REPORT

보고서 작성 서약서

1. 나는 타학생의 보고서를 복사(Copy)하지 않았습니다.

2. 나는 타학생의 보고서를 인터넷에서 다운로드 하여 대체하지 않았습니다.

3. 나는 타인에게 보고서 제출 전에 보고서를 보여주지 않았습니다.

4. 보고서 제출 기한을 준수하였습니다.

나는 보고서 작성시 위법 행위를 하지 않고,
성.균.인으로서 나의 명예를 지킬 것을 약속합니다.

과 목 명 : 논리회로실험

담당 교수: 전 일 용

주　　차 : 10 주차

학　　과 : 건설환경공학부

학　　번 : 2019314487

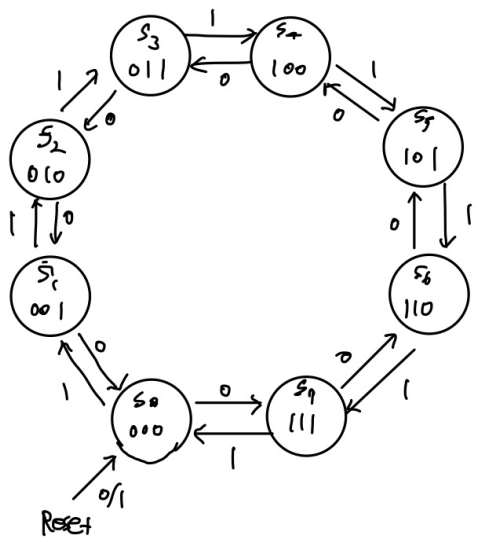
이　　름 : 김 경 수

1.Purpose of experiment

- Moore machine과 mealy machine을 포함한 fsm 동작의 메커니즘을 이해하고, 각각의 방법으로 3-bit up-down counters을 설계한다. 또한 testbench를 직접 설계하여 3-bit up counters의 작동을 검증한다.

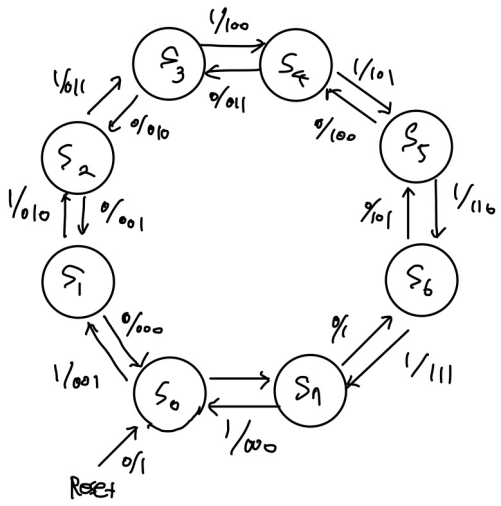
2. Theoretical Approach

1)Moore machine



Explain: Moore machine output은 current state에 의해 결정되는데, reset이 1이라면 machine은 작동하지 않지만 reset이 1일 경우 mode에 의하여 machine이 작동하는데 mode가 1일 경우 up counter, mode가 0일 경우 down counter로 작동하도록 설계한다. 또한 3 bit counter을 설계하려면 3개의 bit 자리와 8개의 state가 필요하다. 그래서 위의 내용을 요약해 Moore machine state diagram을 표현하면 위와 같다.

2)Mealy machine



Explain: Mealy machine 역시 reset이 0로 될 때 작동하며, output은 state와 input에 의존한다. Moore machine과 마찬가지로 8개의 state상태가 필요하며, input이 1일 때 state가 1씩 증가하며 up counter로 작동한다. Input이 0일 때는 state가 1씩 감소하면 down counter로 작동한다. 그래서 위의 내용을 요약해 Mealy machine state diagram을 표현하면 위와 같다.

3.Verilog implementation

1) 3-bit up down counter (Moore machine)

C:/2019314487w10/moore_3_bit_up_down_counter.v (/tb_w11/moore_counter) - Default *

```
Ln#
 1  module moore_3_bit_updown_counter (clk, mode,rst, out, state, next_state);
 2
 3    input clk, mode, rst;
 4
 5    output [2:0] out, state, next_state;
 6    reg    [2:0] out, state, next_state;
 7
 8    //States
 9    parameter S0 = 0, S1 = 1, S2 = 2, S3 = 3, S4 = 4, S5 = 5, S6 = 6, S7 = 7;
10
11    //Determine the output depending on the current state only
12    always @ (state)begin
13      case(state)
14        S0 : out = S0;
15        S1 : out = S1;
16        S2 : out = S2;
17        S3 : out = S3;
18        S4 : out = S4;
19        S5 : out = S5;
20        S6 : out = S6;
21        S7 : out = S7;
22        default : out = S0;
23      endcase
24    end
25
26    //Update the current state
27    always @ (posedge clk)
28    begin
29      if (rst == 1'b1)
30        state = S0;
31      else
32        state = next_state;
33    end
34
```

