Lab #4 Finite State Machine

Class : 00

201602004 박태현

I. 실습 목적

밀리 머신과 무어 머신의 설계

II. Design procedure

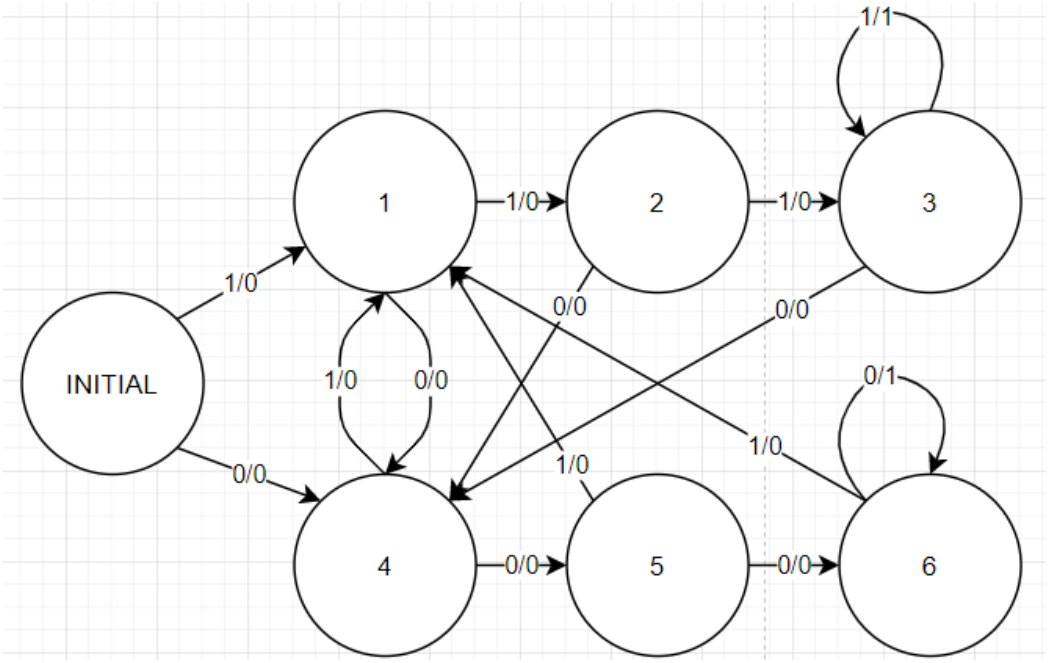
밀리 머신의 상태 전이도

현재 상태	입력	다음 상태	출력
INITIAL	1	I1	0
	0	O1	0
I1	1	I2	0
	0	O1	0
I2	1	I3	0
	0	O1	0
I3	1	I3	1
	0	O1	0
O1	1	I1	0
	0	O2	0
O2	1	I1	0
	0	O3	0
O3	1	I1	0
	0	O3	1

