'b0010)     // T_A, T_B와 상관없이 상태가 S_1이면
  begin
   s = next_s;              // 이전상태는 S_1(0010)로 바뀌고
   next_s = next_s<<1;      // 다음상태는 S_2(0100)로 바뀐다.
   la= 2'b10;               // L_A는 red
   lb= 2'b00;               // L_B는 green
  end

 else if(t[1]==0 && next_s==4'b0100)  // T_A와 상관없이 T_B가 0이고 상태가 S_2이면
  begin
   s = next_s;              // 이전상태는 S_2(0100)로 바뀌고
   next_s = next_s<<1;      // 다음상태는 S_3(1000)로 바뀐다.
   la= 2'b10;               // L_A는 red
   lb= 2'b01;               // L_B는 yellow
  end
else if(next_s==4'b1000)    // T_A, T_B와 상관없이 상태가 S_3이면
  begin
   s = next_s;              // 이전상태는 S_3(1000)로 바뀌고
   next_s = next_s>>3;      // 다음상태는 S_1(0001)로 바뀐다.
   la= 2'b00;               // L_A는 green
   lb= 2'b10;               // L_B는 red
  end
end

assign S=s;                 // reg s는 output S로 연결
assign next_S=next_s;       // reg next_s 는 output next_S로 연결
assign La=la;               // reg la는 output La로 연결
assign Lb=lb;               // reg lb는 output Lb로 연결

endmodule
```

Testbench 코드

```
module traffic_light_tb;
reg [1:0]T;
reg rst,clk;
wire [3:0]S,next_S;
wire [1:0]La,Lb;

traffic_light t(.rst(rst),.clk(clk),.T(T),.S(S),.next_S(next_S),.La(La),.Lb(Lb));
 // traffic_light 모듈과 testbench를 연동시킨다.

initial
begin
clk = 1;
forever #5 clk=~clk;
end                          // clock을 0.01ns주기로 준다.

