실험 Verilog-14: 결과보고서

전공: 컴퓨터공학 학년: 2 학번: 20171645 이름 박찬우

1. 목적

Verilog를 통해 moore 및 mealy machine, sequence detector에 대해 이해한다.

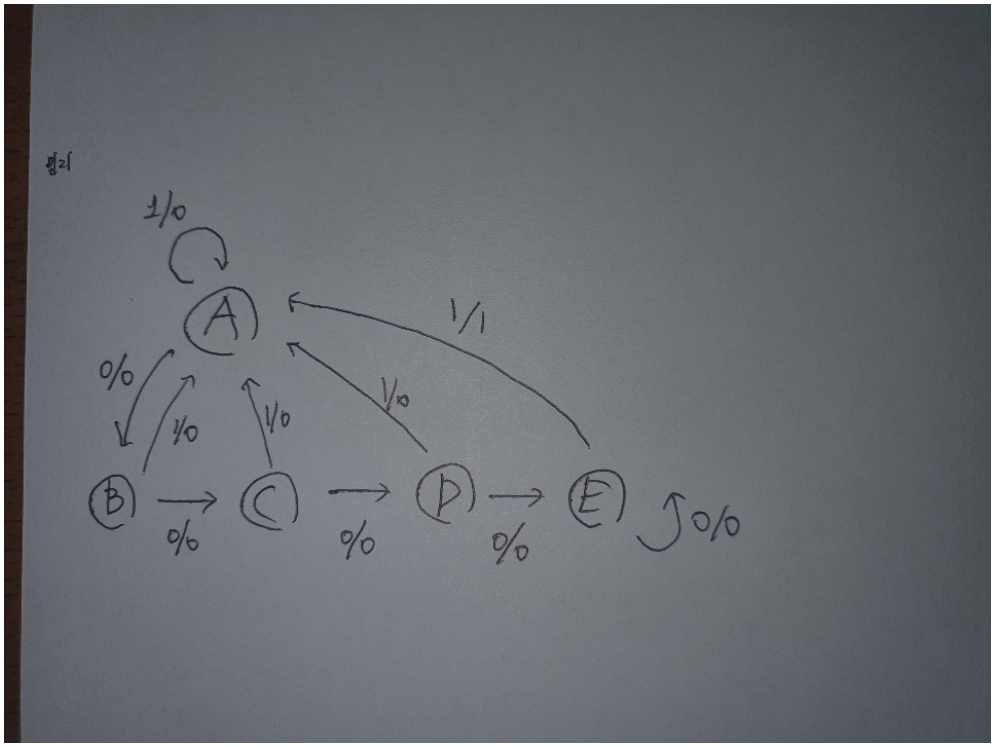
2. 요구 사항

1) 밀리 머신과 무어 머신의 특징과 차이점에 대해 서술하시오..

밀리 머신은 출력값이 입력값과 현재 상태 모두에 의존한다. 예를들면, 현재 상태가 S3라고 해도 입력값이 0일때와 1일때 출력값이 다를 수 있다는 것이다. 반면 무어 머신은 출력값이 오로지 현재 상태에 따라서만 결정된다. 즉, 현재 상태가 S3이면 입력이 0이든 1이든 출력값은 반드시 같다는 것이다. 이러한 두 FSM의 차이점 때문에, 밀리 머신은 상태의 수를 줄여야 하는 상황에 사용하고, 무어 머신은 행위를 단순화해야 하는 상황에 사용한다.

2) 밀리 머신을 사용하여 sequence detect 하는 예시를 만들고 설명하라.

밀리 머신을 사용하여 sequence '00001' 를 detect 하는 sequence detector를 만들어본다.
우선, 해당 밀리 머신의 State Diagram은 다음과 같다.



이를 State Table로 나타내면 아래와 같다.

