SCHOOL OF ELECTRONICS ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)

VLSI LABORATORY REPORT

(EC-3095)

Submitted By

Name: Debagnik Kar

Roll No: 1804373

Section: ETC-06

Semester: 6th semester

Experiment: 5th

Mealy and moore machines

Experiment No - 05

Aim: (i) Modeling finite state machine (FSM) in Verilog.

(ii) Design of sequence detector (overlapping and non-overlapping sequence) using Mealy FSM.

(iii) Design of sequence detector (overlapping and non-overlapping sequence) using Moore FSM.

Softwar Used: EDA playground

Theory :-

Finite State Machine (FSM) are sequential circuit used in many digital systems to control the behavior of systems and data flow paths. Examples of FSM include control units and sequencers. This experiment introduces the concept of two types of FSMs, Mealy and Moore, and the modeling styles to develop such machines. Please refer to the The two types shown below.

Debagnik Kar 1804373

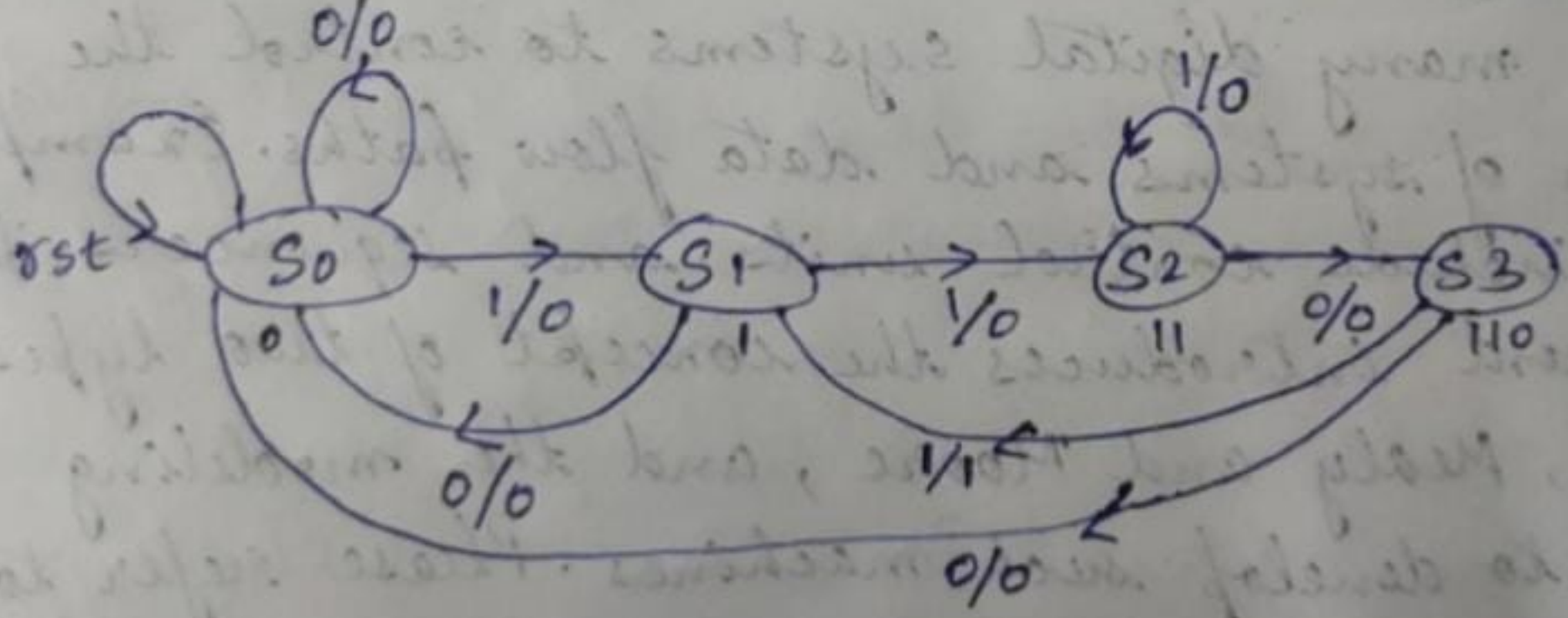
⟩ Mealy Machine (FSM)

A sequence detector is a sequential state machine which takes an input string of bits and generates an output 1 whenever the target sequence has been detected. In a Mealy machine, output depends on the present state and the external input (x). Hence in the state diagram, the output is written outside the states, along with inputs.