initial
begin
rst=1; #3;                  // reset을 입력시켜 출력을 초기화
rst=0; T=2'b01; #20;        // 0.02ns동안 T_B=0, T_A=1 입력
rst=0; T=2'b11; #20;        // 0.02ns동안 T_B=1, T_A=1 입력
rst=0; T=2'b10; #30;        // 0.02ns동안 T_B=1, T_A=0 입력
rst=0; T=2'b11; #20;        // 0.02ns동안 T_B=1, T_A=1 입력
rst=0; T=2'b01; #30;        // 0.03ns동안 T_B=0, T_A=1 입력
rst=0; T=2'b11; #20;        // 0.02ns동안 T_B=1, T_A=1 입력
rst=0; T=2'b10; #20;        // 0.02ns동안 T_B=1, T_A=0 입력
rst=0; T=2'b11; #20;        // 0.02ns동안 T_B=1, T_A=1 입력
rst=1; #3;                  // reset을 입력시켜 출력 초기화
rst=0;                      // reset해제
#100 $stop;                 // 0.1ns 뒤에 시뮬레이션 종료
end
endmodule
```

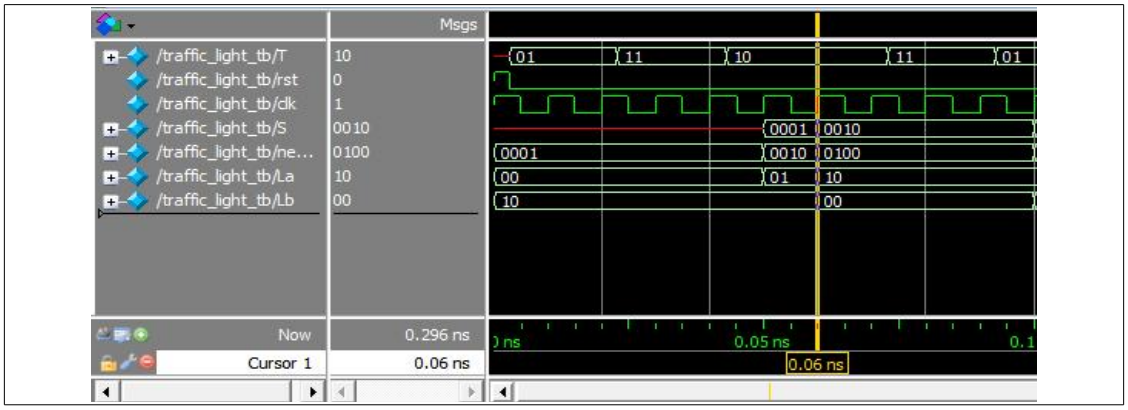
4. 실험결과

1) modelsim simulation 결과 화면

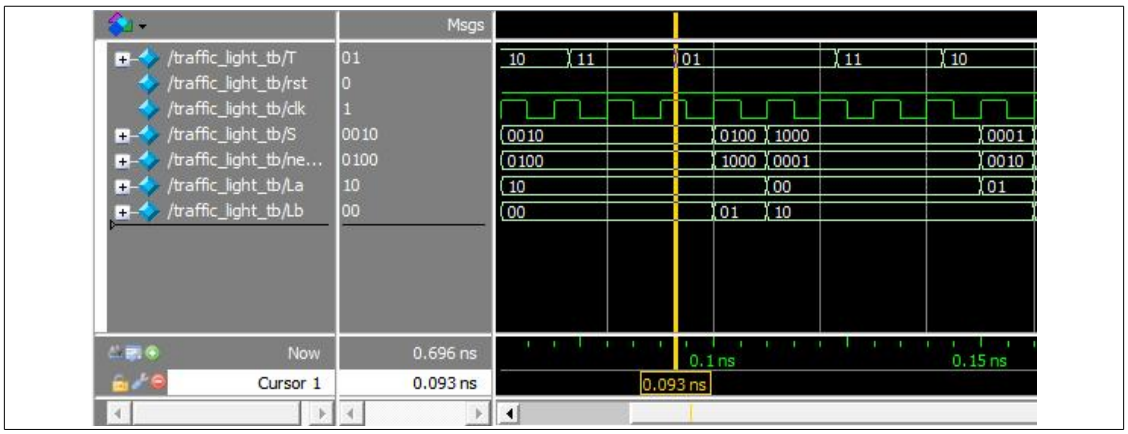


