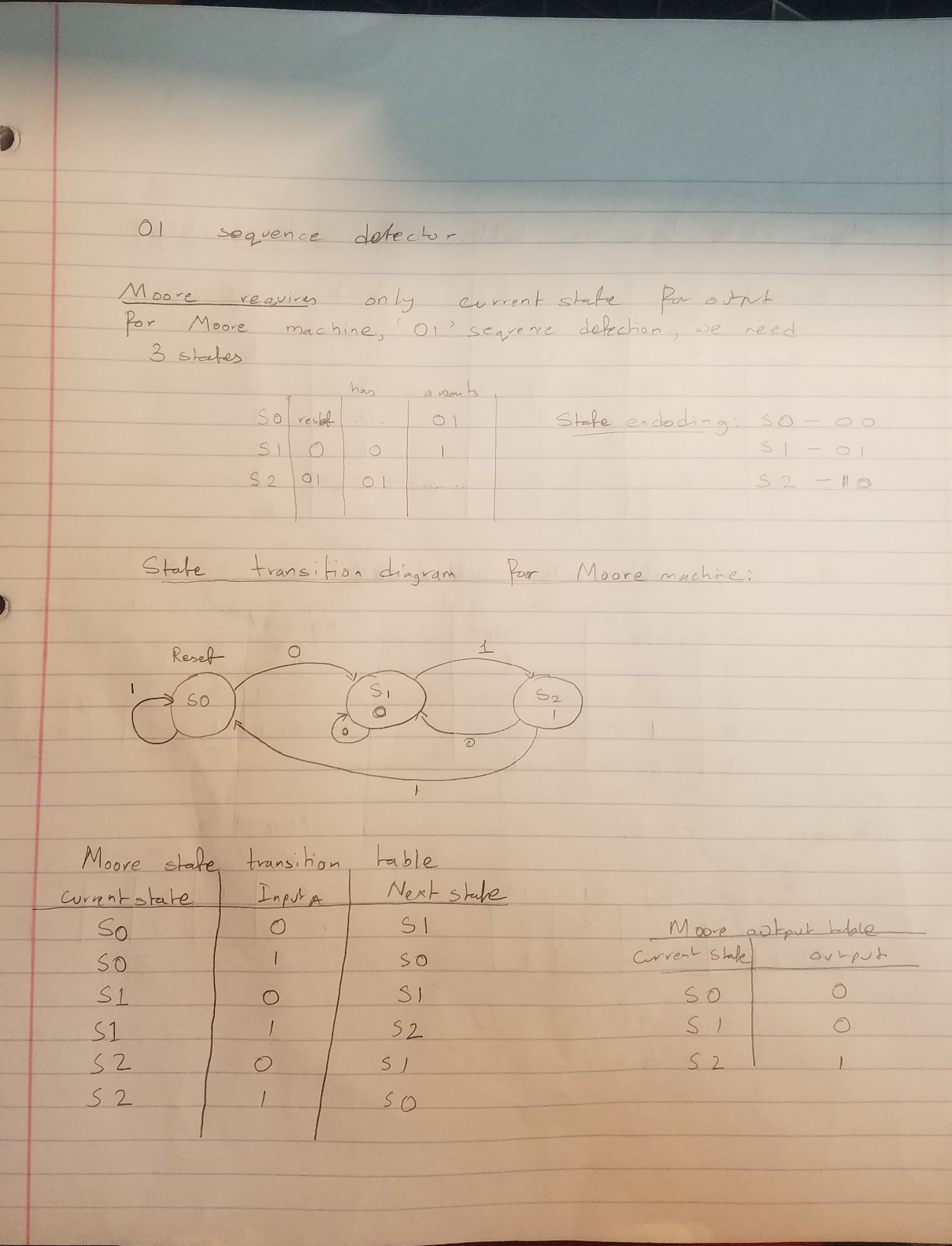
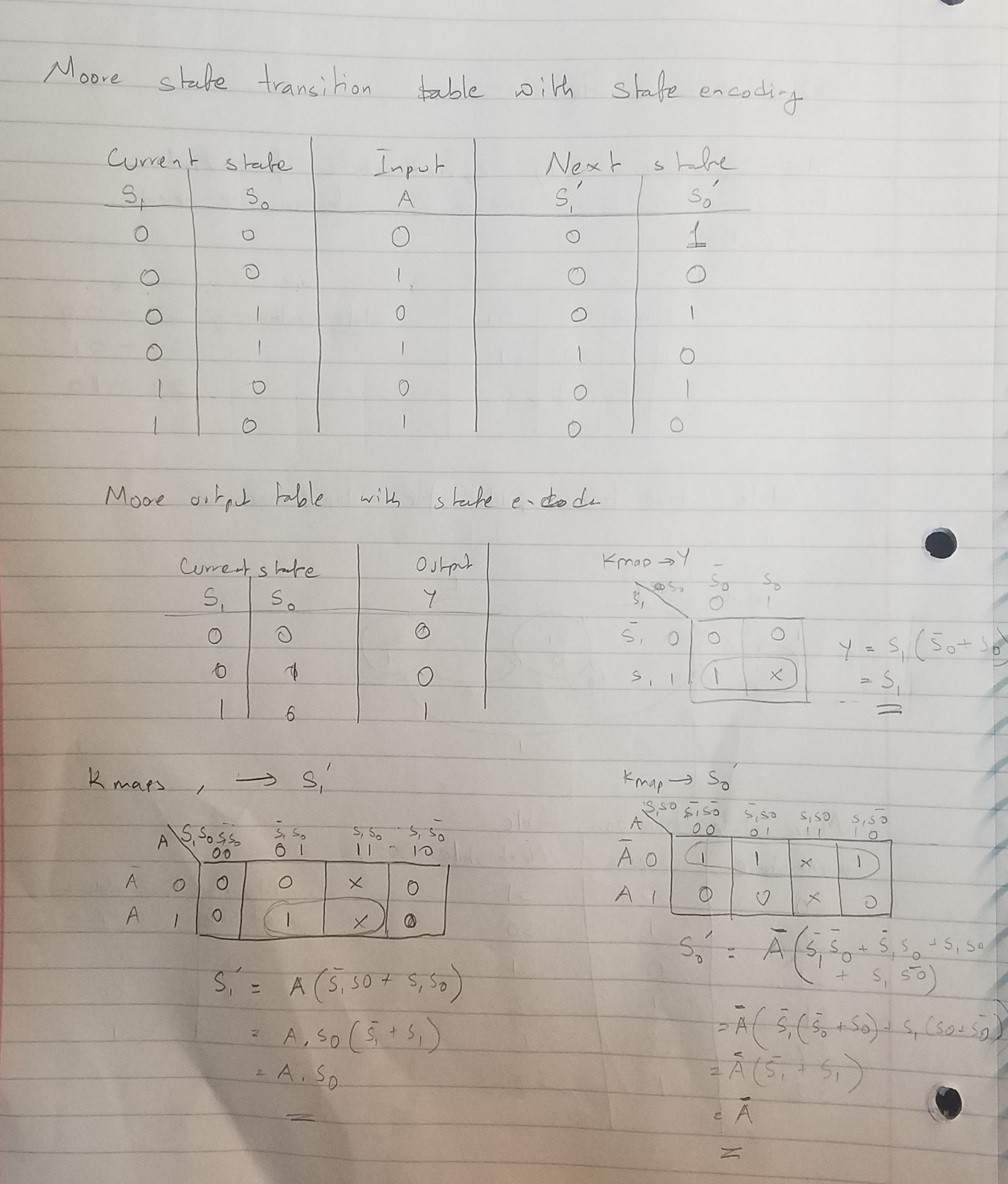
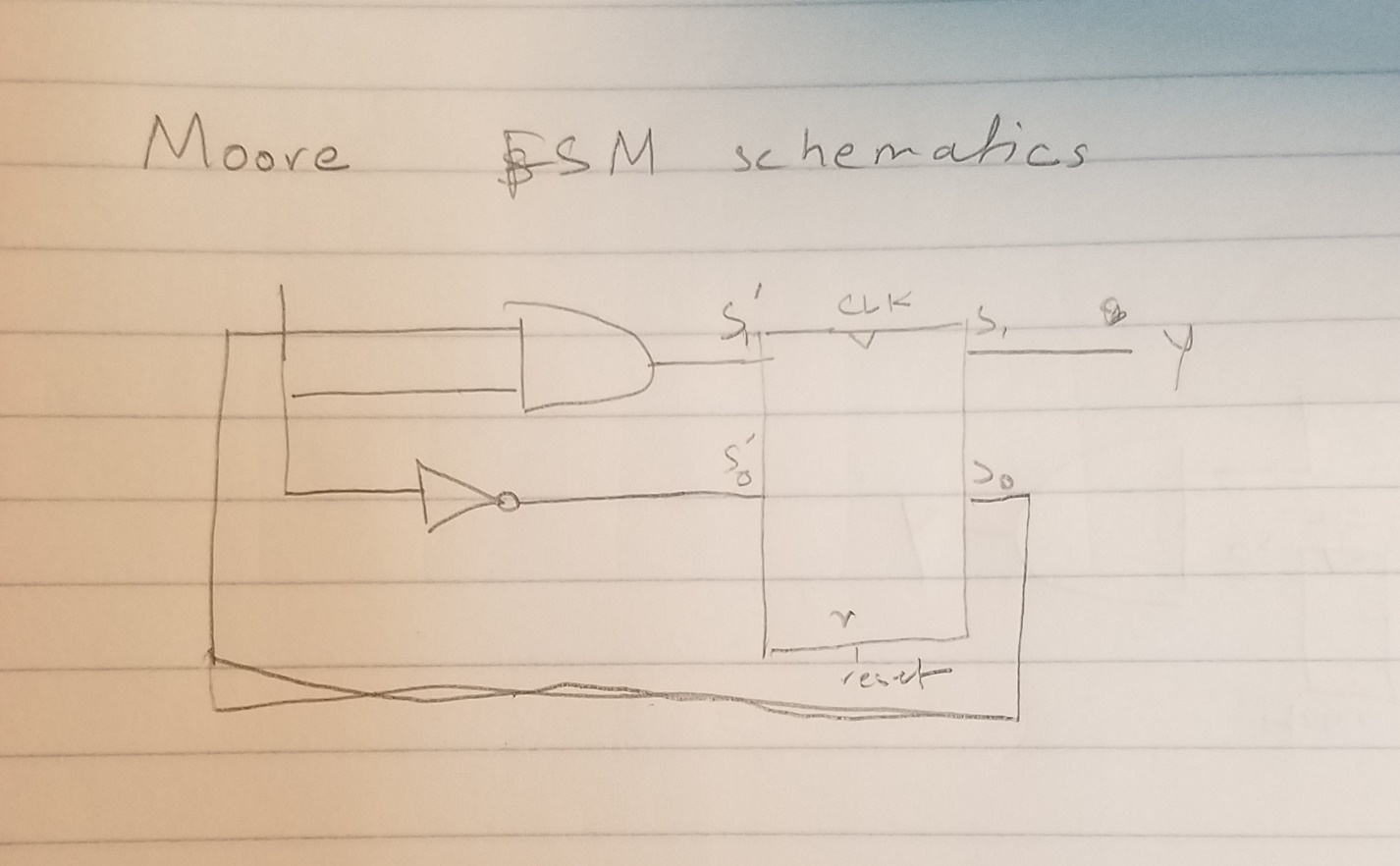
HW 3 report 2

