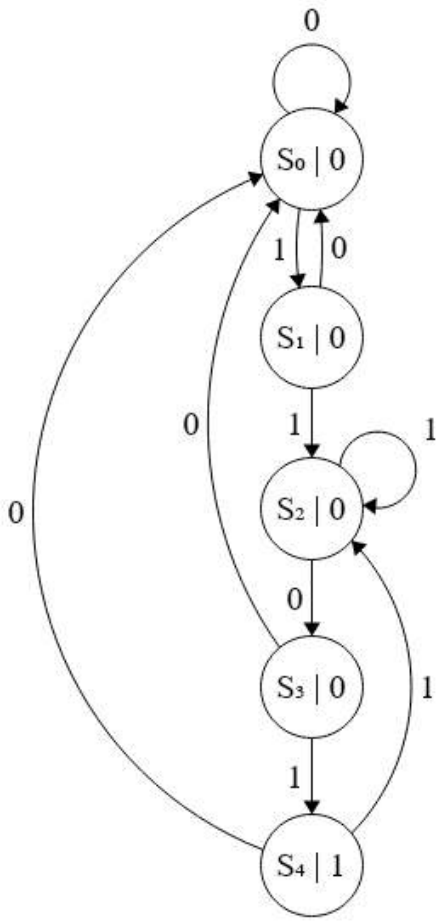
14주차 결과보고서

전공: 신문방송학과 학년: 4학년 학번: 20191150 이름: 전현길

1. Overlapping 방식의 Sequence Detector 1101 Moore machine 구현

(verilog source, simulation 결과, 상태도(State Table) 및 상태표(State Diagram) 작성)

sequence detector 1101 moore machine state diagram

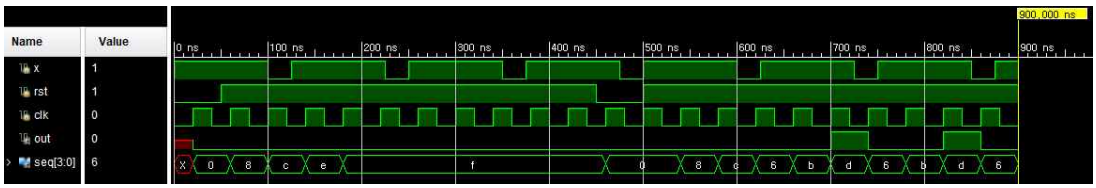


Q(t)	next state Q(t+1)	
	x = 0	x = 1
S_0	S_0	S_1
S_1	S_0	S_2
S_2	S_3	S_2
S_3	S_0	S_4
S_4	S_0	S_2

source code	testbench code

`timescale 1ns / 1ps
 	
module moore_1101_detector (
input x, clk, rst,
output reg out,
output reg [3:0] seq
);
parameter S0 = 3'b000, S1 = 3'b001, S2 = 3'b010, S3 = 3'b011, S4 = 3'b100;
reg [2:0] current_state, next_state;
 	
always@ (posedge clk) begin
if (!rst) begin // active-low
current_state <= S0;
out <= 1'b0;
seq <= 4'b0000;
end
else begin
out = 1'b0;
case (current_state)
S0: next_state = x ? S1 : S0;
S1: next_state = x ? S2 : S0;
S2: next_state = x ? S2 : S3;
S3: next_state = x ? S4 : S0;
S4: begin
next_state = x ? S2 : S0;
out = 1'b1;
end
default: next_state = S0;
endcase
// sequence 출력
seq[3] <= x;
seq[2] <= seq[3];
seq[1] <= seq[2];
seq[0] <= seq[1];
// 상태 전환
current_state <= next_state;
end
end
 	
endmodule
	`timescale 1ns / 1ps
	
	module moore_1101_detector_tb;
	
	reg x, rst, clk;
	wire out;
	wire [3:0] seq;
	
	moore_1101_detector u_test(
	.x (x), .rst (rst), .clk (clk),
	.out (out), .seq (seq)
	);
	