상태	상태 배정	다음상태		출력	
		x=0	x=1	x=0	x=1
A	000	001	000	0	0
B	001	010	000	0	0
C	010	011	000	0	0
D	011	100	000	0	0
E	100	100	000	0	1

이와 같은 Truth table을 바탕으로 작성한 Verilog Code는 아래와 같다.

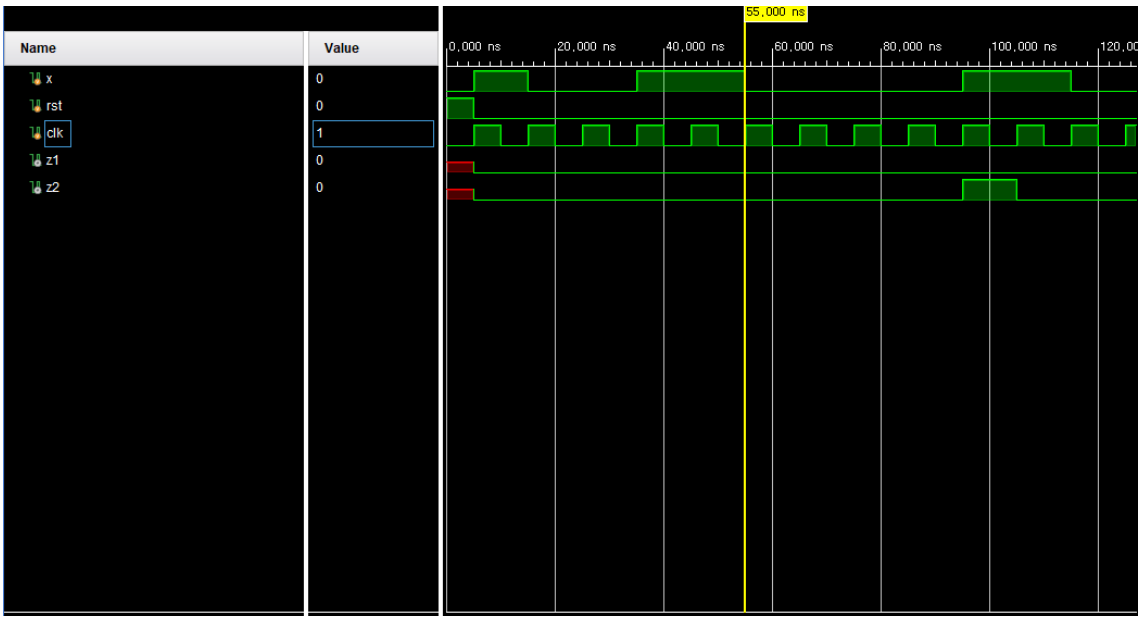
```
`timescale 1ns / 1ps
module shift_reg(input x,rst,clk, output reg z1, z2);
    reg     [2:0]     s0     =     3'b000,     s1     =     3'b001,     s2=3'b010,
s3=3'b011,s4=3'b100,s5=3'b101;
    reg [2:0] pstate;

    always@(posedge clk)
    begin
       if (rst) pstate <= 0;
       else
          case(pstate)
          s0 : if (x==1'b0) begin pstate <= s1; z1<=0; z2<=0; end
          else begin pstate <= s0; z1<=0; z2<=0; end
          s1 : if (x==1'b0) begin pstate <= s2; z1<=0; z2<=0; end
          else begin pstate <= s0; z1<=0; z2<=0; end
          s2 : if (x==1'b0) begin pstate <= s3; z1<=0; z2<=0; end
          else begin pstate <= s0; z1<=0; z2<=0; end
          s3 : if (x==1'b0) begin pstate <= s4; z1<=0; z2<=0; end
          else begin pstate <= s0; z1<=0; z2<=0; end
          s4 : if (x==1'b1) begin pstate <= s0; z1<=0; z2<=1; end
          else begin pstate <= s4; z1<=0; z2<=0; end
          default : begin pstate <= s0; z1<=0; z2<=0; end
          endcase
          end
endmodule
```