Sequence detector is of two types:

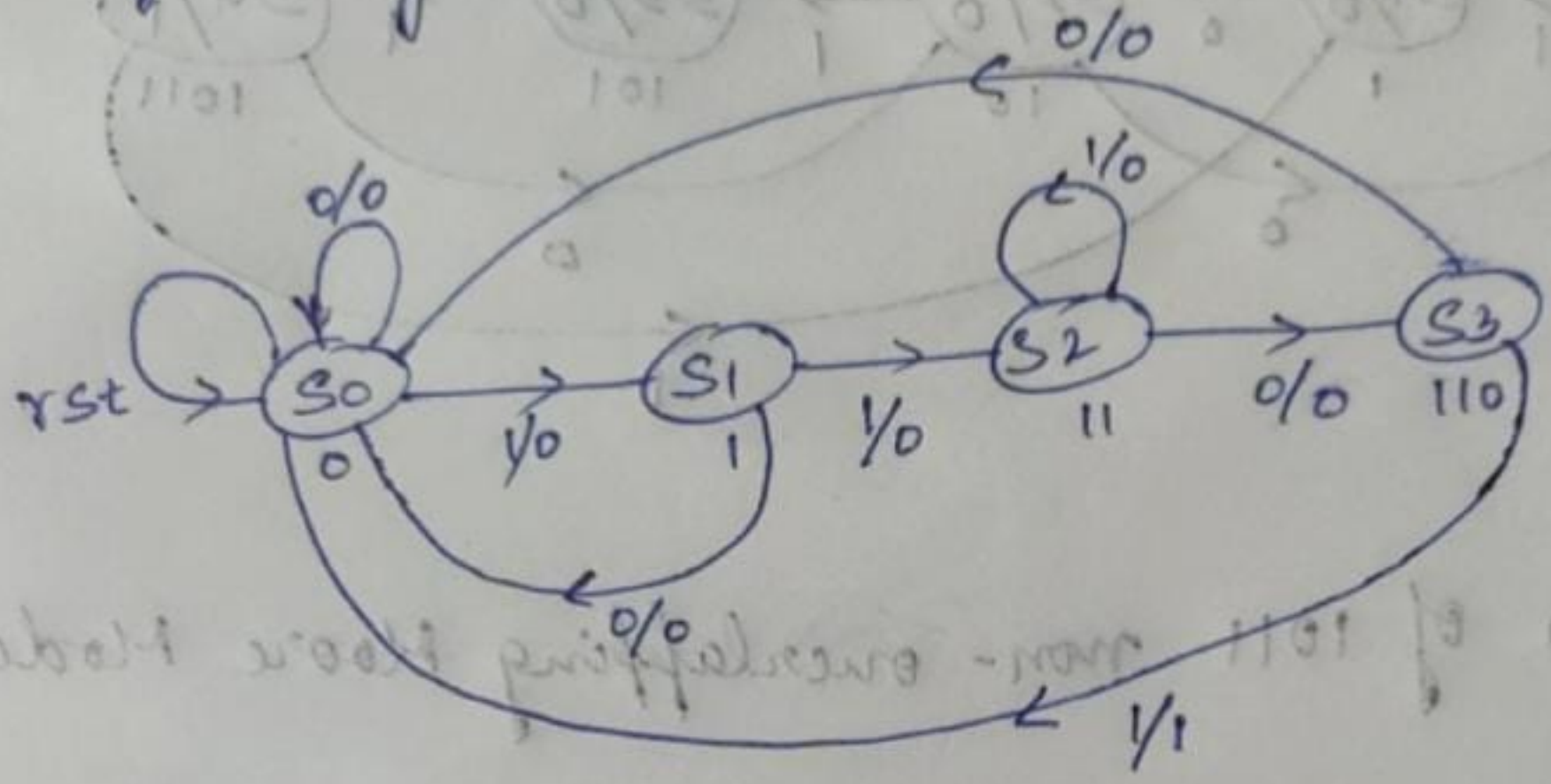
1) Overlapping - In an overlapping sequence detector the last bit of one sequence becomes the first bit of next sequence.