	initial begin
	x = 1'b1; rst = 1'b0; clk = 1'b0;
	// 시작할 때 값을 reset
	forever #20 clk = ~clk;
	end
	
	always@ (x) begin
	#100 x <= 1'b0;
	#25 x <= 1'b1;
	end
	
	initial begin
	#50 rst = 1'b1;
	#300
	#100 rst = 1'b0;
	#50 rst = 1'b1;
	#400
	$finish;
	end
	
	endmodule

1101 sequence detector moore machine simultation



sequence detector란 연속적으로 입력되는 bit sequence에서 특정한 pattern의 bit sequence를 검출하는 소자이다.

이번에 구현할 1101 sequence detector moore machine의 경우 입력 sequence에서 '1101'을 검출하면 High 신호를 출력하고, 그 외에는 Low를 출력한다. moore model로 구현했기 때문에 입력값과 무관하게 현재 상태에 따라 값을 출력하며, 5가지 상태 S_0-S_4 를 갖는다.

Verilog 코드 상의 구현을 위해 always @(posedge clk)문을 사용해 클럭이 상승 엣지일 때 입력값 x의 현재 값에 따라 상태를 변화시키도록 했다. 현재

상태와 다음 상태를 저장하기 위해 3bit register 변수 reg[2:0] current_register, next_register 변수를 선언했다. 입력값과 현재 상태에 따라 다음 상태를 결정하기 위해서는 case (current_case)문을 사용했다.

parameter문으로 선언된 S0, S1, S2, S3, S4는 각 상태 bit에 이름을 적어준 것인데, 코드의 가독성을 높이기 위해 사용했다. parameter를 선언하지 않고 대신 3'b000, 3'b111 등을 그대로 사용하더라도 동일하게 동작한다. 또 case별 상태의 변화를 간결하게 나타내기 위해서 삼항 연산자를 사용했다.

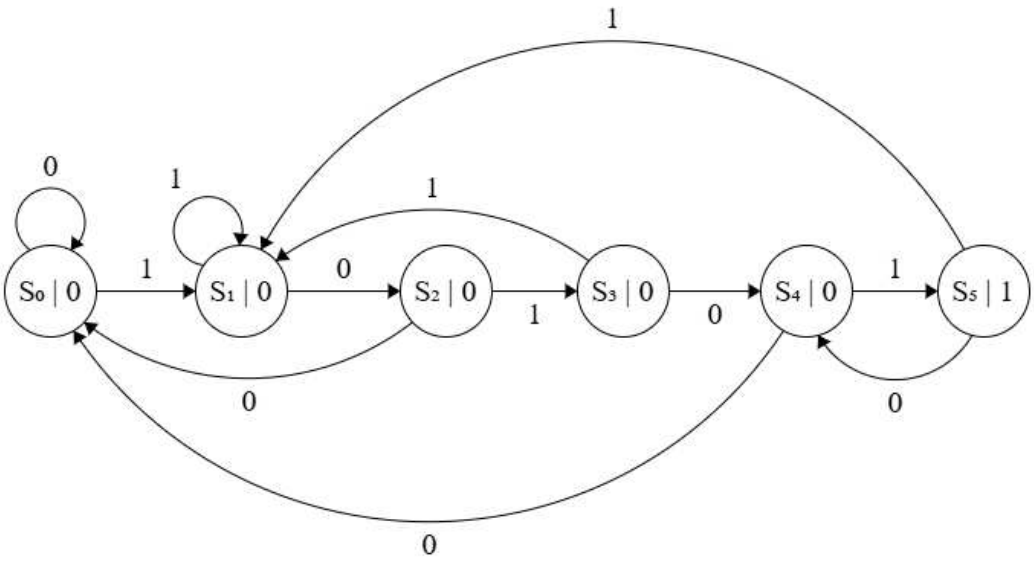
다르게 구현할 수도 있다. next_state 변수를 생략하고 current_state를 즉시 바꿔 3bit register를 절약할 수도 있다. 또 입력된 sequence에는 관심이 없고 '1101'이 입력됐는지에만 관심이 있다면 seq에 쓰이는 4bit register를 절약할 수도 있다. 입력된 seq 변수를 이용해 if (seq == 4b'1101)문을 사용해 1101이 입력되었는지 조사할 수도 있다. 단 이런 구현의 경우 FSM의 개념과는 조금 거리가 있다.