밀리 머신의 특징처럼 현재 상태 pstate와 입력값 x 두개를 바탕으로 다음 state를 결정하는 모습을 확인할 수 있다.
이제 아래와 같은 testbench code를 통해 simulation 결과를 확인해본다.

```
`timescale 1ns / 1ps
module tb_shiftreg4;
reg x;
reg rst;
reg clk;
wire z1, z2;
shift_reg t (.x(x), .rst(rst), .clk(clk), .z1(z1), .z2(z2));
```

```
initial begin
rst=1; x=0; clk=0;
#5 rst=0; x=1;
#10 x=0;
#10 x=0;
#10 x=1;
#10 x=1;
#10 x=0;
#10 x=0;
#10 x=0;
#10 x=0;
#10 x=1;
#10 x=1;
#10 x=0;
#10 x=0;
#10 x=0;
#10 x=0;
#10 x=1;
#10 x=1;
#10 x=1;
#10 x=0;
end
always begin
#5 clk = ~clk;
end
endmodule
```

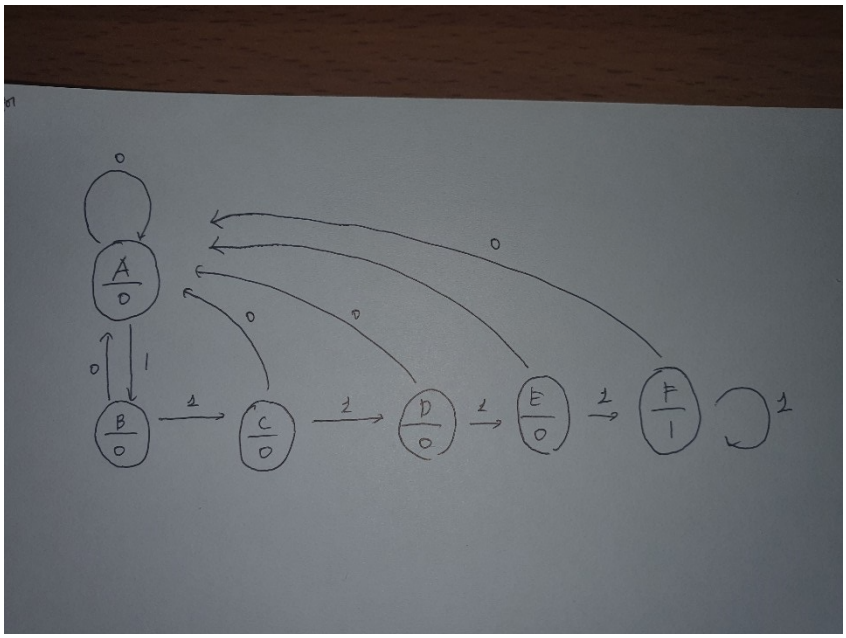


노란 선이 시작되는 위치에서부터 시작하여 X=0 인 상태에서 CLK가 4번 들어오고, 그 뒤 X값이 1로 변하자 결과값 d2 의 값이 1로 변하는 걸 확인할 수 있다. 이는 00001 이라는 Sequence를

Detect했다고 볼 수 있다. 이후 X값이 변화하자 다시 0으로 결과값이 변하는 모습 역시 확인할 수 있다.

3) 무어 머신을 사용하여 sequence detect 하는 예시를 만들고 설명하라.

무어머신을 사용하여 sequence '11111'을 detect 하는 sequence detector를 만들어본다.
우선, 해당 무어 머신의 State Diagram은 아래와 같다.



이를 Truth table로 나타내면 아래와 같다.