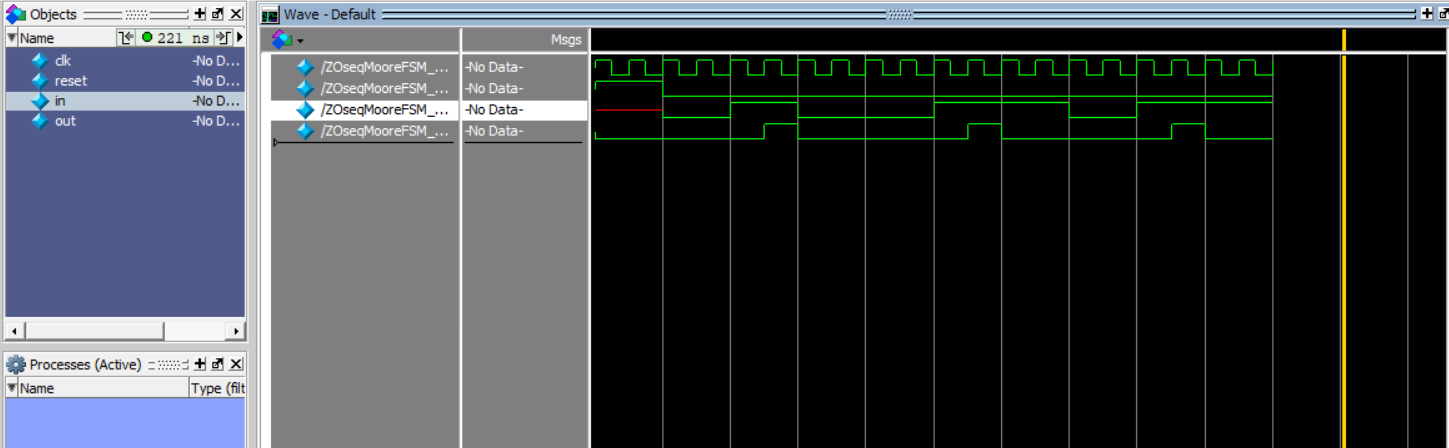
1) the snail that detects bit pattern '01'







‘01’ Sequence detector waveform from Verilog. Moore FSM.



