11/28/2016

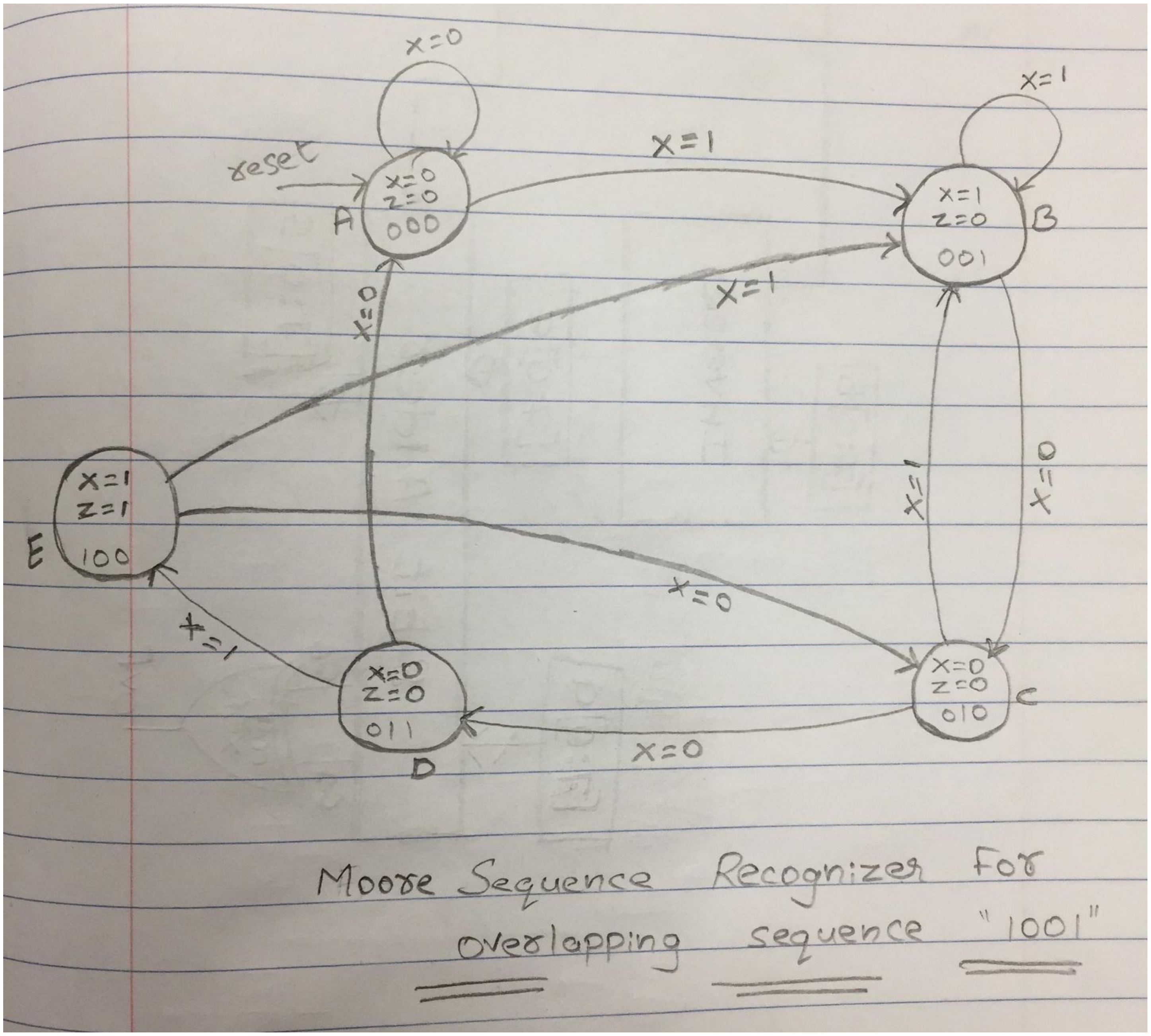
Project 3

Sequence Recognizer 1001

Bibodi, Jay Nikhil
Sec 2: 5:30-6:45
Fall 2016

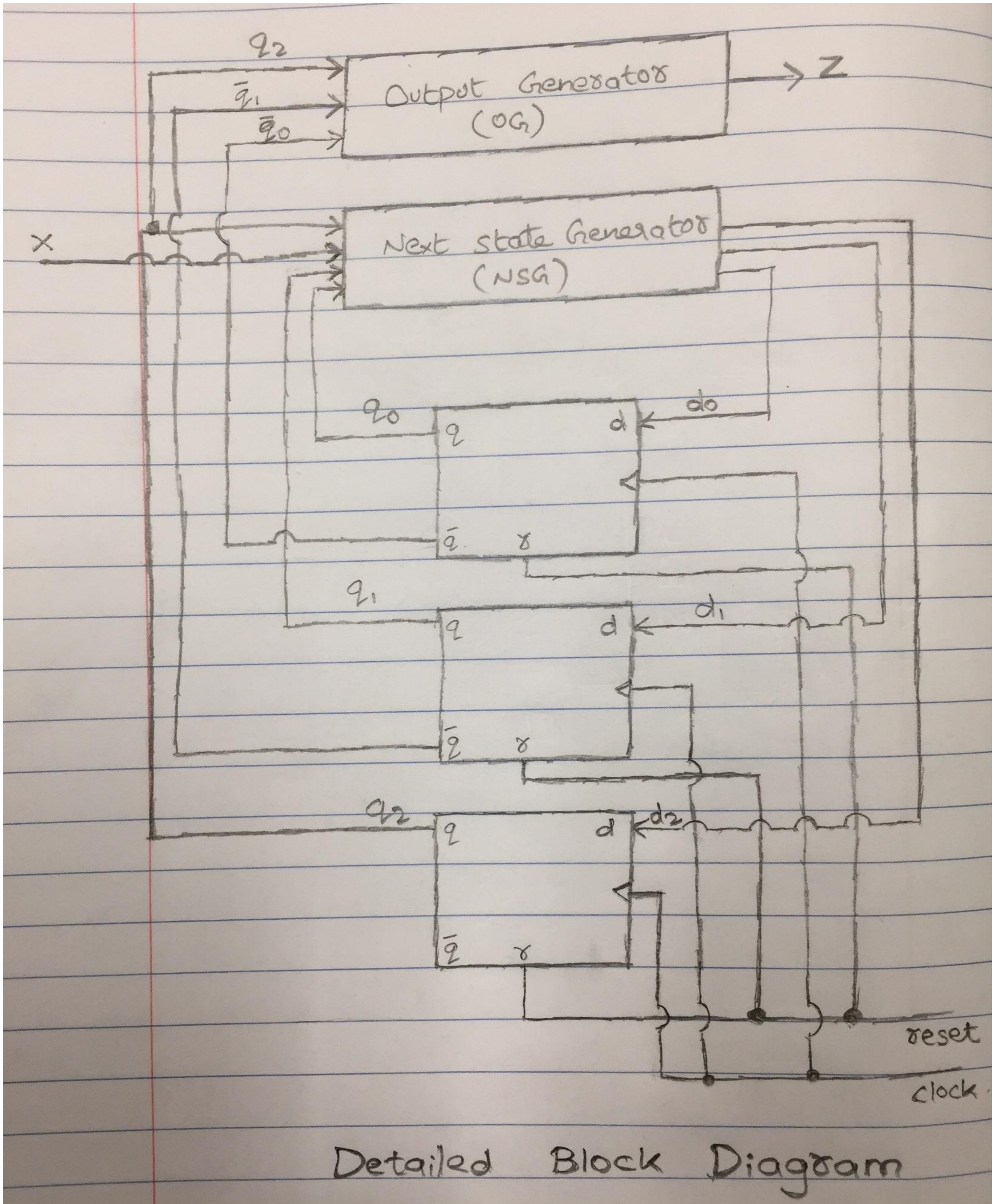
1. Consider problem 5.10

a. **Moore's Sequence Recognizer for overlapping sequence 1001**



Moore Sequence Recognizer for overlapping sequence "1001"

b. Detailed block diagram



Detailed Block Diagram

c. Truth table for OG

State Name	Current State			Output
	q2	q1	q0	Z
A	0	0	0	0
B	0	0	1	0
C	0	1	0	0
D	0	1	1	0
E	1	0	0	1
	d	d	d	d

