FACULTY OF ENGINEERING AND TECHNOLOGY
ELECTRICAL AND COMPUTER ENGINEERING DEPARTMENT
ADVANCED DIGITAL DESIGN ENCS3310
COURSE PROJECT

Summer 2023

Prepared By: Ody shbayah-1201462

Section 2.

Dr: Abd_elatif- abu_essa.

Date: 22/8/2023

Brief introduction and background.

Introduction :

The project focuses on the design and implementation of a Moore Finite State Machine (FSM) aimed at detecting the specific binary sequence "1011" within a stream of binary input data. Finite State Machines are essential components in digital logic design, used to model and control sequential behavior. The project utilizes T-flip-flops as key building blocks to create a Moore FSM tailored to recognize the target sequence.

Background :

Digital systems often need to identify specific patterns in sequences of 0s and 1s. A finite state machine (FSM) is a structured way to handle these patterns. The Moore FSM is one type of FSM where outputs connect to different states, and changes between states are triggered by inputs.

The T-flip-flop is a component that changes its state when given a particular signal. This feature is important for this project. It enables smooth transitions between states when detecting specific binary patterns. By connecting multiple T-flip-flops and designing their interactions, we can create a Moore FSM capable of recognizing complex sequences.

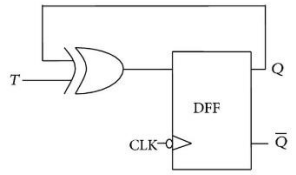
As an example, let's consider the sequence "1011." This sequence helps illustrate the project's core ideas. Detecting "1011" involves responding to specific combinations of inputs and moving through different states until reaching the final accepted state. The design process includes creating a diagram representing the sequence detection process, assigning states to T-flip-flops, determining transitions between states based on inputs, and setting up the proper output logic for each state.

Design philosophy

Components for the project :

- T-flip-flops
  The "T Flip Flop" is designed by passing the AND gate's output as input to the NOR gate of the "SR Flip Flop". The inputs of the "AND" gates, the present output state Q, and its complement Q' are sent back to each AND gate. The toggle input is passed to the AND gates as input. These gates are connected to the Clock (CLK) signal. In the "T Flip Flop", a pulse train of narrow triggers are passed as the toggle input, which changes the flip flop's output state. The circuit diagram of the "T Flip Flop"

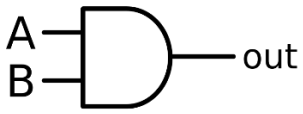
The "T Flip Flop" is formed using the "D Flip Flop". In D flip - flop, the output after performing the XOR operation of the T input with the output "Q_{PREV}" is passed as the D input. The logical circuit of the "T-Flip Flop" using the "D Flip Flop" is given below:



- And gates

The AND gate plays an important role in the digital logic circuit. The output state of the AND gate will always be low when any of the inputs states is low. Simply, if any input value in the AND gate is set to 0, then it will always return low output(0).

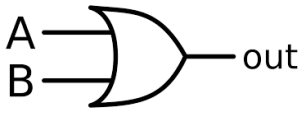
The logic or Boolean expression for the AND gate is the logical multiplication of inputs denoted by a full stop or a single dot as A.B=Y and the figure below shows the and gate circuit :



- Or gates

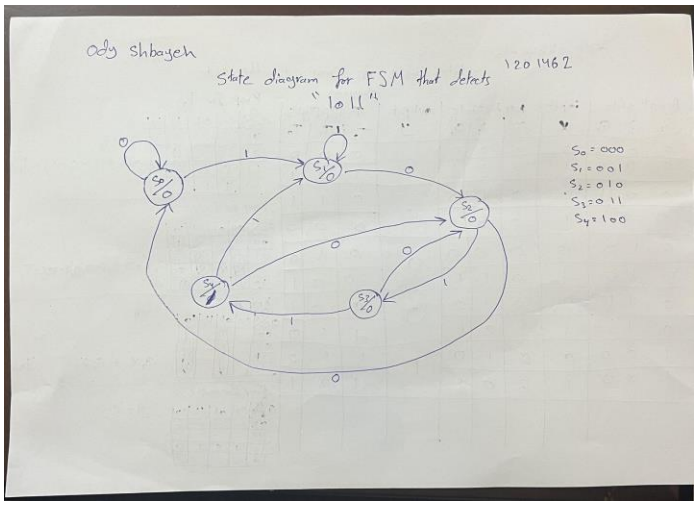
The OR gate is a mostly used digital logic circuit. The output state of the OR gate will always be low when both of the inputs states is low. Simply, if any input value in the OR gate is set to 1, then it will always return high-level output(1).