Moore FSM Verilog code for ‘01’ sequence detector.

module MooreFSM(input logic clk,

input logic reset,

input logic A,

output logic Y);

typedef enum logic [2:0] { S0,S1,S2 } State;

State currentState, nextState;

always\_ff @(posedge clk)

if(reset) currentState <= S0;

else currentState <= nextState;

always\_comb

case(currentState)

S0: if(A) nextState = S0;

else nextState = S1;

S1: if(A) nextState = S2;

else nextState = S1;

S2: if(A) nextState = S0;

else nextState = S1;

default: nextState = S0;

endcase

assign Y = {currentState == S2};

endmodule

Testbench:

`timescale 1ns/1ns

module ZOseqMooreFSM\_TB;

logic clk;

logic reset;

logic in;

logic out;

MooreFSM dut(

.clk(clk),

.reset(reset),

.A(in),

.Y(out)

);

initial

begin //please run for 200ns

reset <= 1; # 20;

reset <= 0; in <= 0; # 20;

in <= 1; #20;

in <= 0; #20;

in <= 0; #20;

in <= 1; #20;

in <= 1; #20;

in <= 0; #20;

in <= 1; #20;

end

always

begin

clk <= 1; #5;

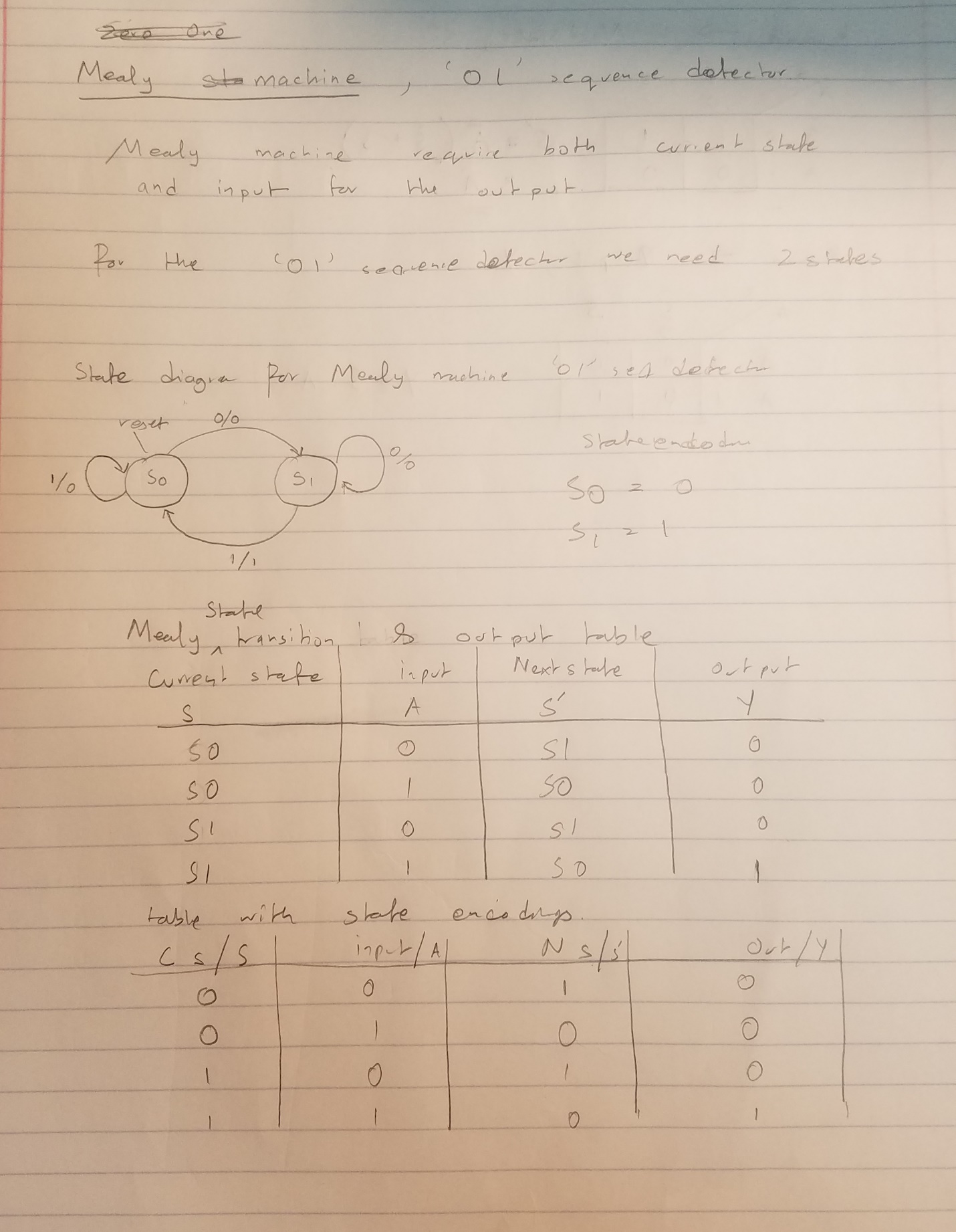
clk <= 0; #5;