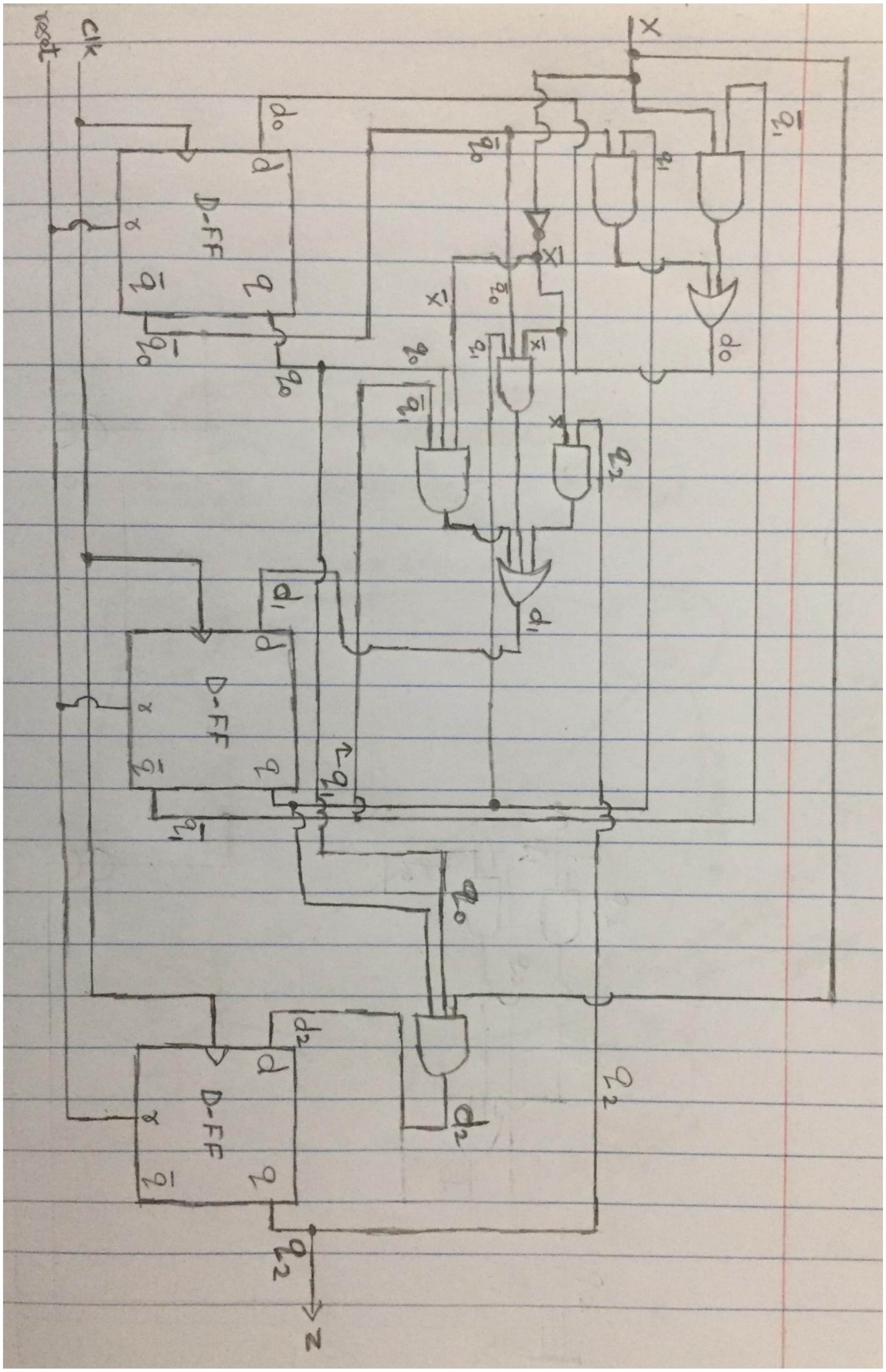
d. Truth table for NSG

State Name	Current State			Input	Next State			Next State Name
	q2	q1	q0	x	d2	d1	d0	
A	0	0	0	0	0	0	0	A
A	0	0	0	1	0	0	1	B
B	0	0	1	0	0	1	0	C
B	0	0	1	1	0	0	1	B
C	0	1	0	0	0	1	1	D
C	0	1	0	1	0	0	1	B
D	0	1	1	0	0	0	0	A
D	0	1	1	1	1	0	0	E
E	1	0	0	0	0	1	0	C
E	1	0	0	1	0	0	1	B
-	d	d	d	d	d	d	d	-

d – don’t Cares in the truth table

e. **Logic Expression for output and next state variables.**

f. Final Circuit for FSM



g. Behavioral Model for OG. (OG.v)

```
module OG
(
	input q2,
	output  z
);

 assign z = q2;
endmodule
```

h. Behavioral Model for NSG. (NSG.v)

```
module NSG
(
	input [2:0] q,
	input x,
	output [2:0] d
);

assign d[2] = q[1] & q[0] & x;
assign d[1] = (q[2] & ~x) | (q[1] & ~q[0] & ~x) | (~q[1] & q[0] & ~x);
assign d[0] = (~q[1] & x) | (q[1] & ~q[0]);

endmodule
```

i. Behavioral Model for FF. (FlipFlop.v)

```
module FlipFlop
(
	input clock,reset,
	input [2:0] nextState,
	output reg [2:0] currentState
);

always@(posedge clock,posedge reset)
begin
	if(reset == 1)
		currentState <= 3'b000;
	else
		currentState <= nextState;
end

endmodule
```

j. Combined Models to create FSM (Project.v)

