UNIVERSITY OF INFORMATION TECHNOLOGY

COMPUTER ENGINEERING

REPORT LAB3 VERILOG
DESIGN FSM AND SRAM

Student: *Trương Thiên Quý*
Student code: *23521321*

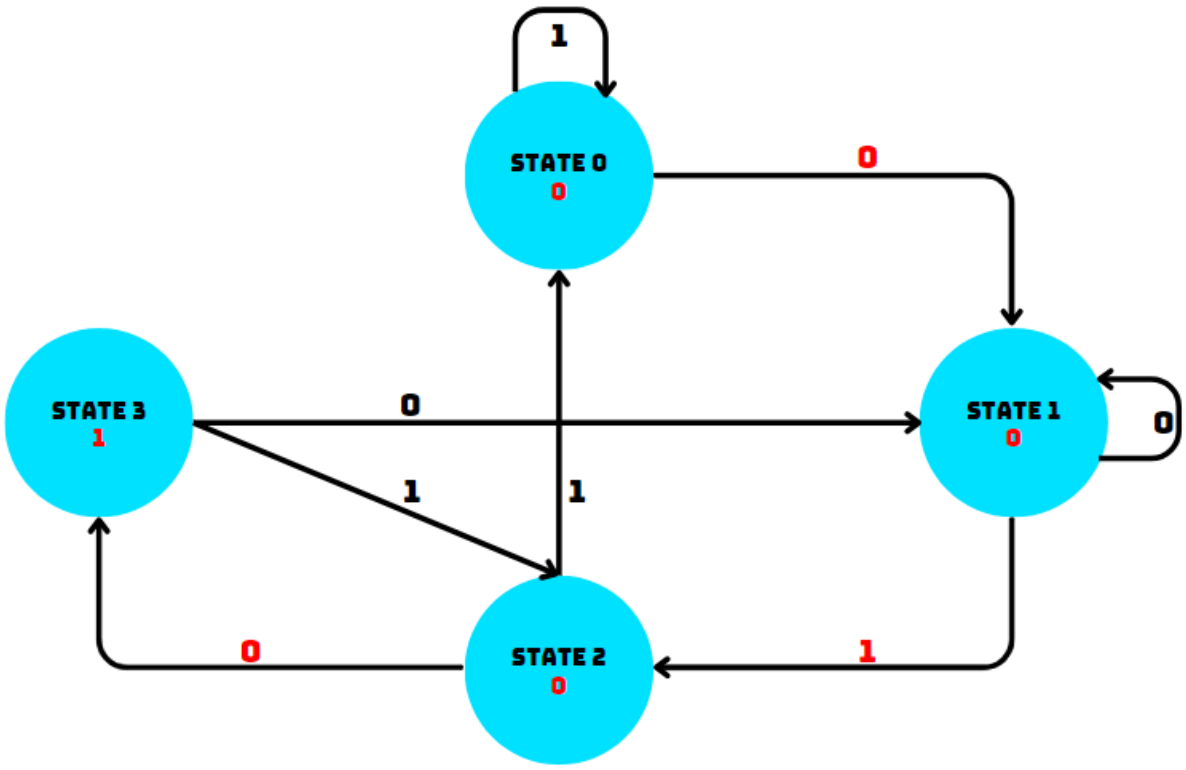
Table of Contents

A. FSM:

1. Request:

Sử dụng ngôn ngữ Verilog HDL thiết kế một mạch tuần tự theo mô hình máy trạng thái kiểu Moore. Hệ tuần tự này có chức năng phát hiện 3 bit ngõ vào (X) liên tiếp có giá trị là 010 thì ngõ ra Z = 1.