* Diagram of 1101 overlapping Mealy Model



2) Non-overlapping ⇒ In non-overlapping sequence detector the last bit of one sequence does not become the first bit of next sequence

Diagram of 1101 non-overlapping Mealy Model



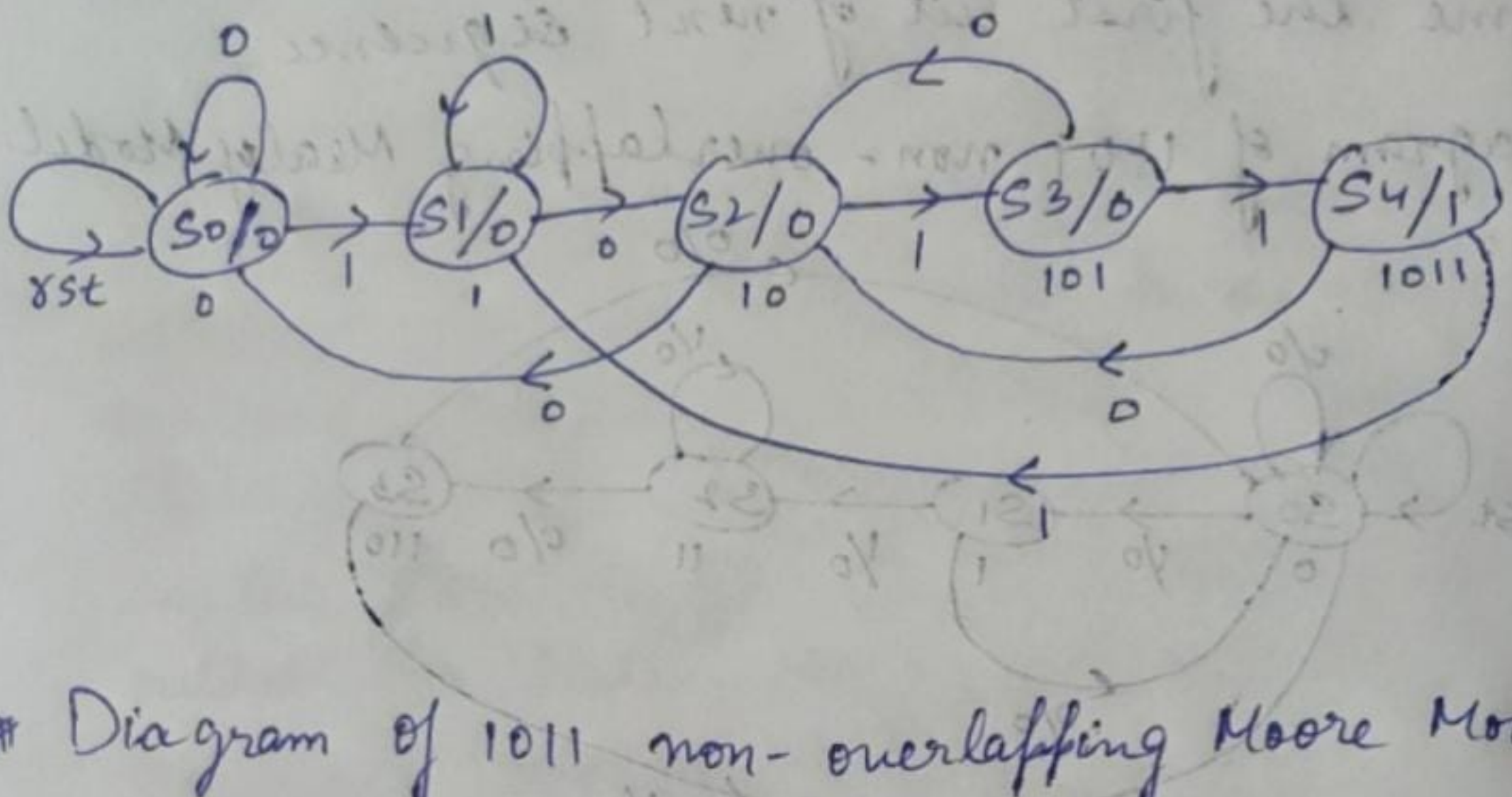
2) Moore Machine (FSM)