- 0.003ns동안 reset을 1로 줌으로써 초기출력을 S_0(0001)로 초기화 시킨다. 그 다음 0.02ns동안 입력을 T_A=1, T_B=0로 입력시켜주고, 다음 0.02ns동안 T_A=1, T_B=1을 입력시킴으로서 T_B의 값과 상관없이 T_A의 값이 1이면 S_0의 다음상태는 S_0로 가는 것을 볼 수 있다. 이전 상태는 모르기 때문에 무관항(xxxx)가 나오는 것을 알 수 있다. 상태가 S_0일 때 L_A는 green(00)이고 L_B는 red(10)임을 보여준다.

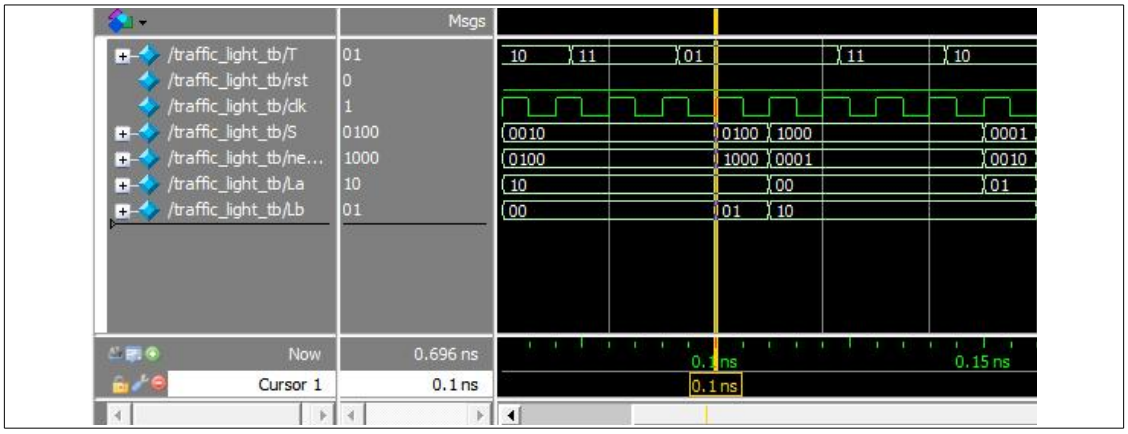
- 입력을 T_A=0로 입력, 입력하고 나서 바로 다음 clock의 상승 edge에서 상태가 S_0(0001)에서 S_1(0010)으로 바뀌고, 이전 상태는 S_0로 표시되는 것을 알 수 있다. 상태가 S_1일 때 L_A는 yellow(01)이고 L_B는 red(10)임을 보여준다.



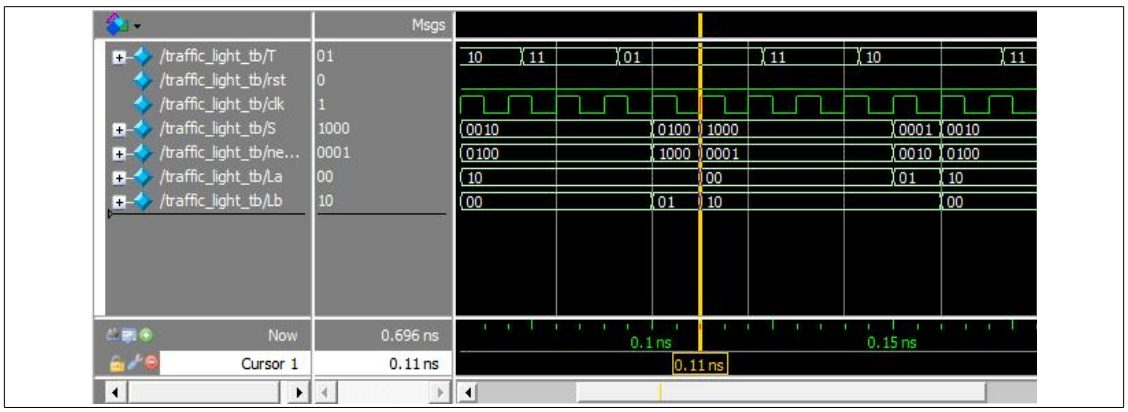
- 상태가 S_1(0010)이 되면 입력과 상관없이 바로 다음 clock에서 S_2(0100)로 바뀌는 것을 볼 수 있다. 이전 상태는 S_1로 표시되는 것을 알 수 있다. 상태가 S_2일 때 L_A는 red(10)이고 L_B는 green(00)임을 보여준다.



- 상태가 S_2(0100)일 때, 입력이 T_A=0, T_B=1와 T_A=1, T_B=1 일 때 상태가 S_2로 유지 되는 것을 알 수 있다. 이것은 T_A의 값과 상관없이 T_B가 1이면 S_2로 유지 되는 것을 보여준다. 따라서 이전 상태도 S_1으로 유지가 되고, L_A와 L_B도 바뀌지 않는 것을 알 수가 있다.



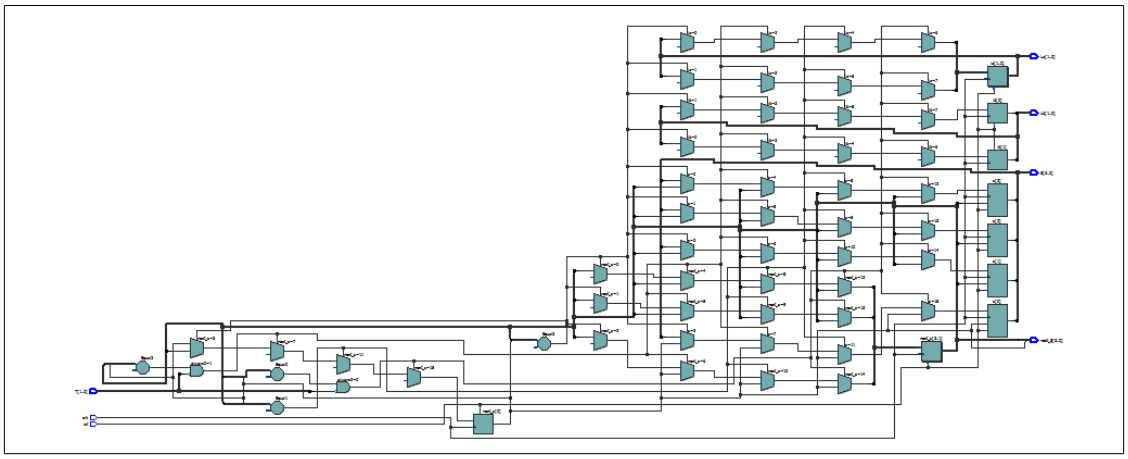
- 입력을 T_B=0로 입력, 입력하고 나서 바로 다음 clock의 상승 edge에서 상태가 S_2(0100)에서 S_3(1000)로 바뀌고, 이전 상태는 S_2로 표시되는 것을 알 수 있다. 상태가 S_3일 때 L_A는 red(10)이고 L_B는 yellow(01)임을 보여준다.