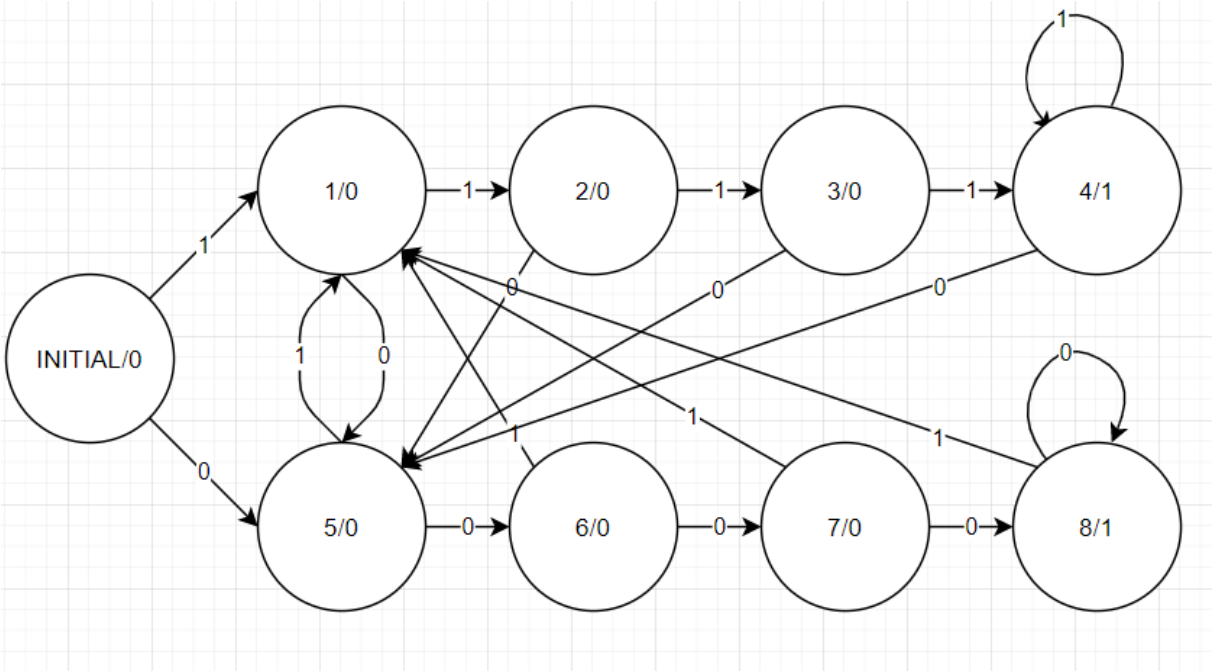
무어 머신의 상태 전이도

현재 상태	입력	다음 상태	출력
INITIAL	1	I1	0
	0	O1	
I1	1	I2	0
	0	O1	
I2	1	I3	0
	0	O1	
I3	1	I4	0
	0	O1	
I4	1	I4	1
	0	O1	
O1	1	I1	0
	0	O2	
O2	1	I1	0
	0	O3	
O3	1	I1	0
	0	O4	
O4	1	I1	1
	0	O4	

밀리 머신의 상태 다이어그램



무어 머신의 상태 다이어그램



III. Simulation

- Mealy Machine

```
`define INIT 3'b000
`define I1 3'b001
`define I2 3'b010
`define I3 3'b011
`define O1 3'b100
`define O2 3'b101
`define O3 3'b110

module mealy(nRESET, CLK, in, out);
   input nRESET, CLK, in;
   output out;
   reg [2:0] curState, nextState;
   reg out;

   always @ (posedge CLK or negedge nRESET)
      if (!nRESET) curState <= `INIT;
      else curState <= nextState;
```

```
always @ (curState or in)
   casex(curState)
      `INIT:
         if (in == 1) begin
            nextState = `I1;
            out = 1'b0;
         end
         else begin
            nextState = `O1;
            out = 1'b0;
         end
```

상태를 미리 define으로 정의를 해두고 클럭과 리셋에 맞춰 state를 업데이트 하도록 설계했습니다. 그리고 현재 상태 또는 입력이 변할 경우 다음 상태와 출력을 변하도록 했습니다.

```
    end
// ONE to THREE to process 1s
`I1:
    if (in == 1) begin
        nextState = `I2;
        out = 1'b0;
    end
    else begin
        nextState = `O1;
        out = 1'b0;
    end
`I2:
    if (in == 1) begin
        nextState = `I3;
        out = 1'b0;
    end
    else begin
        nextState = `O1;
        out = 1'b0;
    end
`I3:
    if (in == 1) begin
        nextState = `I3;
        out = 1'b1;
    end
    else begin
        nextState = `O1;
        out = 1'b0;
    end
    end
// FOUR to SIX to process 0s
`O1:
    if (in == 0) begin
        nextState = `O2;
        out = 1'b0;
    end
    else begin
        nextState = `I1;
        out = 1'b0;
    end
`O2:
    if (in == 0) begin
        nextState = `O3;
        out = 1'b0;
    end
    else begin
        nextState = `I1;
        out = 1'b0;
    end
`O3:
    if (in == 0) begin
        nextState = `O3;
        out = 1'b1;
    end
    else begin
        nextState = `I1;
        out = 1'b0;
    end
default: begin
    nextState = `INIT;
    out=1'b0;
end
```

상태 전이도에서 설계한 대로 각 state에서 (curState, in) 입력에 맞추어 (nextState, out)을 변경해주었습니다.

- Moore Machine

```
always @ (curState or in)
   casex(curState)
      `INIT: begin
         if (in == 1) nextState = `I1;
         else nextState = `O1;
         out = 1'b0;
      end
      // ONE to FOUR to process 1s
      `I1: begin
         if (in == 1) nextState = `I2;
         else nextState = `O1;
         out = 1'b0;
      end
      `I2: begin
         if (in == 1) nextState = `I3;
         else nextState = `O1;
         out = 1'b0;
      end
      `I3: begin
         if (in == 1) nextState = `I4;
         else nextState = `O1;
         out = 1'b0;
      end
      `I4: begin
         if (in == 1) nextState = `I4;
         else nextState = `O1;
         out = 1'b1;
      end
      ----
      // FIVE to EIGHT to process 0s
      `O1: begin
         if (in == 0) nextState = `O2;
         else nextState = `I1;
         out = 1'b0;
      end
      `O2: begin
         if (in == 0) nextState = `O3;
         else nextState = `I1;
         out = 1'b0;
      end
      `O3: begin
         if (in == 0) nextState = `O4;
         else nextState = `I1;
         out = 1'b0;
      end
      `O4: begin
         if (in == 0) nextState = `O4;
         else nextState = `I1;
         out = 1'b1;
      end
      default: begin
         nextState = `INIT;
         out=1'b0;
      end
```