In moore machine, output depends only on the present state and not dependent on the input (x). Hence in the diagram output is written with the states. It has one more state than Mealy model and this also has two types of sequence detectors:-

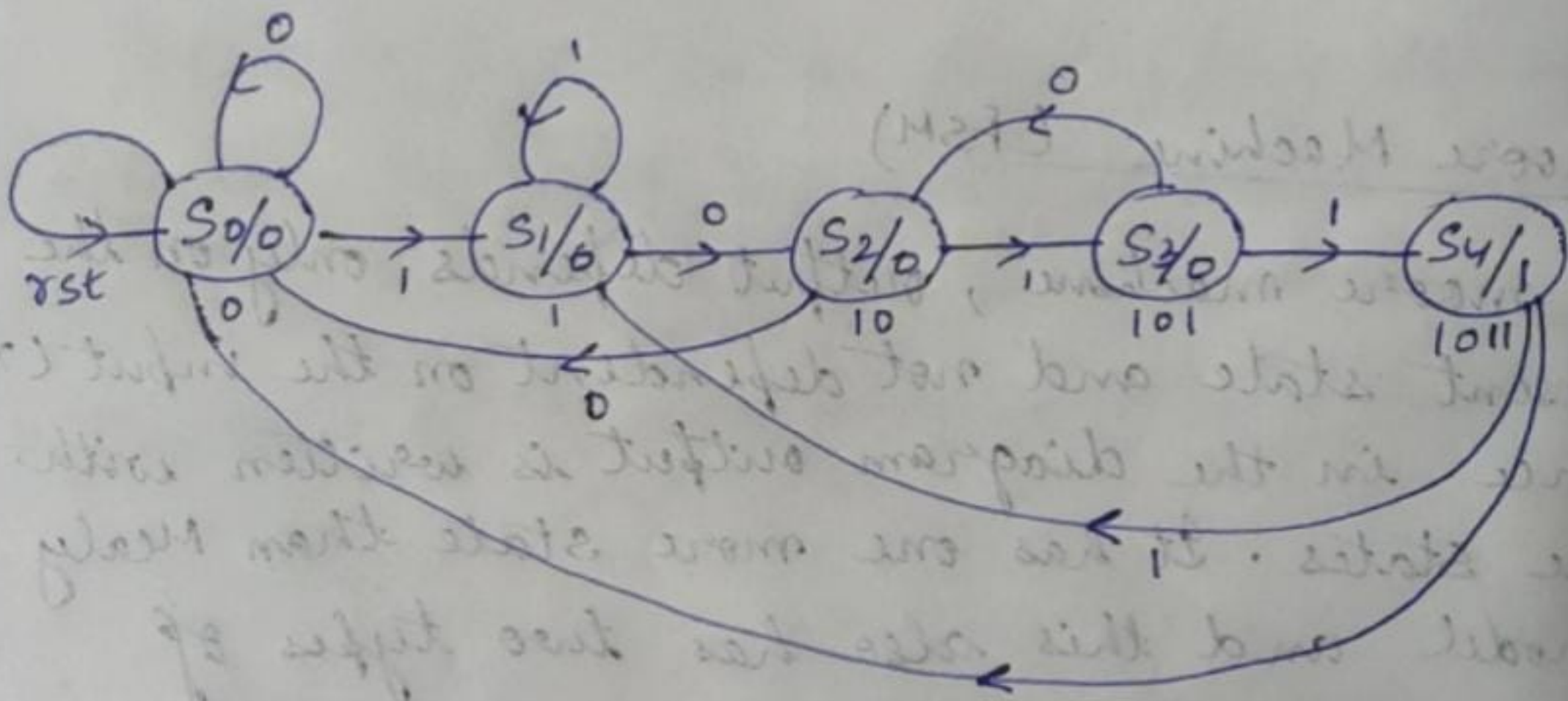
1) Overlapping

2) Non-overlapping

* Diagram of 1011 overlapping Moore Model



** Diagram of 1011 non-overlapping Moore Model



Code
Moore machine
Overlap method:
Design.sv
```
module moore_1011_overLAP(output dout,input din,clock,reset);
 reg dout;
 reg [2:0]state;
 parameter s0=3'b000,s1=3'b001,s2=3'b010,s3=3'b011,s4=3'b100;
 always@(posedge clock or posedge reset) begin
 if(reset) begin
 dout<=0;
 state<=s0;
 end
 else begin
 case(state)
 s0:begin dout<=0;
 if(din==1)
 state<=s1;
 else
 state<=s0;
 end
 s1:begin dout<=0;
 if(din==1)
 state<=s1;
 else
 state<=s2;
 end
 s2:begin dout<=0;
 if(din==1)
 state<=s3;
 else
 state<=s2;
 end
 s3:begin dout<=0;
 if(din==1)
 state<=s4;
 else
 state<=s2;
 end
 s4:begin dout<=1;
 if(din==1)
 state<=s1;
 else
 state<=s2;
 end
 default:begin dout<=0;state<=s0;end
 endcase
 end
```

```
 end
endmodule
```

Testbench.sv

```
module moore_1011_overLAP_test;
 reg din, reset, clock;
 wire dout;
 moore_1011_overLAP m1(dout,din,clock,reset);
 always #10 clock=!clock;
 initial begin
 $dumpfile("dump.vcd"); $dumpvars;
 clock=0;
 reset=1;
 din=1;
 #20 reset=0;din=0;
 #20 din=1;
 #20 din=0;
 #20 din=1;
 #20 din=1;
 #20 din=0;
 #20 din=1;
 #20 din=0;
 #20 din=1;
 #20 din=1;
 #20 din=0;
 #20 din=1;
 #20 din=1;
 #20 din=0;
 #20 din=1;
 #20 din=0;
 end
endmodule
```

Non-overlap Method

Design.sv

```
module moore_1011_non_overLAP(output dout,input din,clock,reset);
 reg dout;