시뮬레이션 결과, rst 값에 따라 회로 상태의 초기화가 정상적으로 이루어졌고, '1101'이 입력되었을 때에만 out 변수가 set되는 것을 확인할 수 있었다. 회로가 1101 sequence detector의 논리적 동작과 동일하게 동작하므로 성공적으로 구현된 것을 확인할 수 있었다.

2-1. Overlapping 방식의 Sequence Detector 10101 moore machine 구현

(verilog source, simulation 결과, 상태도(State Table) 및 상태표(State Diagram) 작성)

sequence detector 10101 moore machine state diagram



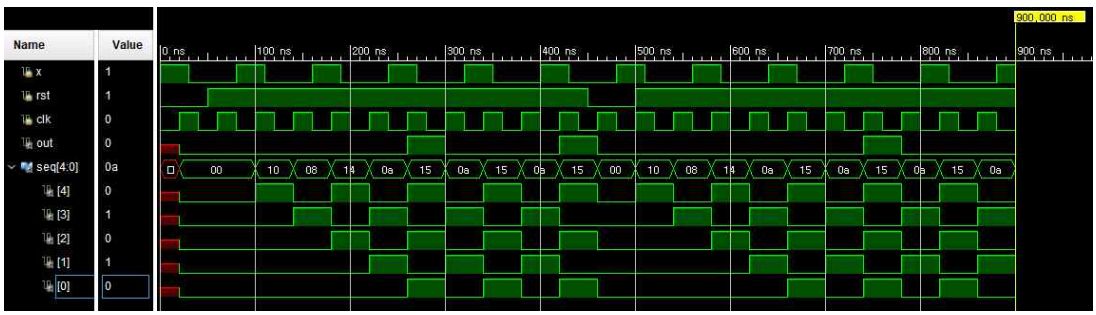
Q(t)	next state Q(t+1)	
	x = 0	x = 1
S_0	S_0	S_1
S_1	S_2	S_1
S_2	S_0	S_3
S_3	S_4	S_1
S_4	S_0	S_5
S_5	S_4	S_1

source code	testbench code

```
`timescale 1ns / 1ps

module moore_10101_detector (
    input x, clk, rst,
    output reg out,
    output reg [4:0] seq
);
    parameter S0 = 3'b000, S1 = 3'b001, S2 = 3'b010, S3 = 3'b011, S4 = 3'b100, S5 = 3'b101;
    reg [2:0] current_state, next_state;

    always@ (posedge clk) begin
        if (!rst) begin // active-low
            current_state <= S0;
            out <= 1'b0;
            seq <= 5'b00000;
        end
        else begin
            out = 1'b0;
            case (current_state)
                S0: next_state = x ? S0 : S1;
                S1: next_state = x ? S2 : S1;
                S2: next_state = x ? S0 : S3;
                S3: next_state = x ? S4 : S1;
                S4: next_state = x ? S0 : S5;
                S5: begin
                    next_state = x ? S2 : S0;
                    out = 1'b1;
                end
                default: next_state = S0;
            endcase
            // sequence 출력
            seq[4] <= x;
            seq[3] <= seq[4];
            seq[2] <= seq[3];
            seq[1] <= seq[2];
            seq[0] <= seq[1];
            // 상태 전환
            current_state <= next_state;
        end
    end

endmodule
```

```
`timescale 1ns / 1ps

module moore_10101_detector_tb;

reg x, rst, clk;
wire out;
wire [4:0] seq;

moore_10101_detector u_test(
    .x (x), .rst (rst), .clk (clk),
    .out (out), .seq (seq)
);

initial begin
    x = 1'b1; rst = 1'b0; clk = 1'b0;
    // 시작할 때 값을 reset
    forever #20 clk = ~clk;
end

always@ (x) begin
    #25 x <= ~x;
end

initial begin
    #50  rst = 1'b1;
    #400 rst = 1'b0;
    #50  rst = 1'b1;
    #400
    $finish;
end

endmodule
```