무어 머신도 상태 전이도에 맞게 각 상태를 변경시켜주었습니다. (curState, in)에 맞게 nextState를 정의해주었습니다. nextState에 따른 out이 나오도록 했습니다.

밀리머신과 무어머신 모두 default 케이스를 통해 래치 합성을 방지했습니다

- Test bench

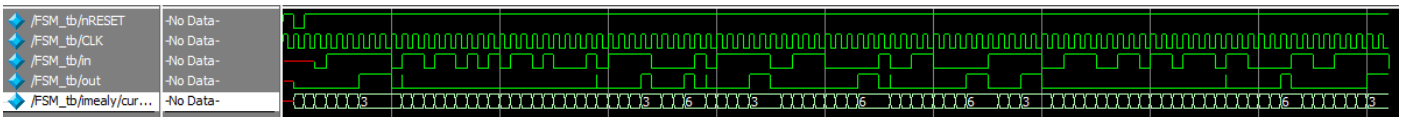
```
`timescale 1ns / 10ps

module FSM_tb;

reg CLK,nRESET, in;
wire out;

integer i;
parameter iter = 100;


always
   #2.5 CLK = ~CLK;

moore imoore(nRESET, CLK, in, out);
//mealy imealy(nRESET, CLK, in, out);
```

```
initial begin

$dumpvars;
CLK=1'b1;
nRESET = 1'b1;

#5 nRESET = 1'b0;
#5 nRESET = 1'b1;
for(i = 0; i < iter; i = i + 1)
   #5 in = $random;
$finish;
end
```

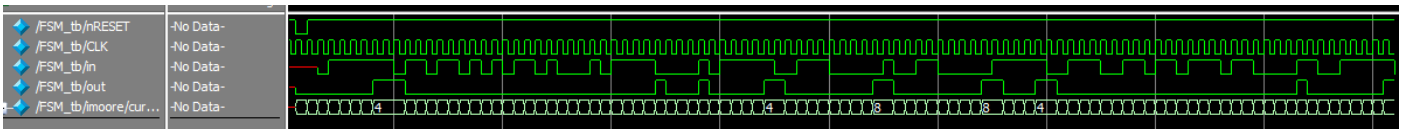
클럭이 2.5ns마다 뒤집어지므로 5ns마다 1클럭이 진행되도록 했습니다. 테스트를 시작하면 리셋을 주어 값을 INITIAL로 초기화 시키고, 임의의 값을 클럭에 맞추어 주었습니다.