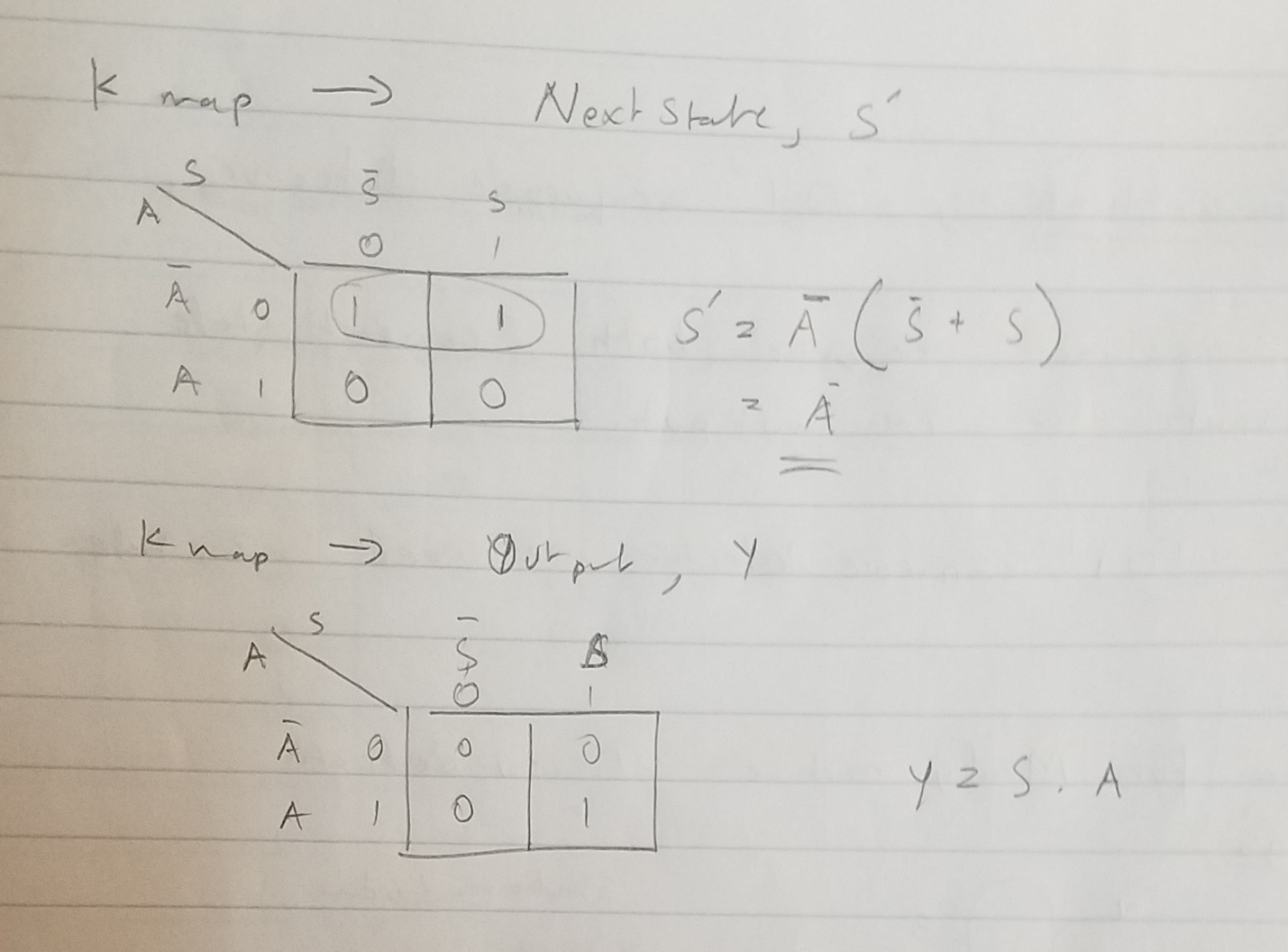
end

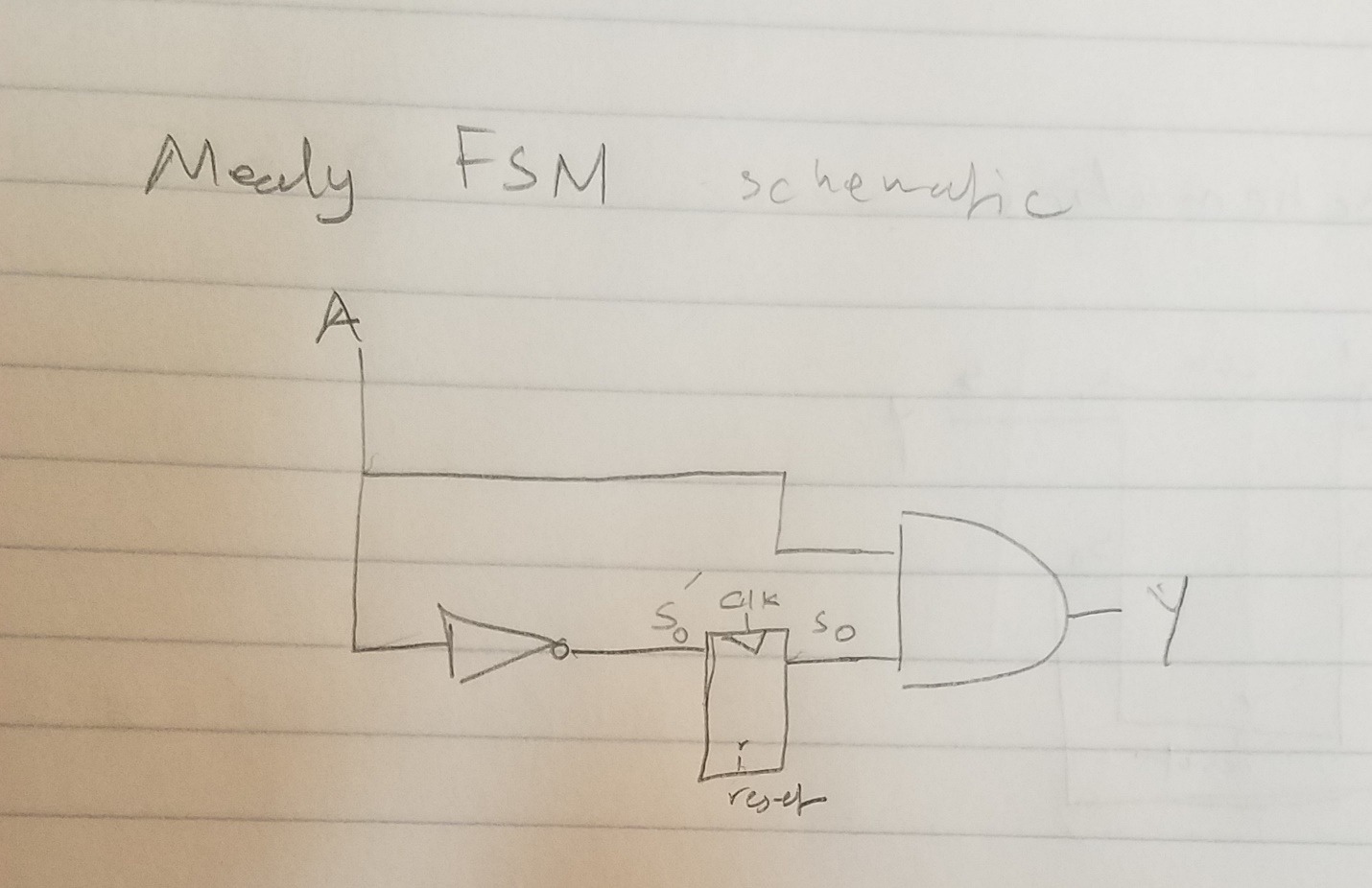
endmodule

(I will attach the system Verilog files with the report )

