Faculty of Engineering & Technology

Electrical & Computer Engineering Department

Advanced Digital Systems Design-ENCS3310

Project Report

Prepared by:

Name: Sondos Farrah **ID:** 1200905

Instructor: Dr. Abdallatif Abuissa

Section: 2

Date: 30.8.2023

Table of Contents

Table of Figures

List of Tables

1. Brief introduction and background

1.1 About project

In this project, we will design a MOOORE FSM for a sequence detector that detects the sequence 1011. by using Active-HDL student Edition to write Verilog codes, simulate the system and check the output.

We designed it by two different implementations, structural using T-Flip Flops and combination logic, then behavioral description uses for verification. The verification implemented in comparison between the structural and behavioral codes.

1.2 T-Flip Flop

A T flip-flop is a type of digital circuit that can store one bit of information. The T flip-flop has only one input, T, which toggles the output state when it is high and holds the previous state when it is low. The characteristic equation for a T flip-flop is given by:

$$Q(t+1) = T \oplus Q(t)$$

The state table for TFF is given by table1 below:

Table 1: T-Flip Flop state table

CLK	T	Q(t+1)
0	×	No change
1	0	No change
1	1	~Q

1.3 MOOORE system

The MOOORE system is a system in which the output is independent of the input. The output depends of the present state ONLY.

1.4 Structural and behavioral modeling

- **Behavioral model** is a way of describing the function of a design as a set of concurrent algorithms. In this model the designer doesn't involve in the component or how the circuit structure.
- **Structural model** is a way of describing functions defined using basic components such as inverters, multiplexers, adders, decoders and basic logic gates.

2. Design philosophy & Results

2.1 Handwritten solution:

To solve this project, I start with synchronous sequential circuit analyses for design, we need state diagram, state table, Kmap for next state and output, then block diagram to imagine how the circuit work to detect 1011 sequence.