- Waveform

밀리 머신

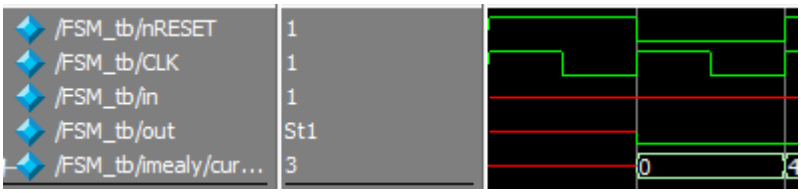


무어 머신

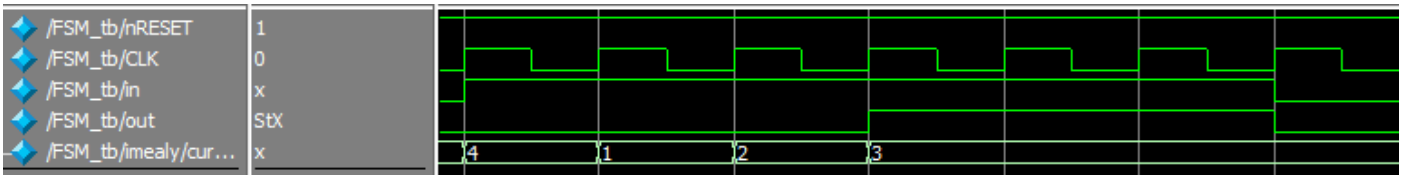


IV. Evaluation

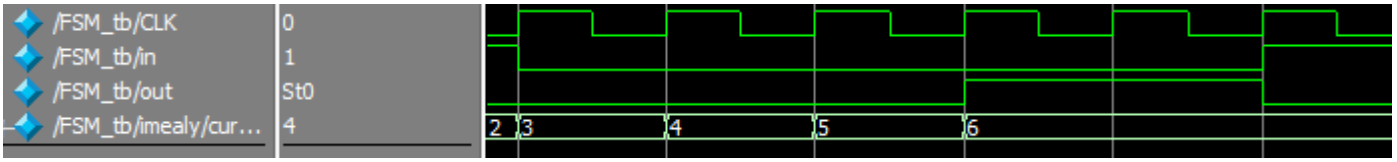
밀리 머신



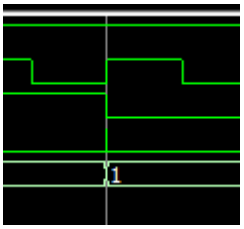
리셋이 들어오고 잘 초기화가 된 것을 확인했습니다



연속된 1이 들어오고 4개가 된 순간 out이 1이 되고 다시 0이된 순간 out이 0이 된 것을 확인했습니다

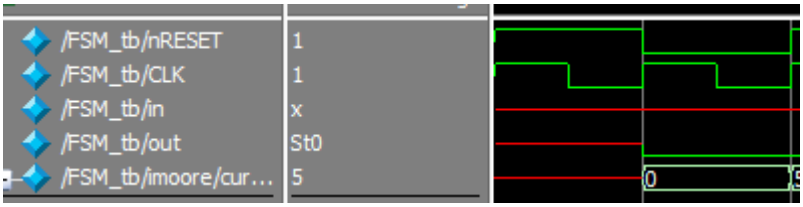


연속된 0이 4개 이상이 된 순간부터 out이 1이 된 것을 확인했습니다

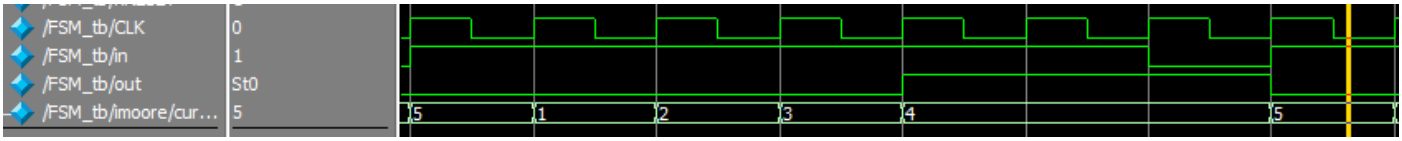


밀리 머신의 경우 글리치가 발생하는 문제가 있다는 것도 확인을 했습니다

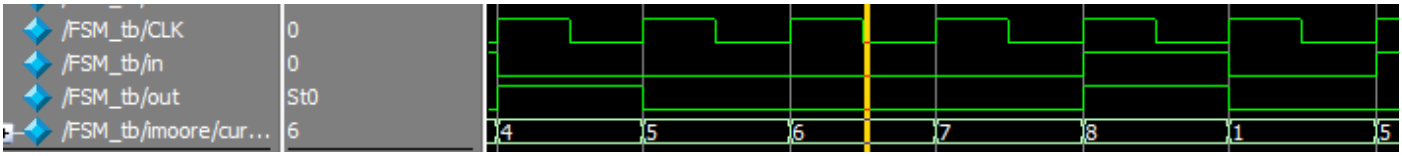
무어 머신



리셋이 들어오고 잘 초기화가 된 것을 확인했습니다



연속된 1 입력이 4개 들어오고 그 다음 사이클부터 총 3개의 사이클에서 out이 1로 출력된 것을 확인했습니다



연속된 0 입력이 4개 들어오고 그 다음 사이클에 out으로 1이 출력된 것을 확인했습니다

V. Discussions

밀리 머신과 무어 머신의 차이점을 알 수 있었습니다

무어 머신이 상태에 따른 출력값을 내보내서 한 사이클 늦게 결과가 나온다는 것을 알았습니다

밀리 머신이 입력값과 상태의 동기화로 인해 출력에서 글리치가 발생할 수 있다는 것을 알았습니다

상태 전이도를 그리지 않고 먼저 다이어그램을 그려서 헷갈렸습니다. 상태 전이도를 먼저 그렸으면 더 수월하게 과제를 할 수 있었을 것 같습니다