The logic or Boolean expression for the OR gate is the logical addition of inputs denoted by plus sign(+) as A+B=Y and the figure below shows the Or gate circuit :

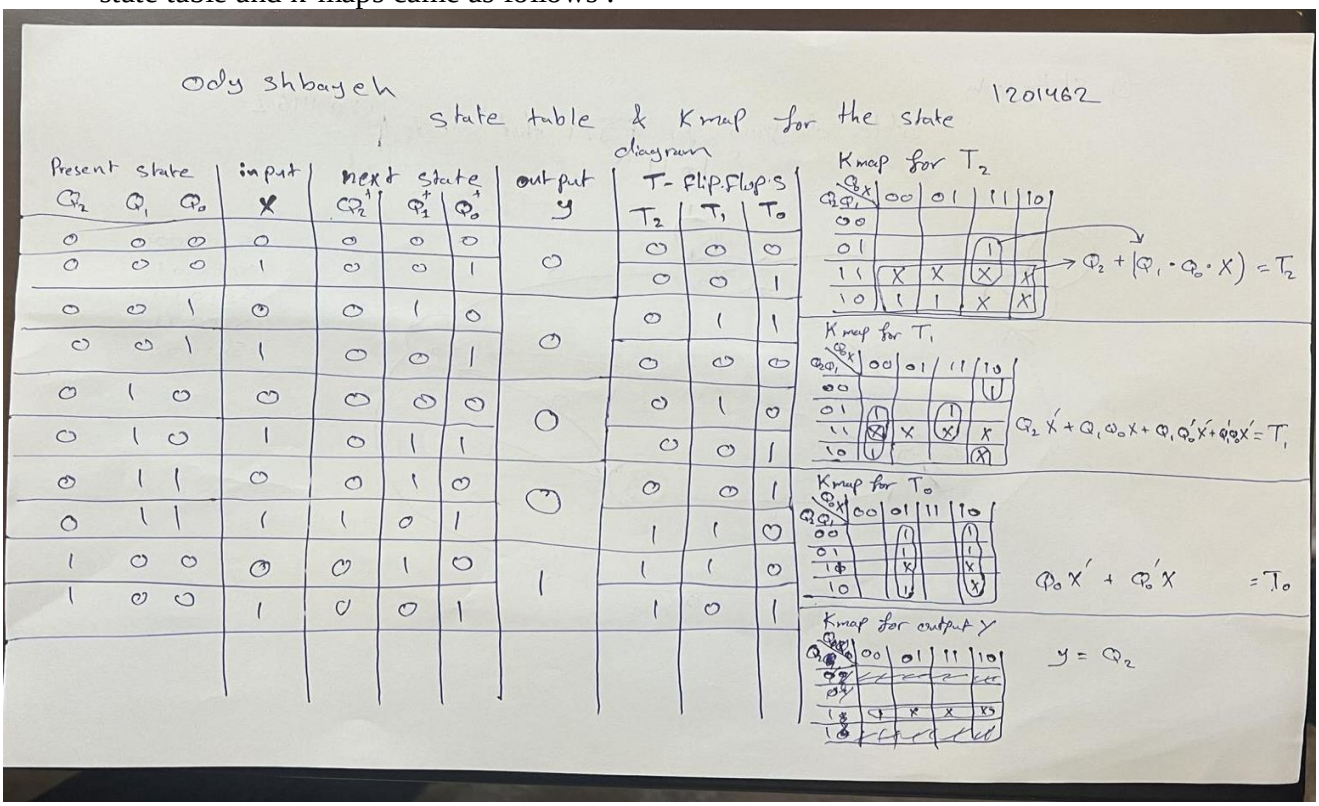


Theory for the outputs

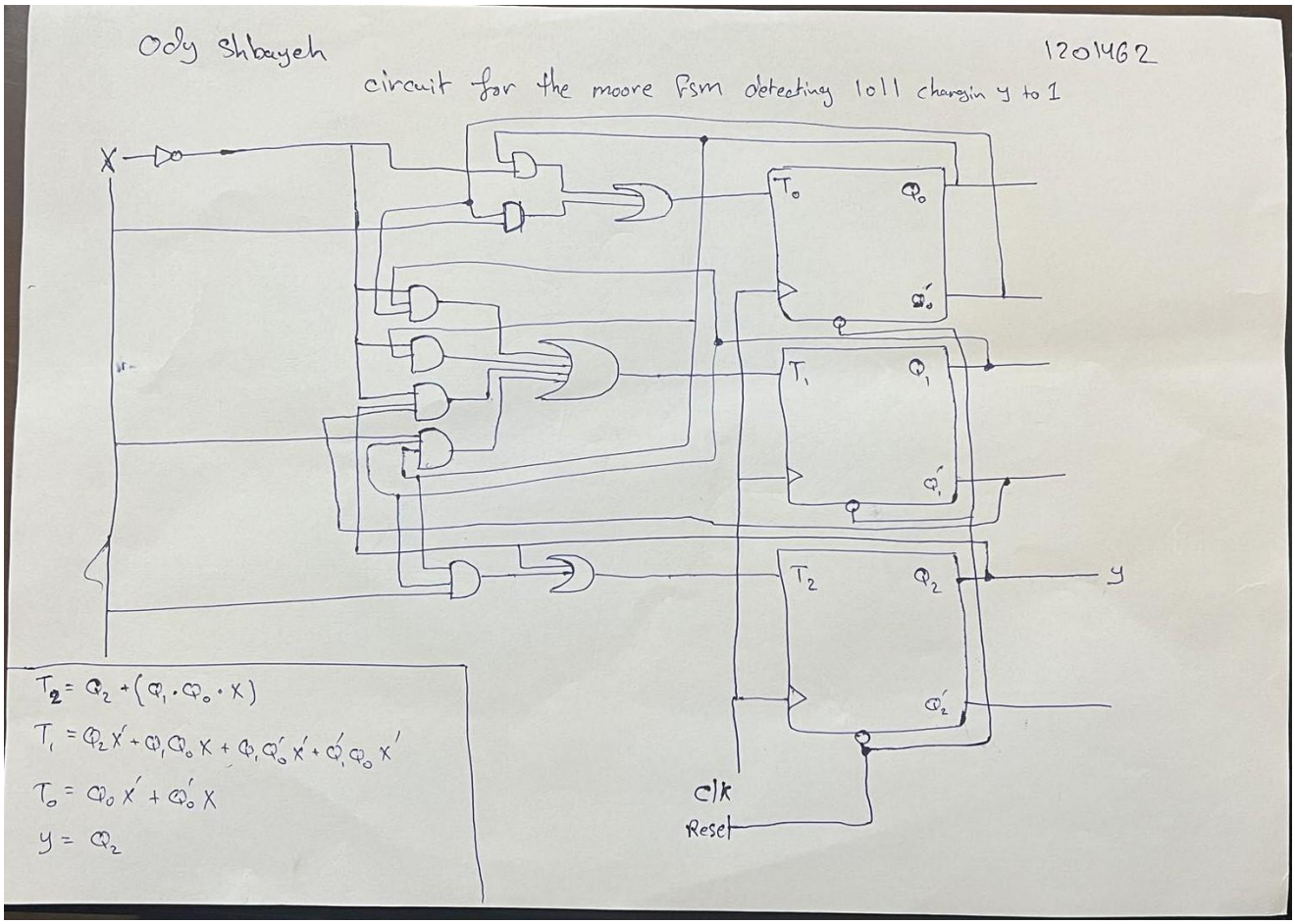
- To create a moore finite state machine for a sequence detector that detects the sequence 1011 we need to write down the state diagram which specifies how the circuit should flows to reach the state of detecting the sequence and when the sequence is completely detected the output "y" status changes from 0 to 1 for a period of time till it reads another pattern that breaks the sequence and the state diagram for the circuit was as follows :