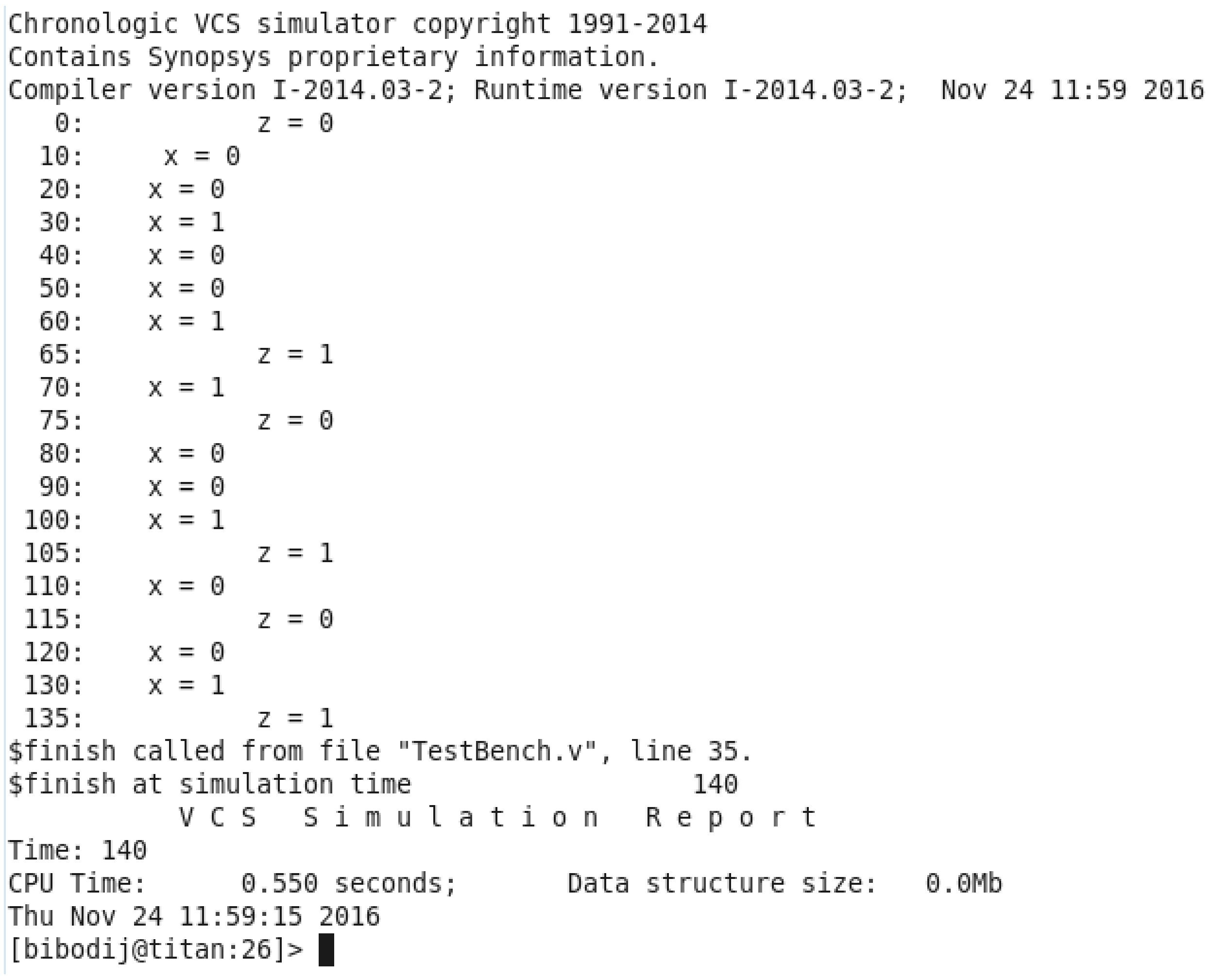
```
`include "OG.v";
`include "FlipFlop.v";
`include "NSG.v";

module Project
(
        input clock,reset,x,
        output out
);

wire [2:0] currentState, nextState;

NSG a(currentState,x,nextState);
FlipFlop b(clock,reset,nextState,currentState);
OG h(currentState[2],out);

endmodule
```

k. TestBench (TestBench.v)

```
`include "Project.v";

module TestBench();
reg clock,reset,x;
wire z;
Project ProjectOne(clock,reset,x,z);

initial begin
$monitor("%4d:                z = %b",$time,z);
clock=0;
reset=1;    //Resets the FlipFlop
x=0;
#10 reset=0;    //end reset
end

always
begin
#5clock=~clock; //generates a clock signal with period 10
end

initial begin  // One input per clock cycle
#10 x = 0; $display ("%4d:       x = %b",$time,x);
#10 x = 0; $display ("%4d:    x = %b",$time,x);
#10 x = 1; $display ("%4d:    x = %b",$time,x);
#10 x = 0; $display ("%4d:    x = %b",$time,x);
```

```
#10 x = 0; $display ("%4d:    x = %b",$time,x);
#10 x = 1; $display ("%4d:    x = %b",$time,x);
#10 x = 1; $display ("%4d:    x = %b",$time,x);
#10 x = 0; $display ("%4d:    x = %b",$time,x);
#10 x = 0; $display ("%4d:    x = %b",$time,x);
#10 x = 1; $display ("%4d:    x = %b",$time,x);
#10 x = 0; $display ("%4d:    x = %b",$time,x);
#10 x = 0; $display ("%4d:    x = %b",$time,x);
#10 x = 1; $display ("%4d:    x = %b",$time,x);
#10 $finish;
end
endmodule
```