2.1.1 State Diagram

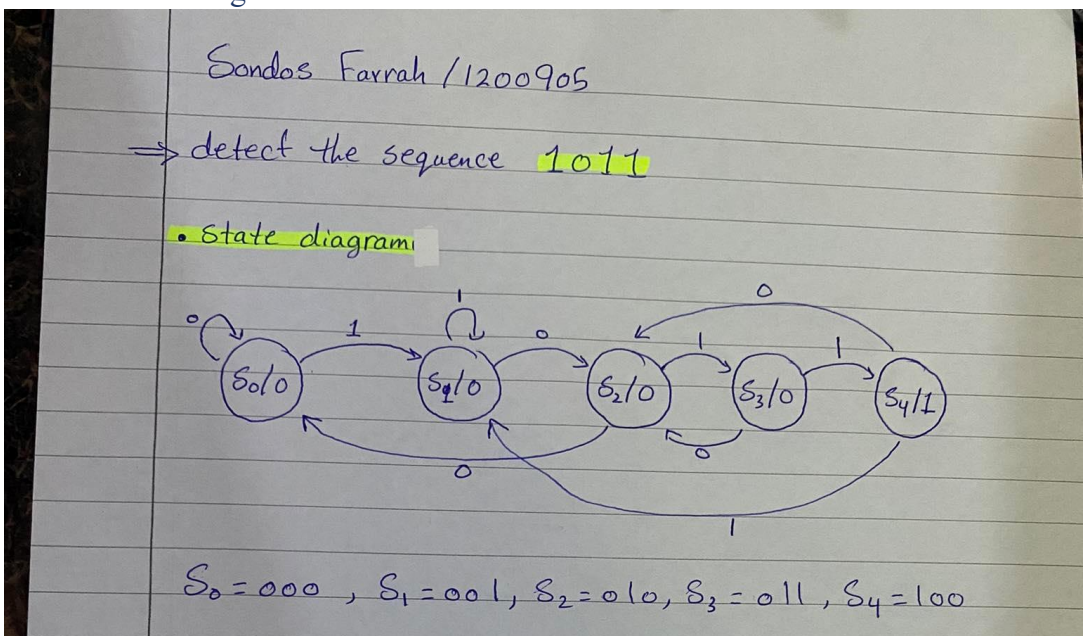

Figure 1: State Diagram

2.1.2 State Table

Table 2: State table

Present State			input	Next State			output	TFF input		
Q2	Q1	Q0	X	Q2(t+1)	Q1(t+1)	Q0(t+1)	Y	T2	T1	T0
0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	0	0	0	1
0	0	1	0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0	0	0	0
0	1	0	0	0	0	0	0	0	1	0
0	1	0	1	0	1	1	0	0	0	1
0	1	1	0	0	1	0	0	0	0	1
0	1	1	1	1	0	0	0	1	1	1
1	0	0	0	0	1	0	1	1	1	0
1	0	0	1	0	0	1	1	1	0	1