sequence detector 10101 moore machine simultation



10101 sequence detector moore machine의 경우 입력 sequence에서 '10101'을 검출하면 High 신호를 출력하고, 그 외에는 Low를 출력한다. meal y model로 구현했기 때문에 입력값과 무관하게 현재 상태에 따라 값을 출력하며, 6가지 상태 S_0-S_5 를 갖는다.

Verilog 코드 상의 구현의 경우 위의 1101 sequence detector와 거의 동

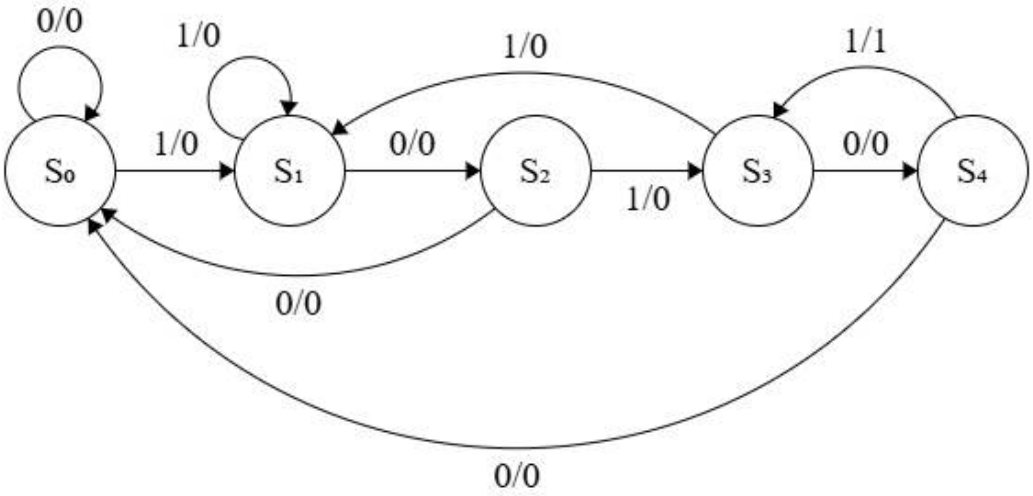
일하다. 테스트벤치 코드 역시 시뮬레이션 중 10101이 들어올 수 있도록 x의 값의 변화를 조정해준 것 외에 큰 변동 사항이 없다.

시뮬레이션 결과, rst 값에 따라 회로 상태의 초기화가 정상적으로 이루어졌고, '10101'이 입력되었을 때에만 out 변수가 set되는 것을 확인할 수 있었다. 회로가 10101 sequence detector의 논리적 동작과 동일하게 동작하므로 성공적으로 구현된 것을 확인할 수 있었다.

2-2. Overlapping 방식의 Sequence Detector 10101 mealy machine 구현

(verilog source, simulation 결과, 상태도(State Table) 및 상태표(State Diagram) 작성)