 reg [2:0]state;
 parameter s0=3'b000,s1=3'b001,s2=3'b010,s3=3'b011,s4=3'b100;
 always@(posedge clock or posedge reset) begin
 if(reset) begin
 dout<=0;
 state<=s0;
 end
 else begin
 case(state)
```

```
 s0:begin dout<=0;
 if(din==1)
 state<=s1;
 else
 state<=s0;
 end
 s1:begin dout<=0;
 if(din==1)
 state<=s1;
 else
 state<=s2;
 end
 s2:begin dout<=0;
 if(din==1)
 state<=s3;
 else
 state<=s2;
 end
 s3:begin dout<=0;
 if(din==1)
 state<=s4;
 else
 state<=s2;
 end
 s4:begin dout<=1;
 if(din==1)
 state<=s1;
 else
 state<=s0;
 end
 default:begin dout<=0;state<=s0;end
 endcase
 end
 end
endmodule
```

testbench.sv

```
module moore_1011_non_overLAP_test;
 reg din, reset, clock;
 wire dout;
 moore_1011_non_overLAP m1(dout,din,clock,reset);
 always #10 clock=!clock;
 initial begin
 $dumpfile("dump.vcd"); $dumpvars;
 clock=0;
 reset=1;
 din=1;
 #20 reset=0;din=0;
 #20 din=1;
```

```
 #20 din=0;
 #20 din=1;
 #20 din=1;
 #20 din=0;
 #20 din=1;
 #20 din=0;
 #20 din=1;
 #20 din=1;
 #20 din=0;
 #20 din=1;
 #20 din=1;
 #20 din=0;
 #20 din=1;
 #20 din=0;
 end
endmodule
```

Mealy machine

Overlap method

Design.sv

```
module mealey_overlap_1101( output dout, input din,reset, clock);
reg dout;
 reg [1:0] state;
 parameter S0=2'b00, S1=2'b01, S2=2'b10,S3=2'b11;
 always @(posedge clock or posedge reset) begin
 if(reset) begin
 dout<= 0;
 state<= S0;
 end
else begin
 case(state)
 S0: begin
 if(din==1)
 state<=S1;
 else
 state<=S0;
 dout<=0;
 end
 S1: begin if(din)
 state<=S2;
 else
 state<=S0;
 dout<=0;
 end
 S2: begin if(din)
 state<=S2;
 else
 state<=S3;
 dout<= 0;
```

```
 end
 S3: begin if(din) begin
 state<=S1; dout<=1; end
 else begin
 state<=S0; dout<=0; end
 end
 endcase
end
 end
endmodule
```