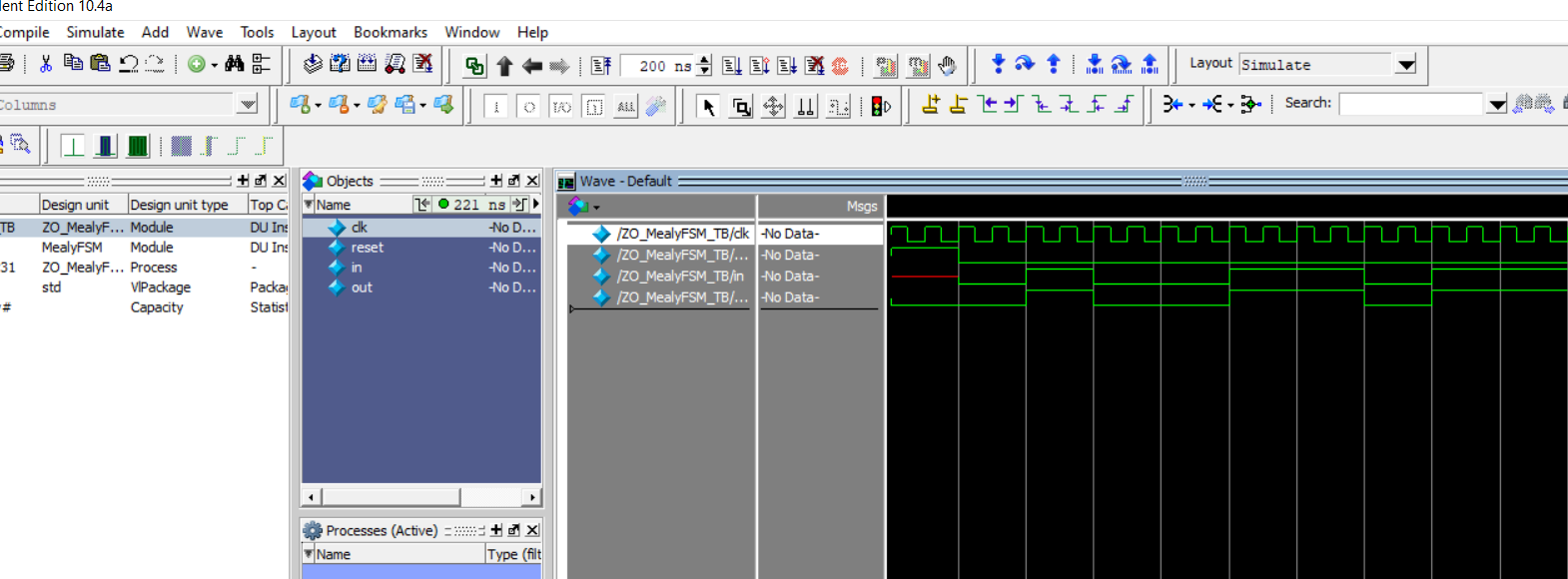
“01” sequence detector with Mealy machines







Verilog waveform for the mealy fsm , the sequence used is 0 1 0 0 1 1 0 1



Verilog code :

module MealyFSM(input logic clk,

input logic reset,

input logic A,

output logic Y);

typedef enum logic [1:0] { S0,S1 } State;

State currentState, nextState;

always\_ff @(posedge clk)

if(reset) currentState <= S0;

else currentState <= nextState;

always\_comb

case(currentState)

S0: if(A) nextState = S0;

else nextState = S1;

S1: if(A) nextState = S1;

else nextState = S1;

default: nextState = S0;

endcase

assign Y = (currentState == S1) & A;

endmodule

Testbench:

`timescale 1ns/1ns

module ZO\_MealyFSM\_TB;

logic clk;

logic reset;

logic in;

logic out;

MealyFSM dut(

.clk(clk),

.reset(reset),

.A(in),

.Y(out)

);

initial

begin //please run for 200ns

reset <= 1; # 20;

reset <= 0; in <= 0; # 20;

in <= 1; #20;

in <= 0; #20;

in <= 0; #20;

in <= 1; #20;

in <= 1; #20;

in <= 0; #20;

in <= 1; #20;

end

always

begin

clk <= 1; #5;

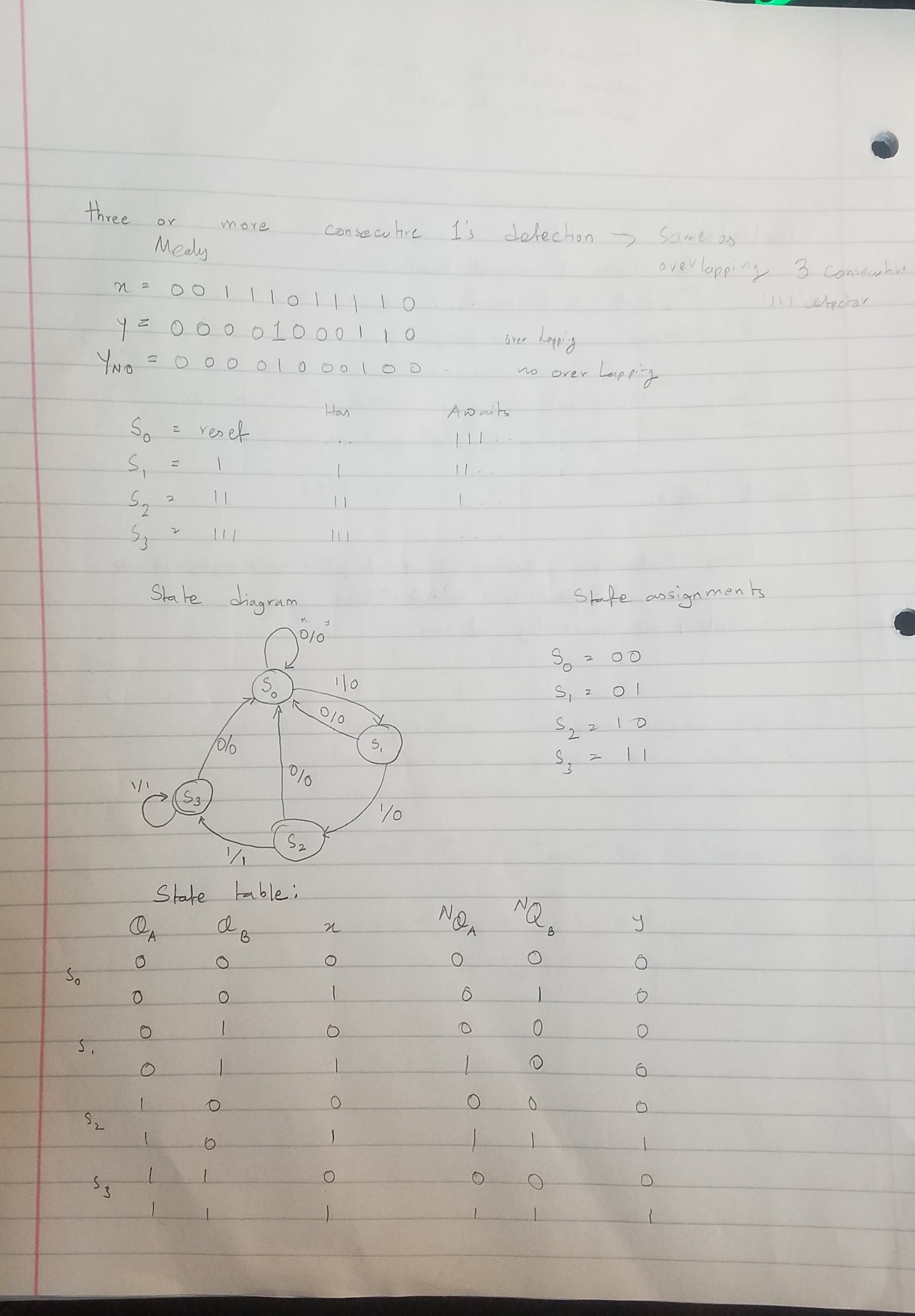
clk <= 0; #5;