- 상태가 S_3(1000)이 되면 입력과 상관없이 바로 다음 clock에서 S_0(0001)로 바뀌는 것을 볼 수 있다. 이전 상태는 S_3로 표시되는 것을 알 수 있다. 그 뒤로는 다시 정상 작동됨을 알 수 있다.

- 다음은 무작위의 상태에서 reset이 입력되는 경우를 보여준다. clock과 비동기식이기 때문에 clock과 상관없이 reset이 입력되면 출력을 초기화돼서 초기값 S_0(0001)로 바뀌는 것을 보여준다. 위의 타이밍도를 보면 reset이 입력되기 전까지 상태가 S_2이기 때문에 이전상태는 S_2가 나옴을 알 수 있다.

2) RTL 회로 결과



5. 결과

1) 실험 내용 요약

- 입력 T_A와 T_B의 편하게 사용하기 위해 변수를 2-bit T로 사용하였다. T[0]은 T_A을 가르키고, T[1]은 T_B을 가르킨다. 출력 상태 S_0, S_1, S_2, S_3의 다음상태를 저장하기 쉽게 하기위해 4-bit 2진수로 표현한 다음 shift를 사용하였다. S_0=0001, S_1=0010, S_2=0100, S_3=1000 로 설정해줘서 생각하기 좀 쉽게 했다. 그리고 신호등이 비상상황일 때는 clock과 상관없이 reset이 동작이 되어야 하기 때문에 clock과 reset은 always문에서 비동기식으로 설정해주었다.

2) 실험 결과 요약

- 처음에는 타이밍도에 이전상태와 다음상태를 표시 안했었다. 그러다가 이전상태와 다음상태를 타이밍도에 표시해줄 때 만약에 무작위의 상태에서 reset을 입력시키면 이전상태 표시는 어떻게 해야할지 고민하다가 상태가 바뀌고 나서의 이전상태는 전 상태에서 넘어가는 다음상태로 저장되는 것을 이용하여서 해결하였다. 그래서 reset이 입력되고도 이전상태는 현재의 전 상태로 표시가 되게 하는 것을 구현할 수 있었다.