Explain: 주어진 2-bit counter의 예시를 참조해 3-bit up down counter를 설계하면 clk,

mode, rst 가 3-bit counter의 input이 된다. 또한 output과 reg는 out, state, next_state 가 되며, 8개의 state를 표현하기 위해 always 문과 case 문으로 각각 output으로 S0~S7까지를 설정하고 000부터 111을 할당한다. 또한 current state는 positive edge에서 rst= 0 라면 next_state 로 넘어가게 하였고 rst=1이라면 S0으로 초기화 되게 설정하였다.

```
//Determine the next state
always @ (posedge clk)
begin
  if (rst == 1'b1)
    next_state <= S0;

  else
  begin
      if (mode)
       begin
          case(state)
            S0 : next_state <= S1;
            S1 : next_state <= S2;
            S2 : next_state <= S3;
            S3 : next_state <= S4;
            S4 : next_state <= S5;
            S5 : next_state <= S6;
            S6 : next_state <= S7;
            S7 : next_state <= S0;
            default : next_state <= S0;
          endcase
      end

      else
      begin
        case(state)
            S0 : next_state <= S7;
            S1 : next_state <= S0;
            S2 : next_state <= S1;
            S3 : next_state <= S2;
            S4 : next_state <= S3;
            S5 : next_state <= S4;
            S6 : next_state <= S5;
            S7 : next_state <= S6;
            default : next state <= S0;
```

```
70              endcase
71            end
72        end
73      end
74    endmodule
```

Explain: Next stat는 d위에서 언급했듯이 rst=0 일 때 moore machine에 따라 작동하며 rst=0 일 때는 s0로 초기화 되도록 if else 문을 사용해 설계하였다. 또한 mode=1(true) 일 때는 up counter, mode=0(false) 일 때는 down counter의 역할을 수행하도록 if문 안에 mode를 넣고 그 안에서 case문을 활용하여 3-bit-counter by moore 설계를 완료하였다.

2) 3-bit up down counter (Mealy machine)

C:/2019314487w10/mealy_3_bit_up_down_counter.v (/tb_w11/mealy_counter) - Default

```
Ln#
 1  module mealy_3_bit_updown_counter (clk, mode, rst, out, state);
 2
 3    input clk, mode, rst;
 4
 5    output [2:0] out, state;
 6    reg [2:0] out, state;
 7
 8    //states
 9    parameter S0 = 0, S1 = 1, S2 = 2, S3 = 3, S4 = 4, S5 = 5, S6 = 6, S7 = 7;
```

Explain: moore machine과 input 과 parameter은 같지만 output은 next state를 포함하지 않는다. (mealy machine 특징)

```
Ln#
11    //Determine the output depending on the current state as well as the incoming input
12    always @ (state or mode)
13    begin
14      if (mode)
15      begin
16        case(state)
17          S0 : out = S1;
18          S1 : out = S2;
19          S2 : out = S3;
20          S3 : out = S4;
21          S4 : out = S5;
22          S5 : out = S6;
23          S6 : out = S7;
24          S7 : out = S0;
25          default : out = S0;
26        endcase
27      end
28
29      else
30      begin
31         case(state)
32           S0 : out = S0;
33           S1 : out = S1;
34           S2 : out = S2;
35           S3 : out = S3;
36           S4 : out = S4;
37           S5 : out = S5;
38           S6 : out = S6;
39           S7 : out = S7;
40           default : out = S0;
41        endcase
42      end
43    end
```

Explain: Output은 current stated와 input에 의해 결정. Always 문으로 state와 mode를 설계하였다.