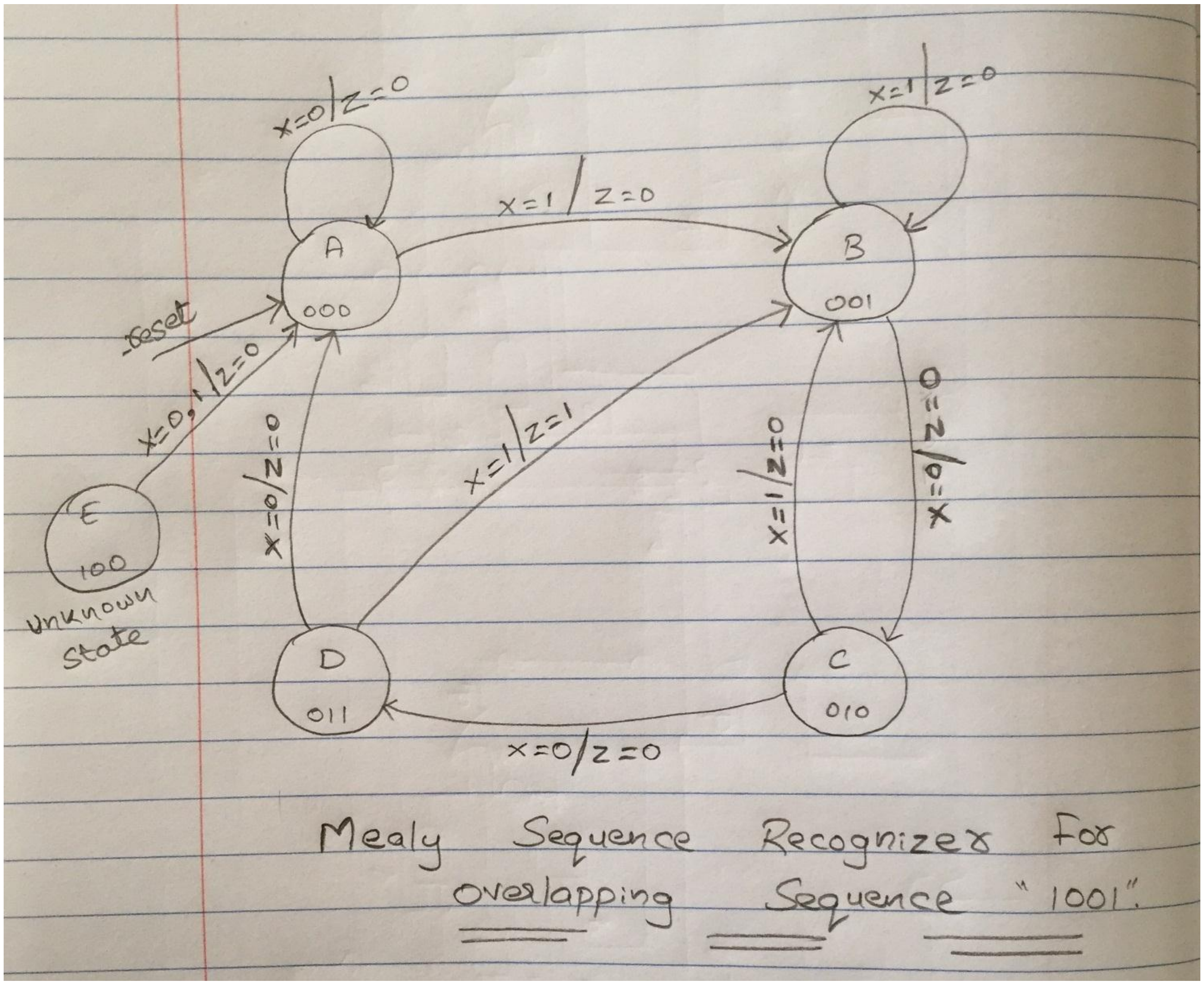
Output:



```
Chronologic VCS simulator copyright 1991-2014
Contains Synopsys proprietary information.
Compiler version I-2014.03-2; Runtime version I-2014.03-2;  Nov 24 11:59 2016
   0:           z = 0
  10:     x = 0
  20:    x = 0
  30:    x = 1
  40:    x = 0
  50:    x = 0
  60:    x = 1
  65:           z = 1
  70:    x = 1
  75:           z = 0
  80:    x = 0
  90:    x = 0
 100:    x = 1
 105:           z = 1
 110:    x = 0
 115:           z = 0
 120:    x = 0
 130:    x = 1
 135:           z = 1
$finish called from file "TestBench.v", line 35.
$finish at simulation time                  140
           V C S   S i m u l a t i o n   R e p o r t
Time: 140
CPU Time:      0.550 seconds;       Data structure size:   0.0Mb
Thu Nov 24 11:59:15 2016
[bibodij@titan:26]>
```