2.1.3 Kmap for Next-State & output

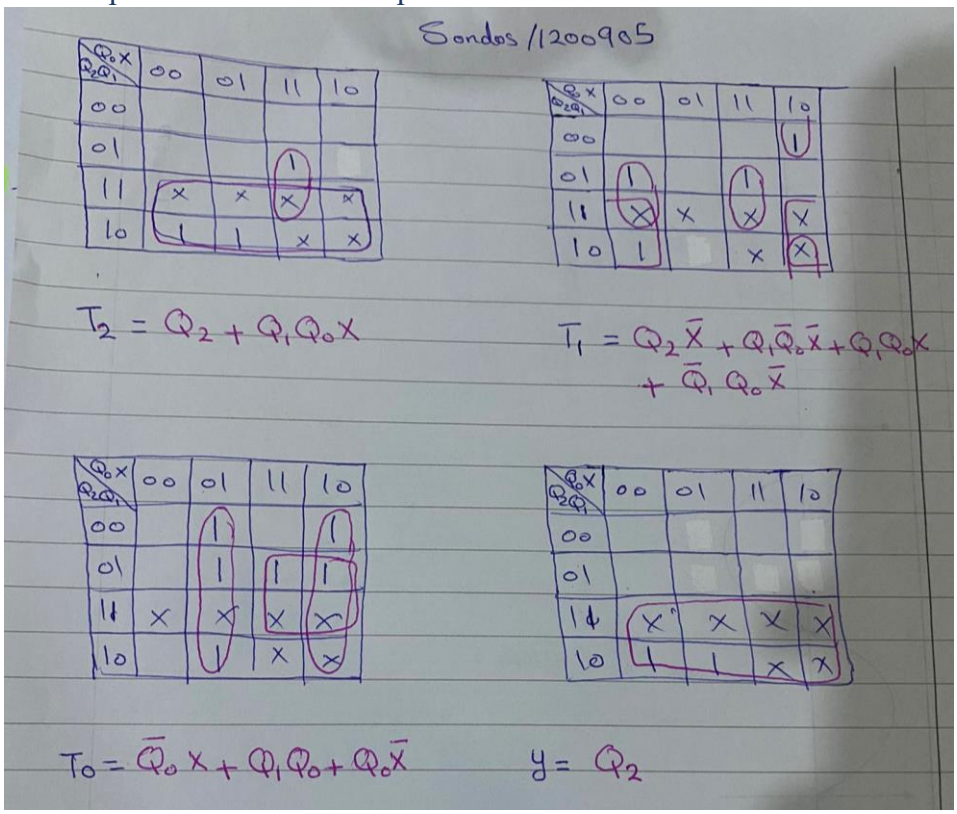

Figure 2: K-map for Next-state and Output

2.1.4 Block diagram

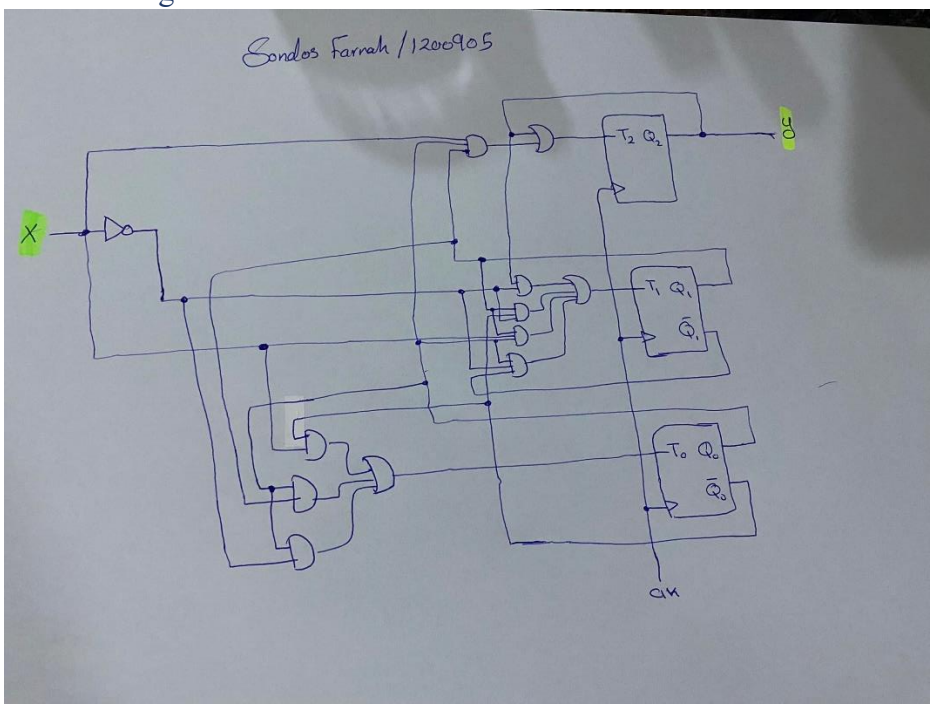

Figure 3: Block diagram

2.2 Structural model:

Doing steps in part 2.1 help me to know the number of T-Flip Flop that should I use, and the equations for Next-state and output to use in structural model.

2.2.1 T-Flip Flop

First of all, I implemented a TFF code to call it in structure code.

```
module TFF(T, clk,rst,Q);//sondos 1200905
     input T, clk,rst;
     output reg Q;
     always_ff @(posedge clk, negedge rst) begin
    if (~rst) begin
      Q  <= 0;
    end else if (T) begin
      Q  <= ~Q;
    end
  end
endmodule
```

Figure 4: T-Flip Flop code

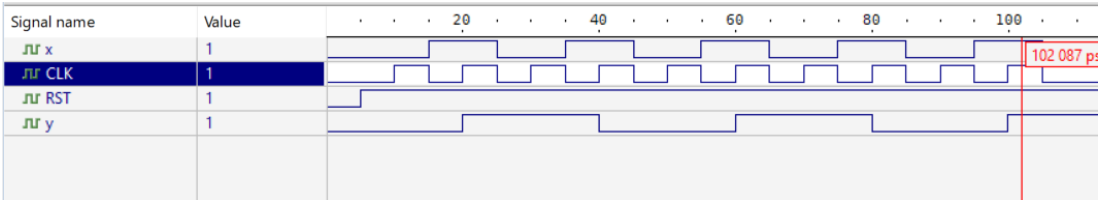

Figure 5: T-Flip Flop Wave

2.2.2 Hole structural code

```
//sondos 1200905
module structural(x,CLK,RST,y);
   input x,CLK,RST;
   output y;
   wire [2:0]Q;
   wire [2:0]T;
//Flip-flip input equations
     assign T[2]= Q[2]|(Q[1] & Q[0] & x);
     assign T[1]= (Q[2] & !x)|(Q[1] & !Q[0] & !x)|(Q[1] & Q[0] & x)|(!Q[1] & Q[0] & !x);
     assign T[0]= (!Q[0] & x)|(Q[1] & Q[0])|(Q[0] & !x);
//Output equation
   assign y = Q[2];
//Instantiate T flip-flops
   TFF F0 (T[2],CLK,RST,Q[2]);
   TFF F1 (T[1],CLK,RST,Q[1]);
   TFF F2 (T[0],CLK,RST,Q[0]);
endmodule
```