sequence detector 10101 mealy machine state diagram



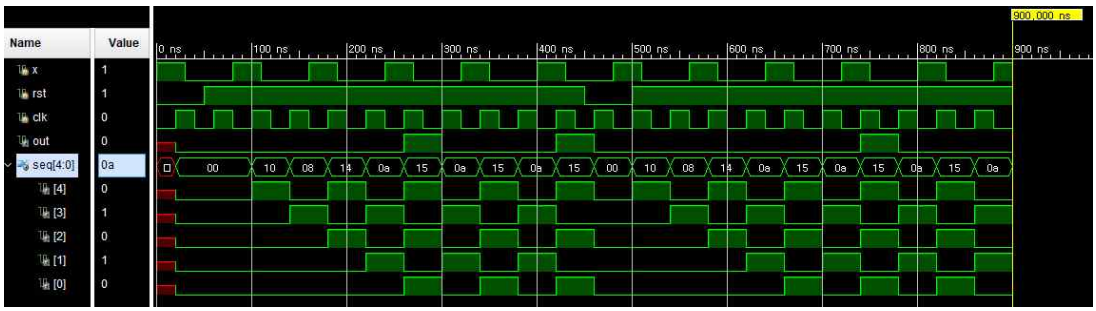
Q(t)	next state Q(t+1)		output	
	x = 0	x = 1	x = 0	x = 1
S_0	S_0	S_1	0	0
S_1	S_2	S_1	0	0
S_2	S_0	S_3	0	0
S_3	S_4	S_1	0	0
S_4	S_0	S_3	0	1

source code	testbench code

```
`timescale 1ns / 1ps

module mealy_10101_detector (
    input x, clk, rst,
    output reg out,
    output reg [4:0] seq
);
    parameter S0 = 3'b000, S1 = 3'b001, S2 = 3'b010, S3 = 3'b011, S4 = 3'b100, S5 = 3'b101;
    reg [2:0] current_state, next_state;

    always@ (posedge clk) begin
        if (!rst) begin // active-low
            current_state <= S0;
            out <= 1'b0;
            seq <= 5'b00000;
        end
        else begin
            out = 1'b0;
            case (current_state)
                S0: next_state = x ? S0 : S1;
                S1: next_state = x ? S2 : S1;
                S2: next_state = x ? S0 : S3;
                S3: next_state = x ? S4 : S1;
                S4: begin
                    next_state = x ? S0 : S3;
                    out = x;
                end
                default: next_state = S0;
            endcase
            // sequence 출력
            seq[4] <= x;
            seq[3] <= seq[4];
            seq[2] <= seq[3];
            seq[1] <= seq[2];
            seq[0] <= seq[1];
            // 상태 전환
            current_state <= next_state;
        end
    end

endmodule
```

```
`timescale 1ns / 1ps

module mealy_10101_detector_tb;

reg x, rst, clk;
wire out;
wire [4:0] seq;

mealy_10101_detector u_test(
    .x (x), .rst (rst), .clk (clk),
    .out (out), .seq (seq)
);

initial begin
    x = 1'b1; rst = 1'b0; clk = 1'b0;
    // 시작할 때 값을 reset
    forever #20 clk = ~clk;
end

always@ (x) begin
    #30 x <= ~x;
    #50 x <= ~x;
end

initial begin
    #50  rst = 1'b1;
    #400 rst = 1'b0;
    #50  rst = 1'b1;
    #400
    $finish;
end

endmodule
```

sequence detector 10101 mealy machine simultation



10101 sequence detector mealy machine의 경우 입력 sequence에서 '1 0101'을 검출하면 High 신호를 출력하고, 그 외에는 Low를 출력한다. mealy model로 구현했기 때문에 입력값과 현재 상태에 따라 값을 출력하며, 5가지 상태 S_0-S_4 를 갖는다.

전체적인 Verilog 코드 상의 구현은 moore model의 구현과 유사하지만,

최종적으로 1이 출력되는 상태 S4에서의 구현이 조금 다르다. 입력값에 따라 출력이 변화하는 것을 나타내기 위해 out = x로 입력값을 할당했다.

시뮬레이션 결과, rst 값에 따라 회로 상태의 초기화가 정상적으로 이루어졌고, '10101'이 입력되었을 때에만 out 변수가 set되는 것을 확인할 수 있었다. 회로가 10101 sequence detector의 논리적 동작과 동일하게 동작하므로 성공적으로 구현된 것을 확인할 수 있었다.