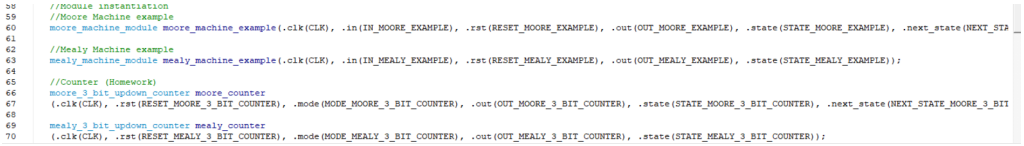
```
//Determine the next state
always @ (posedge clk) begin

  if (rst == 1'b1)
    state <= S0;
  else
  begin
      if (mode == 1'b1)
      begin
          case(state)
            S0 : state <= S1;
            S1 : state <= S2;
            S2 : state <= S3;
            S3 : state <= S4;
            S4 : state <= S5;
            S5 : state <= S6;
            S6 : state <= S7;
            S7 : state <= S0;
          endcase
      end

      else
      begin
        case(state)
            S0 : state <= S7;
            S1 : state <= S0;
            S2 : state <= S1;
            S3 : state <= S2;
            S4 : state <= S3;
            S5 : state <= S4;
            S6 : state <= S5;
            S7 : state <= S6;
        endcase
      end
```

Explain: 마찬가지로 if 문을 활용해 positive clk, rst=0에서 mealy machine이 작동하도록 설계하였고, mode=1 일 때 up-counter, mode=0 일 때 down counter로 설계하였다. 현재 상태는 위와 마찬가지로 case 문을 활용해 표현하였다.

3)Test bench

```
39      //Counter (Homework)
40
41      //input moore machine counter
42      reg RESET_MOORE_3_BIT_COUNTER;
43      reg MODE_MOORE_3_BIT_COUNTER;
44
45      //input mealy machine counter
46      reg RESET_MEALY_3_BIT_COUNTER;
47      reg MODE_MEALY_3_BIT_COUNTER;
48
49      //output moore machine counter
50      wire [2:0] OUT_MOORE_3_BIT_COUNTER;
51      wire [2:0] STATE_MOORE_3_BIT_COUNTER;
52      wire [2:0] NEXT_STATE_MOORE_3_BIT_COUNTER;
53
54      //output mealy machine counter
55      wire [2:0] OUT_MEALY_3_BIT_COUNTER;
56      wire [2:0] STATE_MEALY_3_BIT_COUNTER;
```

Explain: reg와 wire을 사용하여 input과 output을 설정하였다.



```
58    //Module instantiation
59    //Moore Machine example
60    moore_machine_module moore_machine_example(.clk(CLK), .in(IN_MOORE_EXAMPLE), .rst(RESET_MOORE_EXAMPLE), .out(OUT_MOORE_EXAMPLE), .state(STATE_MOORE_EXAMPLE), .next_state(NEXT_STA
61
62    //Mealy Machine example
63    mealy_machine_module mealy_machine_example(.clk(CLK), .in(IN_MEALY_EXAMPLE), .rst(RESET_MEALY_EXAMPLE), .out(OUT_MEALY_EXAMPLE), .state(STATE_MEALY_EXAMPLE));
64
65    //Counter (Homework)
66    moore_3_bit_updown_counter moore_counter
67    (.clk(CLK), .rst(RESET_MOORE_3_BIT_COUNTER), .mode(MODE_MOORE_3_BIT_COUNTER), .out(OUT_MOORE_3_BIT_COUNTER), .state(STATE_MOORE_3_BIT_COUNTER), .next_state(NEXT_STATE_MOORE_3_BIT
68
69    mealy_3_bit_updown_counter mealy_counter
70    (.clk(CLK), .rst(RESET_MEALY_3_BIT_COUNTER), .mode(MODE_MEALY_3_BIT_COUNTER), .out(OUT_MEALY_3_BIT_COUNTER), .state(STATE_MEALY_3_BIT_COUNTER));
```

```
72    initial
73    begin
74       CLK = 1'b0;
75       RESET_MOORE_EXAMPLE = 1'b1;
76       IN_MOORE_EXAMPLE  = 1'b0;
77       RESET_MEALY_EXAMPLE = 1'b1;
78       IN_MEALY_EXAMPLE  = 1'b0;
79       //Counter (Homework)
80
81       //setting reset=1 and mode=0
82       RESET_MOORE_3_BIT_COUNTER = 1'b1;
83       MODE_MOORE_3_BIT_COUNTER = 1'b0;
84       RESET_MEALY_3_BIT_COUNTER  = 1'b1;
85       MODE_MEALY_3_BIT_COUNTER = 1'b0;
86    end
```