Figure 6: Structural code

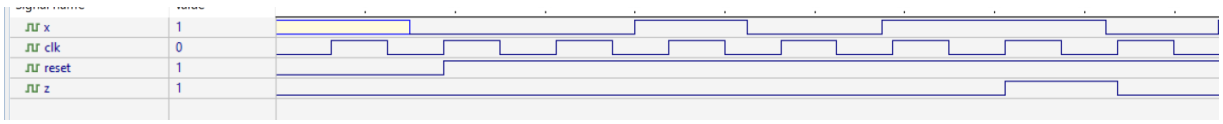

Figure 7: Structural Wave

For structural code, Q is present state respectively Q2Q1Q0, T is input of T-Flip Flop respectively T2T1T0. I used array form because its easy for me and less variables number.

As shown in figure7: when sequence input is 1011 the output will be 1, otherwise its 0, detected sequence in almost 7th clk.

I have 3 T-Flip Flops because I used 4 states (s0, s1, s3, s4) in **state diagram**.

2.3 Behavioral model

Behavioral model depends on state diagram basically.

```
//sondos 1200905
module Behavioral(x, clk, rst, y);
    input x, clk, rst;
    output reg y;
    reg [2:0] state;
    parameter s0 = 3'b000, s1 = 3'b001, s2 = 3'b010, s3 = 3'b011, s4 = 3'b100;

    always @(posedge clk or negedge rst) begin
        if (~rst)
            state <= s0;
        else begin
            case (state)
                s0: if (x) state = s1;
                s1: if (!x) state = s2;
                s2: if (x) state = s3; else state = s0;
                s3: if (x) state = s4; else state = s2;
                s4: if (x) state = s1; else state = s2;
            endcase
        end
    end

    always @(posedge clk or negedge rst) begin
        if (~rst)
            y <= 1'b0;
        else begin
            case (state)
                s0, s1, s2, s3: y <= 1'b0;
                s4: y = 1'b1;
            endcase
        end
    end
endmodule
```

Figure 8: Behavioral code

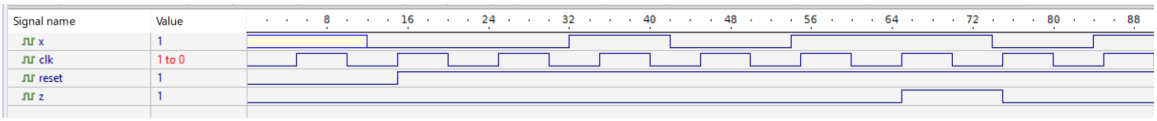

Figure 9: Behavioral Wave

This model is easy one for me, behavioral model uses (always) to change variables every positive edge for clock or every negative edge for reset. That means when clock =0 no change and when reset =0 assign 0 to variables.

As shown in figure7: when sequence input is 1011 the output will be 1, otherwise its 0, detected sequence in almost 7th clk.

2.4 Codes Testing

For testing I use the same testbench for ALL codes in this project. The code shown in figure bellow

```
`timescale 1ns / 1ps

module testbench;
    // Inputs
    reg x;
    reg clk;
    reg reset;
    // Outputs
    wire z;
    // Instantiate the Unit Under Test (UUT)
    Behavioral uut (x,clk,reset,z);

initial
    begin
        clk = 1'b0;
        reset = 1'b0;
        #15 reset = 1'b1;
    end

always #5 clk = ~ clk;

initial begin
        #12 x = 0;#10 x = 0 ; #10 x = 1 ; #10 x = 0 ;
        #12 x = 1;#10 x = 1 ; #10 x = 0 ; #10 x = 1 ;
        #12 x = 1;#10 x = 0 ; #10 x = 0 ; #10 x = 1 ;
        #12 x = 0;#10 x = 1 ; #10 x = 1 ; #10 x = 0 ;
        #10 $finish;
    end


endmodule
```

Figure 10: TestBench code

To test any code, I just change the module which I called in testbench. In figure 10 above, I was tested behavioral code.