- To create the state table we look back to the state diagram and read it as the present state and input and what the next state should be and what in the output when the next state is reached after that I derived the k-maps to implement equations for the inputs to the T-flip-flops and the state table and k-maps came as follows :



- finally we can draw the circuit depending on the next state equations for the T-flip-flops we implemented earlier which came as follows :



Codes for the circuit :

- T-flip-flops

I implemented the T-flip-flop behaviorally depending on the positive edge of the clock or the negative edge of the reset to obtain the output :

```
1  //odyshbayeh-1201462
2
3  //for this T-flipflop module i used the notes from the book
4  //i can also define the same module by entering the clk & reset input into a and gate for the input edge timer for the T-flipflop
5  //as the notes describes.
6  module t_ff (t, clk, reset, q);
7
8      input t, clk, reset;
9      output reg q;
10
11     always @(posedge clk or negedge reset)
12         begin
13         if (~reset)
14             q <= 1'b0;
15         else
16             q <= q ^ t;
17     end
18
19 endmodule
20
```

- Circuit module structurally :

I connected the circuit structurally by assigning values to the inputs for the T-flip-flops and entering them into the module by calling them in the code as follows :

```
//for this module i figured the state diagram and wrote down the state table and got the 4 equations for the circuit
//to detect the sequence 1011 which are : T0,T1,T2,Y <== were the output from the k map and build the design structurally
module sequence_detector (clk,reset,x,y);

    input  clk,reset,x;
    output y;

    wire t0, t1, t2;
    reg q0, q1, q2;

    t_ff tf0 (.t(t0), .clk(clk), .reset(reset), .q(q0));
    t_ff tf1 (.t(t1), .clk(clk), .reset(reset), .q(q1));
    t_ff tf2 (.t(t2), .clk(clk), .reset(reset), .q(q2));

    assign t0 = (q0 & ~x) | (~q0 & x);
    assign t1 = (q2 & ~x) | (q1 & ~q0 & ~x) | (q1 & q0 & x) | (~q1 & q0 & ~x);
    assign t2 = q2 | (q0 & q1 & x);

    assign  y = q2;

endmodule
```

- Circuit module behaviorally :

I also made a new module for the same circuit depending on the outputs of the state table and k-map results for the T-flip-flops to execute the code and work as specified by assigning the posedge and the negedge for the clk and reset to detect the sequence 1011 and the aim for this module is to compare the output from it to the structural module and see the comparisons between them and the implemented module code as follows :

```
//this is the module for the circuit but impleminted behaviourly so that we can see the out put from the structural circuit
// and from the behavioural circuit and compare the outputs from the modules.
module sequence_detector_behavioral(clk, reset, x, y);
   input clk, reset, x;
   output reg y;

   wire t0, t1, t2;
   reg q0, q1, q2;

   assign t0 = (q0 & ~x) | (~q0 & x);
   assign t1 = (q2 & ~x) | (q1 & ~q0 & ~x) | (q1 & q0 & x) | (~q1 & q0 & ~x);
   assign t2 = q2 | (q0 & q1 & x);

   always @(posedge clk or posedge reset)
        begin
      if (reset)
         q0 <= 1'b0;
      else
         q0 <= t0;
   end

   always @(posedge clk or posedge reset)
        begin
      if (reset)
         q1 <= 1'b0;
      else
         q1 <= t1;
   end

   always @(posedge clk or posedge reset)
        begin
      if (reset)
         q2 <= 1'b0;
      else
         q2 <= t2;
   end

   always @(posedge clk or posedge reset)
        begin
      y <= q2;
   end

endmodule
```