2. Diagram:



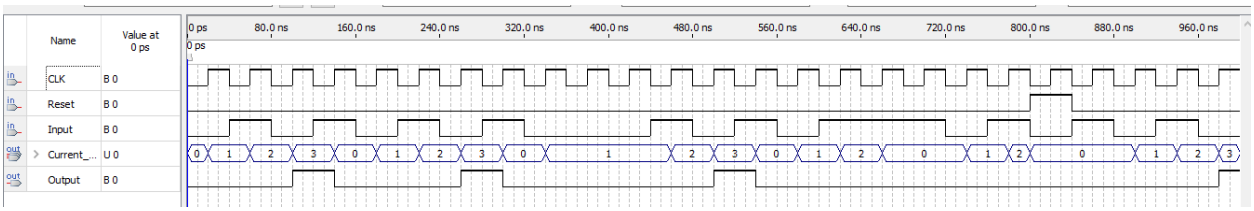
3. Verilog Code:

```
1   module FSM(Output, Current_State, Input, Reset, CLK);
2   //define_IO
3      input Input, Reset, CLK;
4      output reg Output;
5
6   //define_wire/reg
7      output reg [1:0] Current_State;
8      reg [1:0] Next_State;
9   //body
10     always @(posedge CLK or posedge Reset) begin
11        if(Reset)
12           Current_State <= 2'b00;
13        else
14           Current_State <= Next_State;
15     end
16
17     always @(Input or Current_State) begin
18        case (Current_State)
19           2'd0: begin
20                    if (~Input)
21                       Next_State <= 2'd1;
22                    else
23                       Next_State <= 2'd0;
24                    Output <= 0;
25                 end
26           2'd1: begin
27                    if (Input)
28                       Next_State <= 2'd2;
29                    else
30                       Next_State <= 2'd1;
31                    Output <= 0;
32                 end
33           2'd2: begin
34                    if (~Input)
35                       Next_State <= 2'd3;
36                    else
37                       Next_State <= 2'd0;
38                    Output <= 0;
39                 end
40           2'd3: begin
41                    Next_State <= 2'd0;
42                    Output <= 1;
43                 end
44           default: begin
45                       Next_State <= 2'd0;
46                       Output <= 0;
47                    end
48        endcase
49     end
50  endmodule
```

Explain:

- There are 4 states perform for 4 cases:
  - State 1 (00): the chain we must recognize is 010, when the first input equal 0 the it changes to state 2 (01), now we have had the first bit of chain, else it still State 0 (mean don't have any fit bit). Output is 0.
  - State 2 (01): we already have bit 0 from State 1, we must recognize 010, so if the next input is 1, it changes to State 3(10), else it still in State 0 because the input equal 0 (the first bit of chain). Output is 0.
  - State 3(10): we already have 01 after state 2, and we must recognize 010, so to change to state 4 (11), the next input must be 0. Else it comebacks state 0(01) because if input is 1, the chain now is 011, very different with the chain we must recognize (010). Output is 0.
  - State 4(11): we already have 010 after state 3, if the next input is 0, we will have 0100, easy to see that the last bit is 0 – the first bit in chain 010, so it change to state 2(02). Else if the next input is 1, we will have 0101, similar the previous, we easy to see 2 last bit is 01 – 2 first bit in chain 010, so it change to state 3 (01). We completed recognize 010, so output is 1.
- Begin, the state is 00, depend on input it will change to different state.

4. Simulation:



B. SRAM:

1. Request:

Sử dụng ngôn ngữ Verilog HDL, hiện thức thiết kế bộ nhớ dữ liệu (Data Memory) SRAM có các tín hiệu sau: Address[31:0], WriteData[31:0], ReadData[31:0], WriteEn, ReadEn và viết testbench kiểm tra chức năng trên trên phần mềm mô phỏng ModelSim.

2. Verilog Code:

```
1  module SRAM (Address, WriteData, ReadData, WriteEn, ReadEn);
2  //define_IO
3     input [31:0] Address, WriteData;
4     output reg [31:0] ReadData;
5     input WriteEn, ReadEn;
6
7  //define_wire/reg
8     reg [31:0] memory [4294967295:0]; // Khai báo một mảng 2D 255x32
9
10 //body
11    always begin
12          if (WriteEn) begin
13              memory[Address] <= WriteData; // Ghi dữ liệu vào địa chỉ chỉ định khi WriteEn được kích hoạt
14              ReadData <= 32'dZ;
15          end
16          else if (ReadEn) begin
17              ReadData <= memory[Address]; // Đọc dữ liệu từ địa chỉ chỉ định khi ReadEn được kích hoạt
18          end
19    end
20
21 endmodule
```

Explain:

- First we check WriteEn signal. If it equals 1, SRAM only perform write function, now output = Z.
- After that, if WriteEN signal equal 0, we check ReadEN signal, ef ReadEN signal equal 1, SRAM only perform write function, now output = data in the input address (if in that address don't have data, output equal X). Else do nothing.

3. Simulation