Sets time scale = 1n/1p to show wave form clearly.

In this code, initially clk=0, reset=0, then after 15 unit of time set reset=1 to test all options.

Every 5 units of time change clock.

For input x, this sequence I did to observe the sequence 1011 to show if it detected or not.

2.5 Verification

To test the design using verification, by compare the output of structural code and output of behavioral code. I used model that the behavioral code gets the output of structural code as input. Then test the two outputs. As shown in figure bellow.

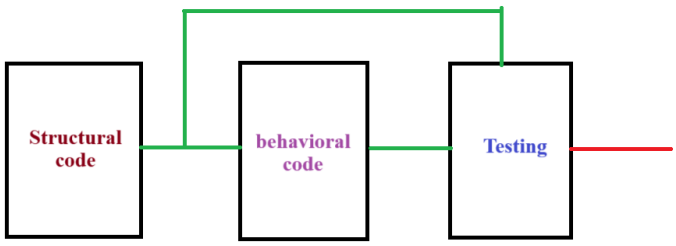

Figure 11: Block diagram for Verification

First of all, I wrote code to compare the two outputs that I mentioned before (block Testing in figure 11 above).

```
//sondos 1200905
module testcode(x,clk,rst,y);
    input x, clk,rst;
    output y;
    wire Sout,Bout;
    structural s1(x,clk,rst,Sout);
    Behavioral b1(Sout,clk,rst,Bout);
    assign y = (Sout == Bout) ? 1'b1 : 1'b0;
    always @(posedge clk, negedge rst) begin
     $monitor("Time %0d Structural_output=%b  Behavioral_output=%b Test_Output=%b", $time,Sout,Bout,y);
     end
endmodule
```

Figure 12: comparison code

In this code, I used monitor to display variables (time, structural output, behavioral output, test output) whenever a clock or reset changes during simulation run. This step enables me to track the result.

I tested this code in figure 12 above also using the same testbench in part **Codes Testing**.

Figure 13: Verification Wave

```
run 100ns
# KERNEL: Time 0 Structural_output=0  Behavioral_output=0 Test_Output=1
# KERNEL: Time 65000 Structural_output=1  Behavioral_output=0 Test_Output=0
# KERNEL: Time 75000 Structural_output=0  Behavioral_output=0 Test_Output=1
# KERNEL: Time 95000 Structural_output=1  Behavioral_output=0 Test_Output=0
# KERNEL: stopped at time: 100 ns
```

Figure 14:Verification Output

The output of monitor shown in figure 14, as we see the output will be 1 if and only if structural output = behavioral output.

3. Challenges and difficulties

I have faced few challenges and difficulties in this project, I will mention here.

- Doing state table by hand and Kmaps It needs focus because any mistake at the beginning will lead to a series of errors and thus a wrong result.

- Behavioral code is easy except know when must doing blocking or non-blocking assignment. In my code I used blocking assignment because non-blocking causes a wrong delay in the result.

- The most important challenge I have that how can I test my codes. Because doing normal testbench as we learned don't show the sequence 1011 as input, Therefore the result cannot be checked. The normal testbench show input as sequence 101010 and so on, that is I can't check the output. I solve this by set a group of sequence that can show the sequence 1011 as input (you can see it in part **codes testing**).

4. Conclusion and Future works

To conclude, we learned how to design a MOOORE FSM for a sequence detector that detects any sequence. We understood the importance of Flip Flops in synchronous sequential circuits, how and when they work as memory and how to use it structurally to design circuits.

We have realized the effectiveness of testbenches and verification and how they make it easier for us to test any code instead of non-useful way to put timing Not automatic.

Finally, we used Verilog in a deeper way and became more familiar with writing codes and testing systems by using the Active-HDL tool to write codes and simulate our project and see the results.

For the future, I hope that the department will offer more courses related to advanced digital systems as elective courses so that students can delve deeper into these topics and work on them after graduation.