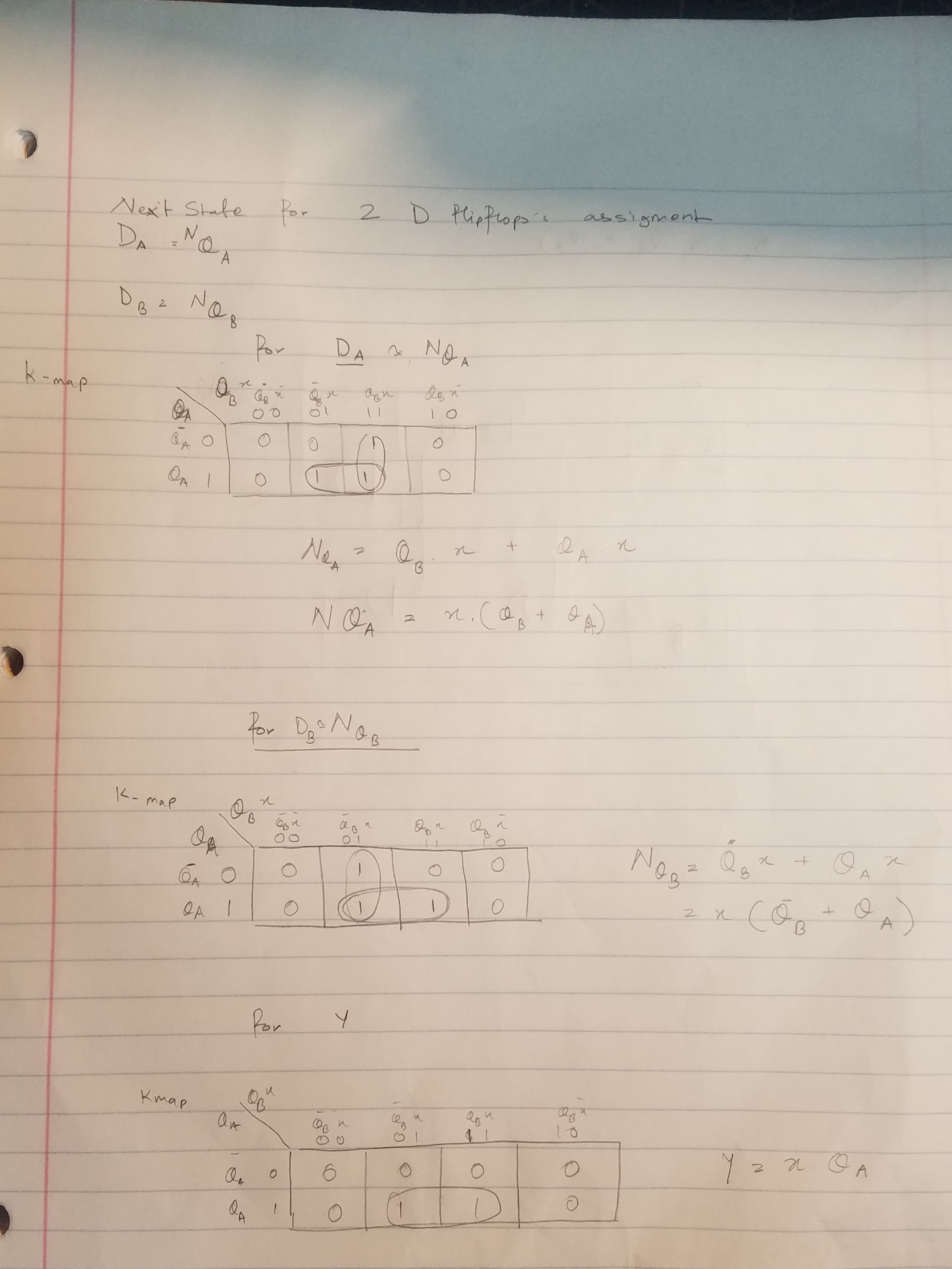
end

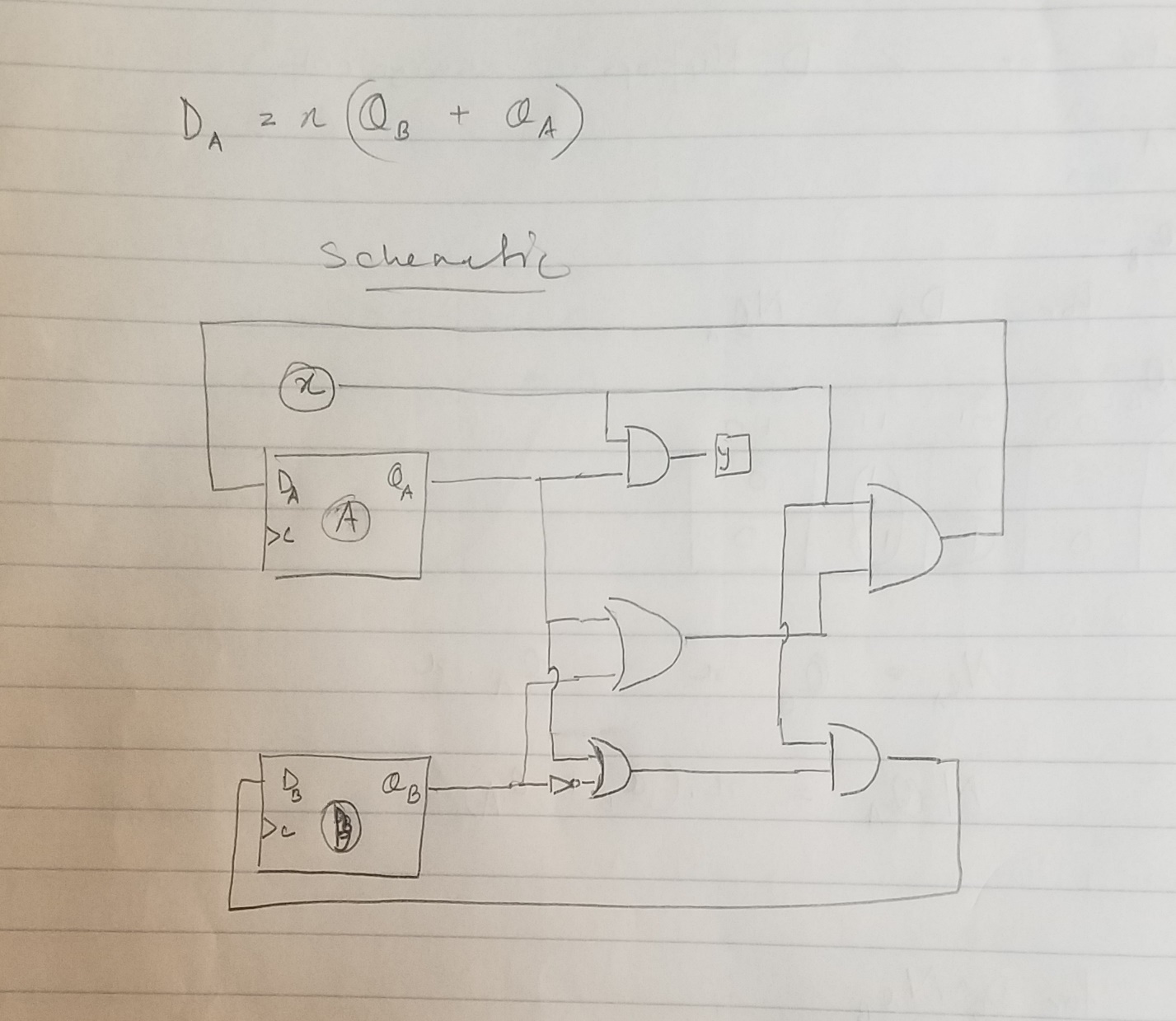
endmodule

(I will attach the System Verilog files along with he report)

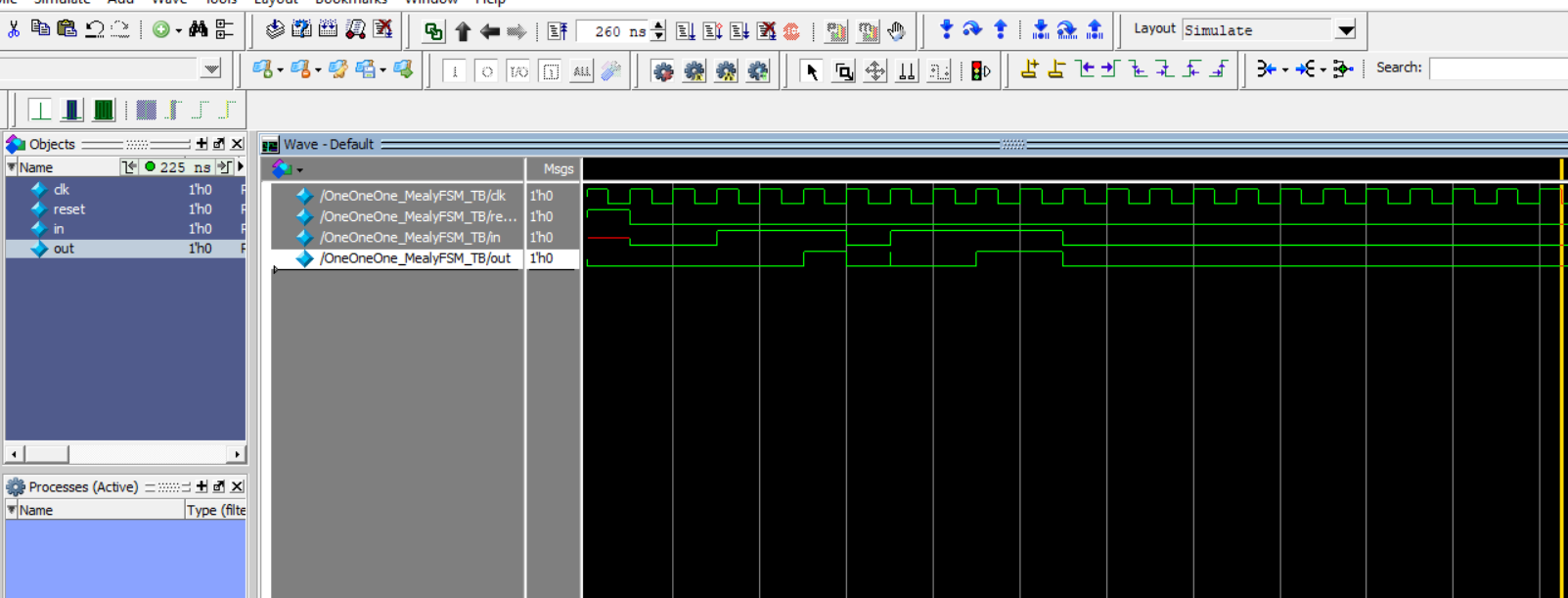
“111” sequence detector : Mealy and Moore FSM







Verilog waveform: Example sequence used is “00111011” as input



System Verilog code:

module MealyFSM(input logic clk,

input logic reset,

input logic A,

output logic Y);

typedef enum logic [2:0] { S0,S1,S2 } State;

State currentState, nextState;

always\_ff @(posedge clk)

if(reset) currentState <= S0;

else currentState <= nextState;

always\_comb

case(currentState)

S0: if(A) nextState = S1;

else nextState = S0;

S1: if(A) nextState = S2;

else nextState = S0;

S2: if(A) nextState = S2;

else nextState = S0;

default: nextState = S0;

endcase

assign Y = (currentState == S2) & A;

endmodule

Testbench:

`timescale 1ns/1ns

module OneOneOne\_MealyFSM\_TB;

logic clk;

logic reset;

logic in;

logic out;

MealyFSM dut(

.clk(clk),

.reset(reset),

.A(in),

.Y(out)

);

initial

begin //please run for 240ns

reset <= 1; # 10;

reset <= 0; in <= 0; # 10;

in <= 0; #10;

in <= 1; #10;

in <= 1; #10;

in <= 1; #10;

in <= 0; #10;

in <= 1; #10;

in <= 1; #10;

in <= 1; #10;

in <= 1; #10;

in <= 0; #10;