Explain: 위에서 생성한 3-bit-countr 의 두가지 모듈(moore, mealy machine)을 instance 화 하고, 초기값 설정을 위해 reset=1 로 mode=0 로 두어 초기화 시킨다.

```
99    begin
100     //test patterns
101     //EXAMPLE
102     #20 RESET_MOORE_EXAMPLE = 1'b0;
103       IN_MOORE_EXAMPLE  = 1'b1;
104       RESET_MEALY_EXAMPLE = 1'b0;
105       IN_MEALY_EXAMPLE  = 1'b1;
106
107     #100 IN_MEALY_EXAMPLE  = 1'b0;
108       RESET_MOORE_EXAMPLE = 1'b1;
109
110
111     //3-bit counter
112     #100 RESET_MOORE_3_BIT_COUNTER = 1'b0;
113          MODE_MOORE_3_BIT_COUNTER = 1'b1;
114          RESET_MEALY_3_BIT_COUNTER = 1'b0;
115          MODE_MEALY_3_BIT_COUNTER = 1'b1;
116
117     #200 RESET_MOORE_3_BIT_COUNTER = 1'b0;
118          MODE_MOORE_3_BIT_COUNTER = 1'b0;
119          RESET_MEALY_3_BIT_COUNTER = 1'b0;
120          MODE_MEALY_3_BIT_COUNTER = 1'b0;
121
122
123     #300 RESET_MOORE_3_BIT_COUNTER = 1'b1;
124          RESET_MEALY_3_BIT_COUNTER = 1'b1;
125
126   end
127 endmodule
128
```

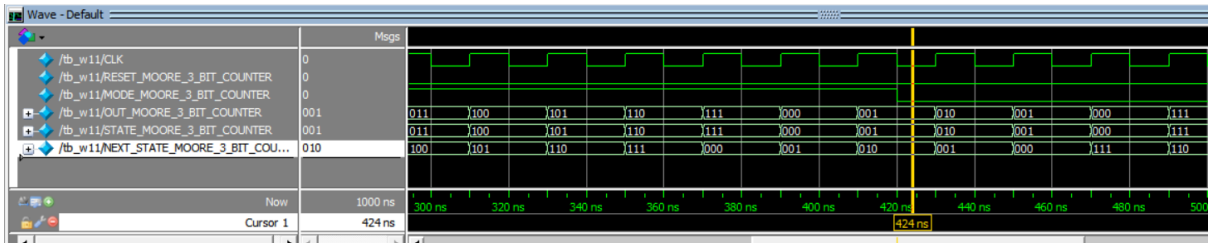
Explain: 위의 2-bit counter test 예시를 참조하여 test code 를 짰는데 처음 100ns 에 reset 을 1 에서 0 로 mode 를 0 에서 1 로 변화시켜 up- counter 로 작동하게 하고 200ns 에서는 mode=0 으로 변화시켜 down counter 의 동작을 확인했다. 마지막으로

300ns 에서는 reset 기능의 정상 작동을 확인하기 위해 reset=1 로 두어 counter 을 초기화 시켜 보았다.

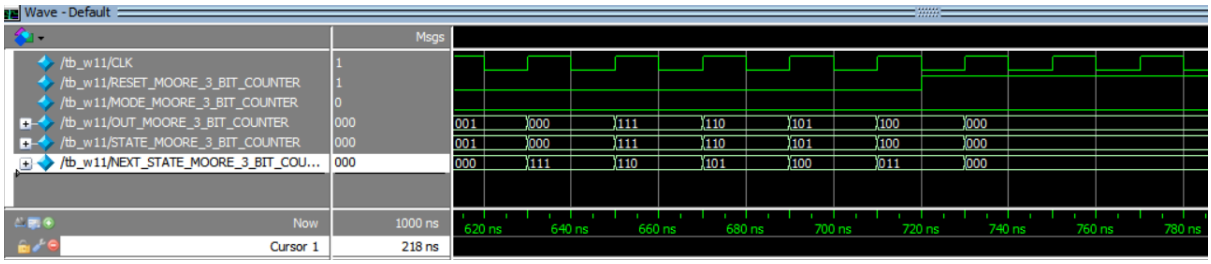
4.실험 결과

1)Moore machine

Explain: 초기 설정 값(reset=1, mode=0)에 의해 처음 그래프는 out이 모두 000로 나오는 것을 볼 수 있다. 이후 clk positive, reset= 0 mode=1로 바뀌며 moore machine의 기능이 up counter로 활성화되어 001, 010,011 식으로 증가하는 것을 확인할 수 있다.