- Test bench for both behavioral and structural modules :

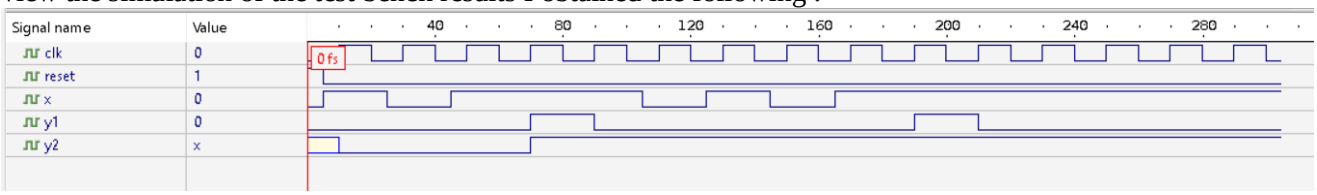
I implemented the test bench as the notes sent in retaj specified and came out with the following test bench that uses a single clock and a reset clock also the input 'x' and both the outputs from the behavioral and structural modules and it came as follows :

```
86
87  //odyshbayeh-1201462
88  module tb_sequence_detector();
89
90      reg clk;
91      reg reset;
92      reg x;
93      wire y1, y2;
94
95      sequence_detector u1 (.clk(clk), .reset(reset), .x(x), .y(y1));
96      sequence_detector_behavioral u2 (.clk(clk), .reset(reset), .x(x), .y(y2));
97
98      initial begin
99          clk = 0;
100         reset = 1;
101         x = 0;
102
103         // resetting the reset input so that the TFF starts working as expected
104         #5 reset = 0;
105
106         // applying a random sequence
107         x = 1;
108         #20 x = 0;
109         #20 x = 1;
110         #20 x = 1;
111         #40;
112         // applying the required input delays to detect the sequence 1011
113         x = 0;
114         #20 x = 1;
115         #20 x = 0;
116         #20 x = 1;
117         #40;
118
119        #100 $finish;
120
121     end
122     always #10 clk = ~clk;
123
124 endmodule
125
```

1. At first I set every input to the circuit to it's initial state which is 0 except for the reset clock which the initial case for it should be 1 .
2. Then I turned off the reset mode so the circuit can work as expected , and I've applied a random clock pulses for the input 'x' to change to see if it will detect anything except the required input.
3. After that I inserted the required time delays for the input to see if the circuit will detect the required sequence and change the output from 0 to 1 for both structural and behavioral modules of the circuit.
4. Also to be noted that the clock cycle is always after 10ns "every 10ns it changes the edge of it"

- Results

After setting the test bench as top-level-entity and initializing the simulation and opening a new waveform to view the simulation of the test bench results I obtained the following :



The output of the two modules 'y1' and 'y2' changed from 0 to 1 it means that they've detected a sequence of 1011 as required.

There's a error on the output 'y2' for the first it gives an a don't care condition which is a strange outcome for the module and I couldn't figure out why , also for the same output there's an error which makes it toggles '1' even after the input has changed and the sequence broke I believe that it's a dataflow logic error or time delaying error because the behavioral module worked at the same time as the structural module did , I tried to fix the error but couldn't so I kept it on believing that at the end ; the module has detected the sequence as required.

- Conclusion and Future works

the project successfully designed and implemented a Moore FSM using T-flip-flops to detect the sequence "1011". The process involved careful state diagram analysis, logic equation derivation, and module implementation. While there was a minor issue with the behavioral module, the project effectively demonstrated the principles of sequential logic and pattern detection using FSMs.

For future works I hope to develop the behavioral module to increase the efficiency of the design and see the specified outcome as the problem suggested and take another part-in working on another design that involves designing a circuit that simulates the behave of the band-pass filters that turns on only on low freq's.

- References used in the report

https://www.javatpoint.com to get all information about the components in the design and the principles of how the moore and mealy fsm works and how to derive the outputs from them.
google search engine to get the photos of the components .