2. Considered Problem 5.11

a. Mealy's Sequence Recognizer for overlapping sequence 1001



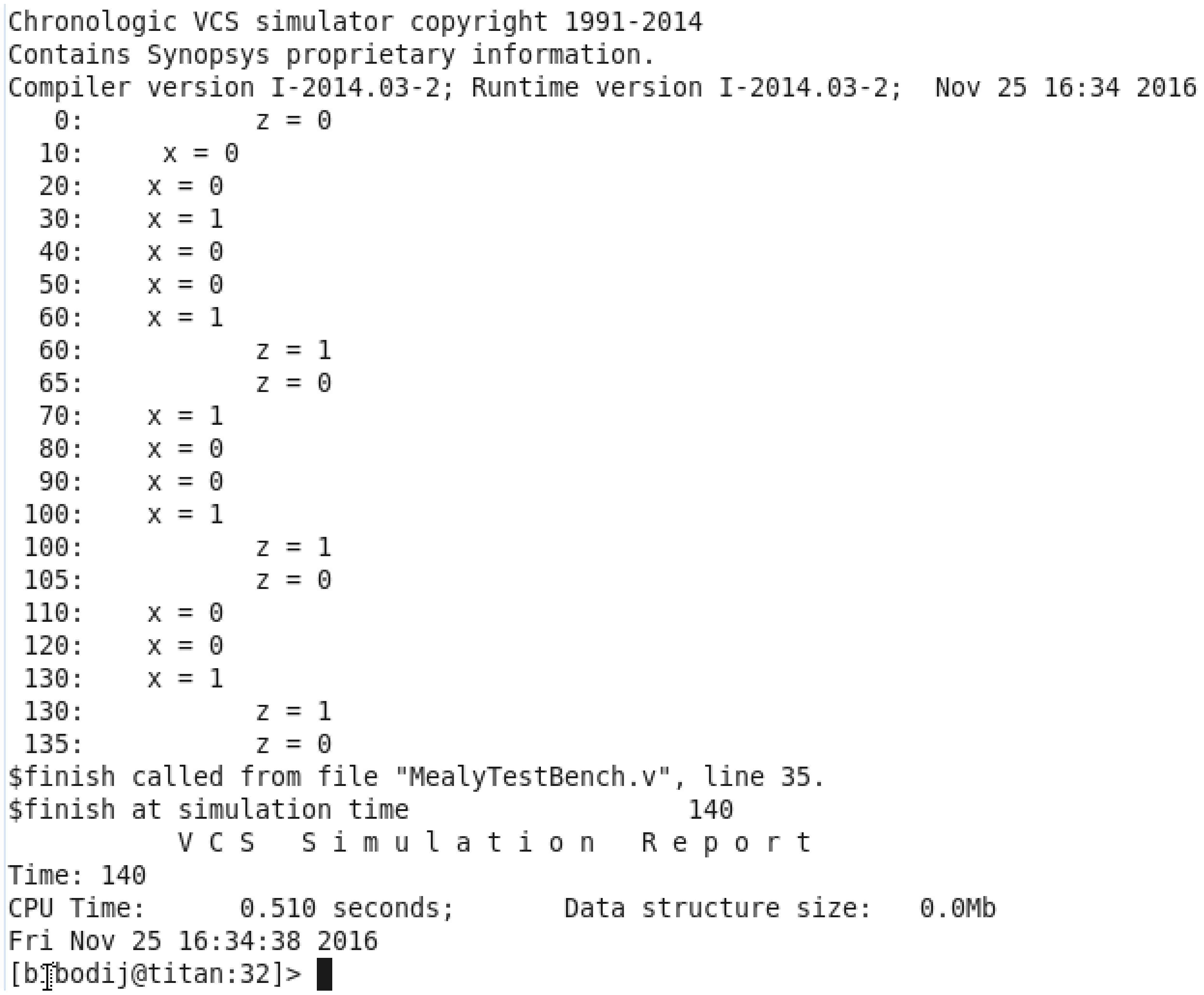
```
module MealyProject
(
        input clock,reset,x,
        output reg z
);

//assign binary encoded codes to the states A through D

parameter    A = 3'b000,
             B = 3'b001,
             C = 3'b010,
             D = 3'b011;

reg[2:0] current_state, next_state;
```

```
    //Section 1: Next State Generator (NSG)
    always@(*)
    begin
        casex (current_state)  // ignore unknown and high impedence (Z) inputs
            A:      if(x == 1)
                        next_state = B;
                    else
                        next_state = A;
            B:      if(x == 1)
                        next_state = B;
                    else
                        next_state = C;
            C:      if(x == 1)
                        next_state = B;
                    else
                        next_state = D;
            D:      if(x == 1)
                        next_state = B;
                    else
                        next_state = A;
            default:
                        next_state = 3'bxxx;
        endcase

    end

    //Section 2: Output Generator (OG)

    always@(*)
    begin
        if((x == 1) && (current_state == D) && (next_state == B ))
            z = 1'b1;
        else
            z = 1'b0;
    end

    //Section 3: FlipFlop

    always@(posedge clock,posedge reset)
    begin
        if(reset ==1)
            current_state <= A;
        else
            current_state <= next_state;
    end

endmodule
```