상태	상태 배정	다음상태		출력	
		x=0	x=1	x=0	x=1
A	000	000	001	0	0
B	001	000	010	0	0
C	010	000	011	0	0
D	011	000	100	0	0
E	100	000	101	0	1
F	101	000	101	0	1

이와 같은 Truth table을 통해 작성한 Verilog Code는 아래와 같다.

```
`timescale 1ns / 1ps
module shift_reg(input x,rst,clk, output reg z1, z2);
    reg     [2:0]     s0     =     3'b000,     s1     =     3'b001,     s2=3'b010,
s3=3'b011,s4=3'b100,s5=3'b101;
    reg [2:0] pstate;

    always@(posedge clk)
    begin
        if (rst) pstate <= 0;
        else
```

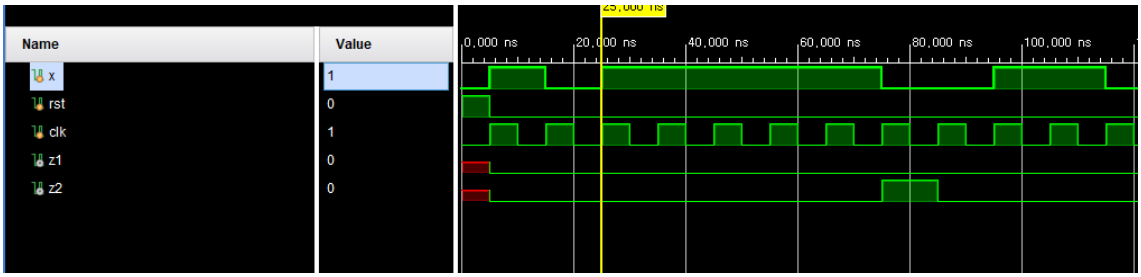
```
            case(pstate)
            s0 : if (x==1'b1) begin pstate <= s1; end
            else begin pstate <= s0; end
            s1 : if (x==1'b1) begin pstate <= s2;end
            else begin pstate <= s0; end
            s2 : if (x==1'b1) begin pstate <= s3; end
            else begin pstate <= s0; end
            s3 : if (x==1'b1) begin pstate <= s4; end
            else begin pstate <= s0; end
            s4 : if (x==1'b1) begin pstate <= s5; end
            else begin pstate <= s0; end
            s5 : if (x==1'b1) begin pstate <= s5; end
            else begin pstate <= s0; end
            default : begin pstate <= s0; z1<=0; z2<=0; end
            endcase

            case(pstate)
            s5 : begin z1<=0; z2<=1; end
            default : begin z1<=0; z2<=0; end
            endcase
            end
endmodule
```

앞선 밀리 머신의 코드와 달리, 무어 머신은 입력값에 관계없이 현재 상태에 의해서만 출력값이 결정된다. 따라서, 입력에 따른 상태의 변화와 상태에 따른 결과 출력을 case문 2개로 나누어 구현하였다. 때문에 입력값(x)은 결과를 출력하는 아래 case문에 영향을 줄 수 없다.
그 후, 아래 testbench code를 통해 simulation 해본다.

```
`timescale 1ns / 1ps
module tb_shiftreg4;
reg x;
reg rst;
reg clk;
wire z1, z2;
shift_reg t (.x(x), .rst(rst), .clk(clk), .z1(z1), .z2(z2));
initial begin
rst=1; x=0; clk=0;
#5 rst=0; x=1;
#10 x=0;
#10 x=1;
#10 x=1;
#10 x=1;
#10 x=1;
#10 x=1;
#10 x=0;
#10 x=0;
```

```
#10 x=1;
#10 x=1;
#10 x=0;
#10 x=0;
#10 x=0;
#10 x=0;
#10 x=1;
#10 x=1;
#10 x=1;
#10 x=0;
end
always begin
#5 clk = ~clk;
end
endmodule
```



노란 선에서부터 시작하여, 입력값 x의 값이 1인 상태로 CLK가 5회 깜빡이자 그 다음 CLK의 결과값이 1로 변하는 걸 확인할 수 있었다. 이는 11111 sequence를 detect한 결과로 볼 수 있다. 그 뒤 x값이 0으로 변하자 결과값 역시 0으로 변하는걸 확인할 수 있다.