Testbench.sv

```
module mealey_overlap_1101_test;
reg din,reset, clock;
 wire dout;
 mealey_overlap_1101 m1(dout, din,reset, clock);
 always #10 clock =!clock;
 initial begin
 $dumpfile("dump.vcd"); $dumpvars;
clock=0;
 reset=1;
 din=1;
 #20 reset = 0; din = 1;
 #20 din=0;
 #20 din=1;
 #20 din=1;
 #20 din=0;
 #20 din=1;
 #20 din=0;
 #20 din=1;
 #20 din=1;
 #20 din=0;
 #20 din=1;
 #20 din=1;
 #20 din=0;
 #20 din=1;
 #20 din=0;
 end
endmodule
```

Non-overlap Method

Design.sv

```
module mealey_non_overlap_1101( output dout, input din,reset, clock);
reg dout;
 reg [1:0] state;
 parameter S0=2'b00, S1=2'b01, S2=2'b10,S3=2'b11;
 always @(posedge clock or posedge reset) begin
 if(reset) begin
 dout<= 0;
```

```
 state<= S0;
 end
else begin
 case(state)
 S0: begin
 if(din==1)
 state<=S1;
 else
 state<=S0;
 dout<=0;
 end
 S1: begin if(din)
 state<=S2;
 else
 state<=S0;
 dout<=0;
 end
 S2: begin if(din)
 state<=S2;
 else
 state<=S3;
 dout<= 0;
 end
 S3: begin if(din) begin
 state<=S0; dout<=1; end
 else begin
 state<=S0; dout<=0; end
 end
 endcase
end
 end
endmodule
```

Testbench.sv

```
module mealey_non_overlap_1101_test;
reg din,reset, clock;
 wire dout;
 mealey_non_overlap_1101 m1(dout, din,reset, clock);
 always #10 clock =!clock;
 initial begin
 $dumpfile("dump.vcd"); $dumpvars;
clock=0;
 reset=1;
 din=1;
 #20 reset = 0; din = 1;
 #20 din=0;
 #20 din=1;
 #20 din=1;
```

```
 #20 din=0;
 #20 din=1;
 #20 din=0;
 #20 din=1;
 #20 din=1;
 #20 din=0;
 #20 din=1;
 #20 din=1;
 #20 din=0;
 #20 din=1;
 #20 din=0;
 end
endmodule
```

Results

Fig 5.1 Moore machine in overlap method

Fig 5.2: Moore machine in non-overlap method

Fig 5.3: Mealy machine overlap method

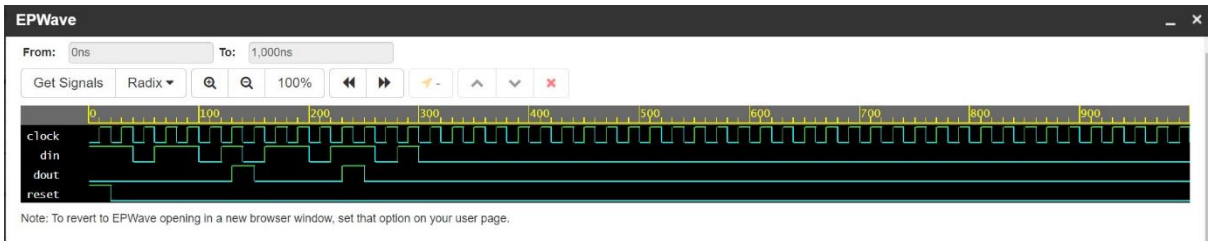


Fig 5.4: Mealy machine non-overlap method

Conclusion : In this experiment we learned about Finite state machines and we designed sequence detectors using mealy as well as Moore models for overlapping & non-overlapping sequences by writing codes in EDA playground and simulated them by writing their respective test bench codes to get outputs.