b. TestBench (MealyTestBench.v)

```
`include "MealyProject.v";

module MealyTestBench();
reg clock,reset,x;
wire z;

MealyProject projectTwo(clock,reset,x,z);

initial begin
$monitor("%4d:                    z = %b",$time,z);
clock=0;
reset=1;     //Resets the FlipFlop
x=0;
#10 reset=0;     //end reset
end

always
begin
#5clock=~clock; //generates a clock signal with period 10
end

initial begin  // One input per clock cycle
#10 x = 0; $display ("%4d:        x = %b",$time,x);
#10 x = 0; $display ("%4d:    x = %b",$time,x);
#10 x = 1; $display ("%4d:    x = %b",$time,x);
#10 x = 0; $display ("%4d:    x = %b",$time,x);
#10 x = 0; $display ("%4d:    x = %b",$time,x);
#10 x = 1; $display ("%4d:    x = %b",$time,x);
#10 x = 1; $display ("%4d:    x = %b",$time,x);
#10 x = 0; $display ("%4d:    x = %b",$time,x);
#10 x = 0; $display ("%4d:    x = %b",$time,x);
#10 x = 1; $display ("%4d:    x = %b",$time,x);
#10 x = 0; $display ("%4d:    x = %b",$time,x);
#10 x = 0; $display ("%4d:    x = %b",$time,x);
#10 x = 1; $display ("%4d:    x = %b",$time,x);
#10 $finish;
end

endmodule
```

Output



```
Chronologic VCS simulator copyright 1991-2014
Contains Synopsys proprietary information.
Compiler version I-2014.03-2; Runtime version I-2014.03-2;  Nov 25 16:34 2016
   0:           z = 0
  10:      x = 0
  20:     x = 0
  30:     x = 1
  40:     x = 0
  50:     x = 0
  60:     x = 1
  60:           z = 1
  65:           z = 0
  70:     x = 1
  80:     x = 0
  90:     x = 0
 100:     x = 1
 100:           z = 1
 105:           z = 0
 110:     x = 0
 120:     x = 0
 130:     x = 1
 130:           z = 1
 135:           z = 0
$finish called from file "MealyTestBench.v", line 35.
$finish at simulation time                  140
           V C S   S i m u l a t i o n   R e p o r t
Time: 140
CPU Time:      0.510 seconds;       Data structure size:   0.0Mb
Fri Nov 25 16:34:38 2016
[bjbodij@titan:32]> █
```