end

always

begin

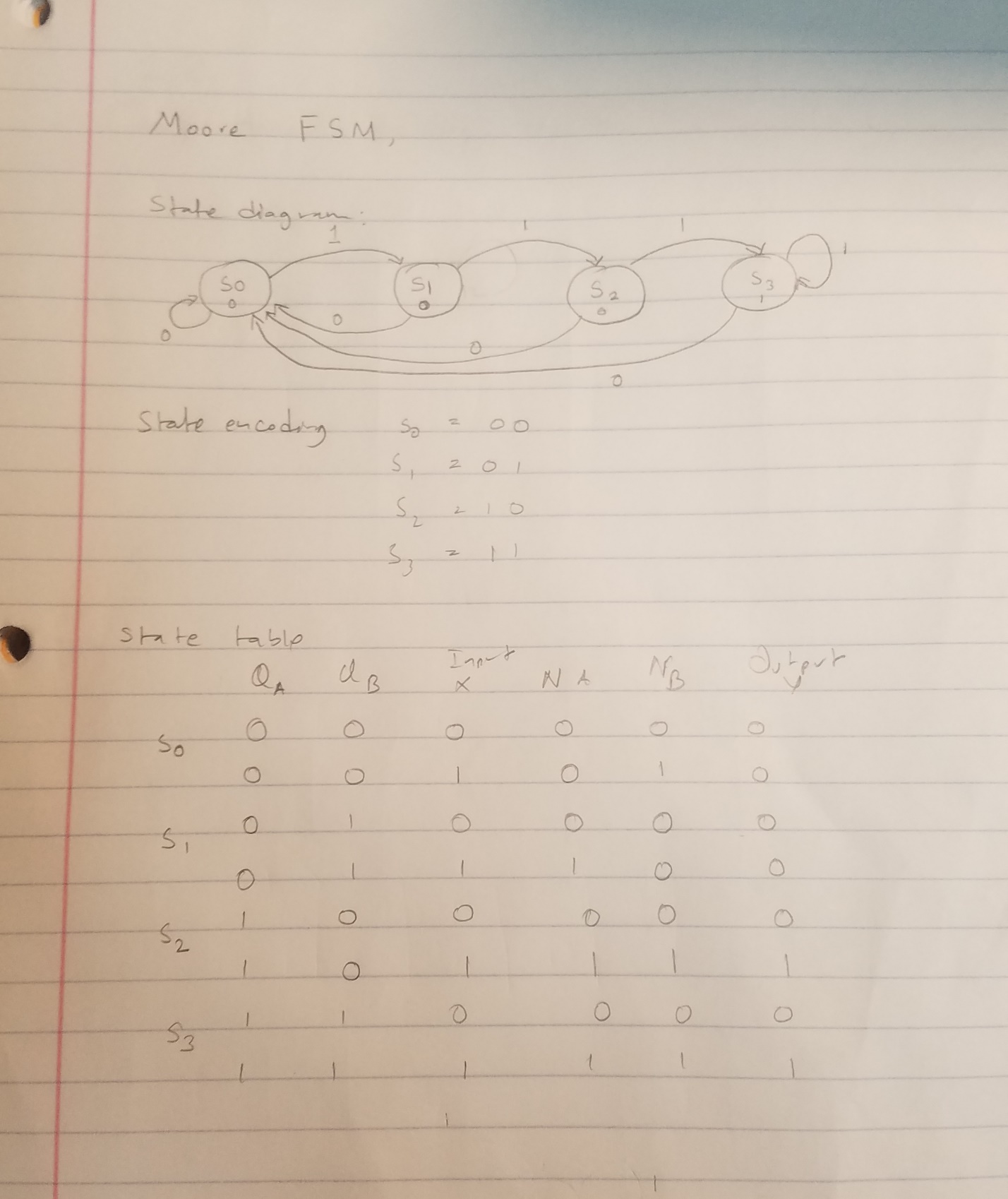
clk <= 1; #5;

clk <= 0; #5;

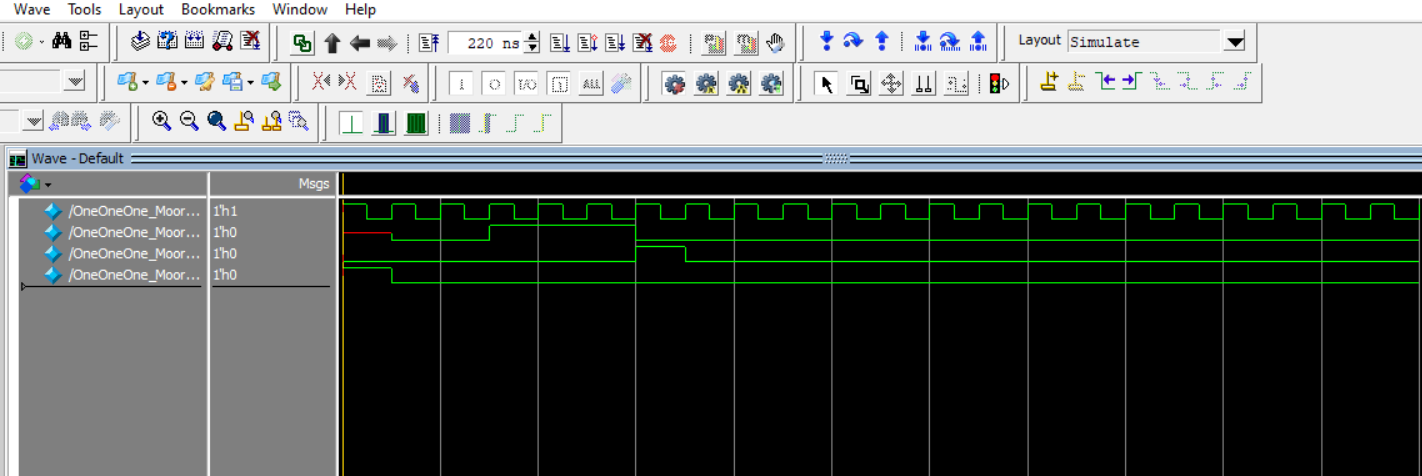
end

endmodule

Moore fsm



Verilog waveform: Sequence used (“01110” as input sequence)



We can see a delay ,showing the delay as Moore’s output changes when state changes at clock and doesn’t change immediately with input change, unlike Mealy machines.

System Verilog code:

module MooreFSM(input logic clk,

input logic reset,

input logic A,

output logic Y);

typedef enum logic [3:0] { S0,S1,S2,S3 } State;

State currentState, nextState;

always\_ff @(posedge clk)

if(reset) currentState <= S0;

else currentState <= nextState;

always\_comb

case(currentState)

S0: if(A) nextState = S1;

else nextState = S0;

S1: if(A) nextState = S2;

else nextState = S0;

S2: if(A) nextState = S3;

else nextState = S0;

S3: if(A) nextState = S3;

else nextState = S0;

default: nextState = S0;

endcase

assign Y = {currentState == S3};

endmodule

TestBench:

`timescale 1ns/1ns

module OneOneOne\_MooreFSM\_TB;

logic clk;

logic reset;

logic in;

logic out;

MooreFSM dut(

.clk(clk),

.reset(reset),

.A(in),

.Y(out)

);

initial

begin //please run for 200ns

reset <= 1; #10;

reset <= 0; in <= 0; #10;

in <= 0; #10;

in <= 1; #10;

in <= 1; #10;

in <= 1; #10;

in <= 0; #10;

end

always

begin

clk = 1; #5;

clk = 0; #5;

end

endmodule

(I will attach the System Verilog files with test bench with the report)