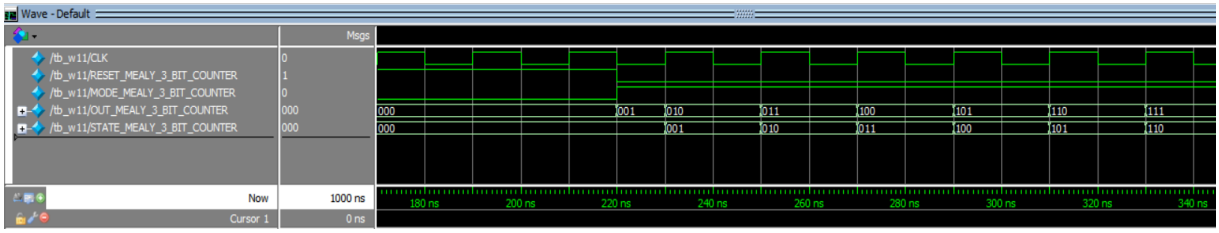


Explain: 이후 reset은 여전히 0이지만 mode=0로 바뀌며 down-counter mode로 변화할 것을 예상할 수 있고 이를 wave에서 확인해보면 노란선 이후 010에서 직전과 직후 bit 모두 001 인 것을 보아 up counter에서 down counter로 기능을 하는 것을 확인할 수 있다.

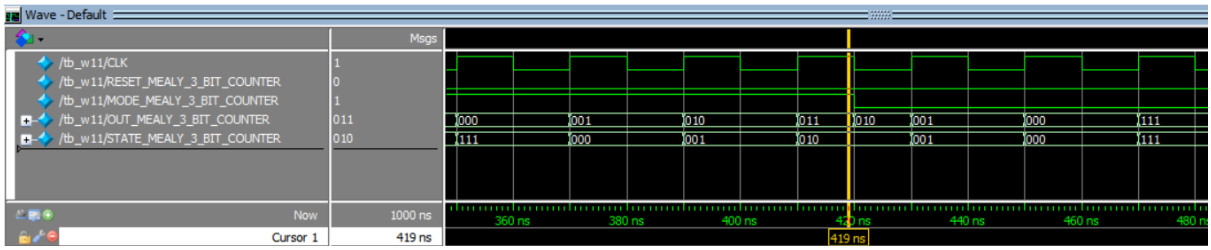


Explain: 마지막으로 reset=1 로 moore machine을 초기화함으로써 reset 기능의 정상 작동을 확인했는데 000으로 초기화가 잘 되는 것을 보아 설계를 제대로 한 것을 확인할 수 있었다.

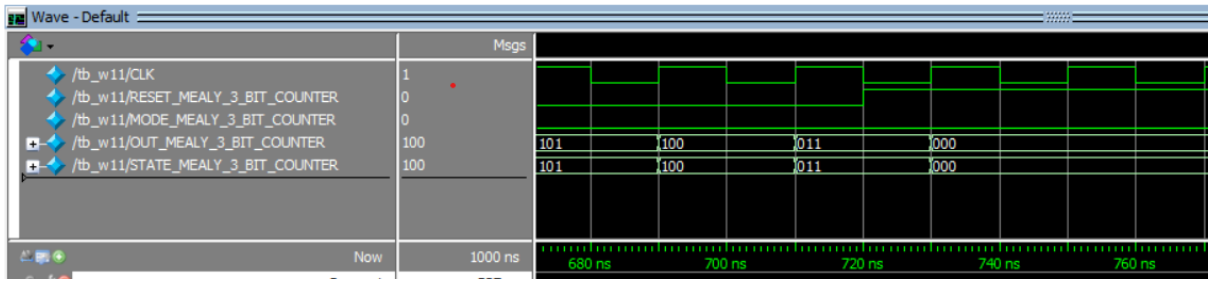
2)Mealy machine



Explain: Mealy machine 역시 설정은 moore machine과 비슷하지만 out이 current state와 mode 둘 다에 의해 결정됨으로 이것을 주의해서 시간 딜레이를 확인해본다.



Explain: 마찬가지로, positive clk, rst=0, mode=0에서 up-counter에서 down-counter으로 변화가 잘 일어나는 것을 확인 가능하다.



Explain: 마지막으로 reset=1 로 mealy machine을 초기화함으로써 reset 기능의 정상 작동을 확인하고 설계의 검증을 완료하였다.

5.Conclusion

- Moore machine과 mealy machine을 활용해 3-bit counter을 설계해보며 counter의 기능을 다시 한번 확인해보고 두가지 FSM의 state diagram을 그려보며 원리를 이해해 보았다. Mode, case, if else 문 등 이번 과제는 비교적 많은 구문을 사용했고 생각해야 할 변수 값도 많아 어려웠던 과제였다. 심지어 코드 오류도 정말 많이 나서 구상하는데 많은 시간이 걸렸는데, 이 과정을 통해 조금 더 Verilog 설계에 친숙해진 것 같다.