TRƯỜNG ĐH CÔNG NGHỆ THÔNG TIN

Đại học Quốc gia TPHCM

Họ tên: Dương Anh Khôi

GVHD: Tạ Trí Đức

MSSV: 22520696

Khoa: Kỹ thuật Máy tính

Báo cáo: CE213 – Lab3

UIT

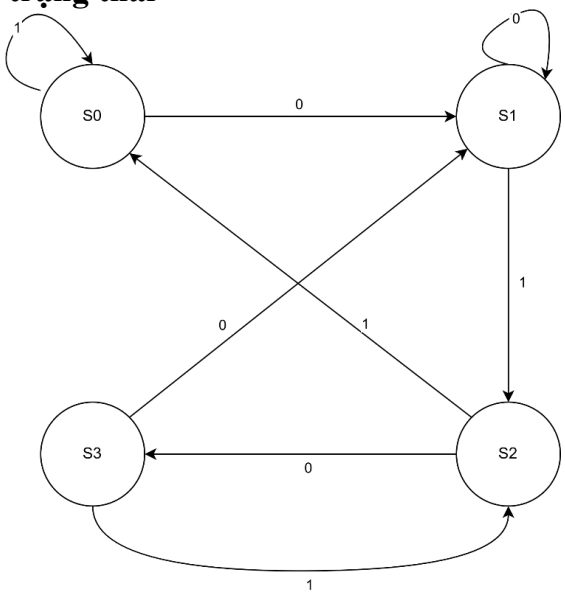
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

BÁO CÁO THỰC HÀNH BÀI 3

THIẾT KẾ MẠCH TUẦN TỰ BẰNG MÔ HÌNH MÁY TRẠNG THÁI HỮU HẠN

I. Thiết kế mạch tuần tự theo mô hình máy trạng thái kiểu moore phát hiện chuỗi (010)

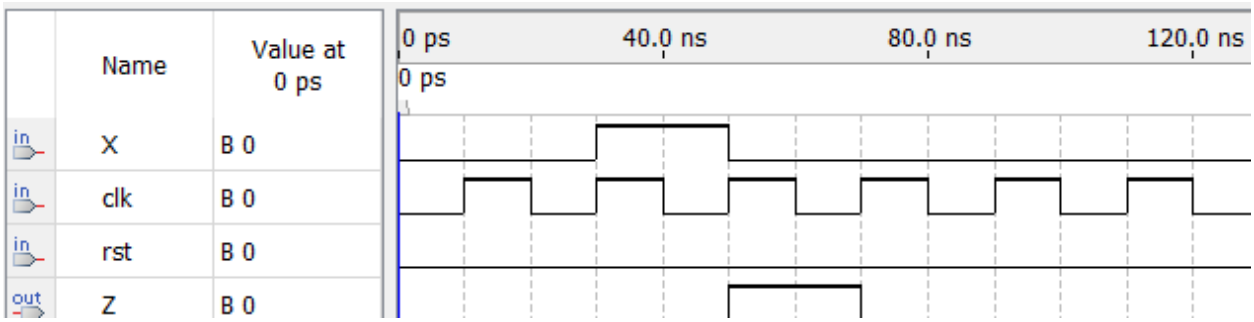
1. Sơ đồ chuyển trạng thái



2. Source Code:

```
module lab_3_moore(
    input wire clk,
    input wire rst,
    input wire X,
    output reg Z
);

    reg[1:0] cur_state, next_state;

    always @(posedge clk or posedge rst) begin
       if (rst)
          cur_state <= 2'b00;
       else
          cur_state <= next_state;
    end

    always @(*) begin
       case (cur_state)
          2'b00: next_state = (X == 1'b0) ? 2'b01 : 2'b00;
          2'b01: next_state = (X == 1'b1) ? 2'b10 : 2'b01;
          2'b10: next_state = (X == 1'b0) ? 2'b11 : 2'b00;
          2'b11: next_state = (X == 1'b0) ? 2'b01 : 2'b10;
          default: next_state = 2'b00;
       endcase
    end

    always @(*) begin
       Z = (cur_state == 2'b11) ? 1'b1 : 1'b0;
    end

endmodule

```

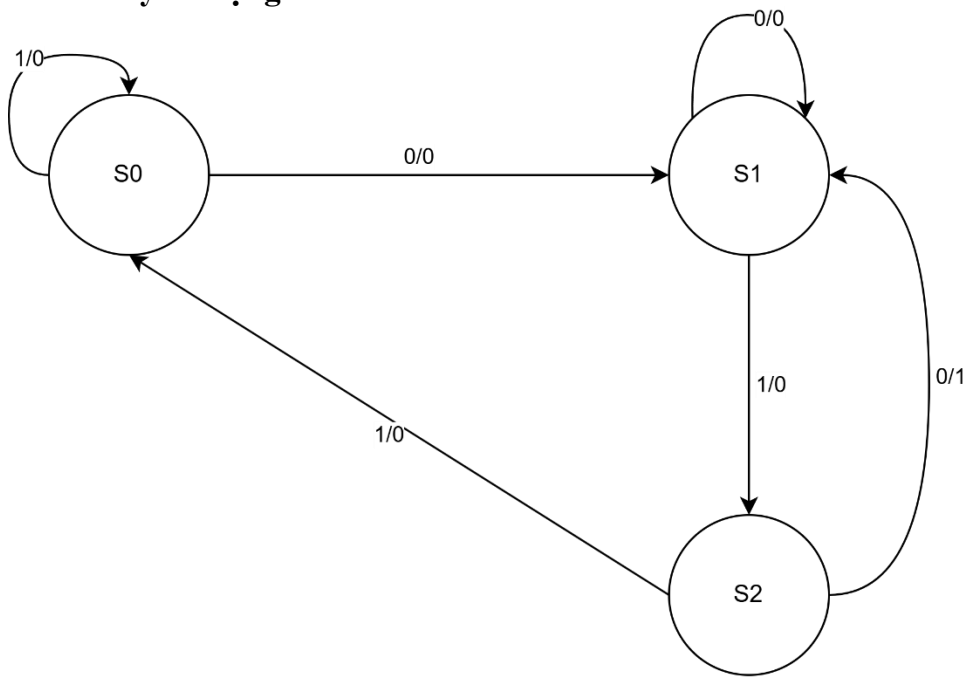
3. Waveform:



- Theo chu kì clock, input X được nhập lần lượt 010 và output Z đã lên 1

II. Thiết kế mạch tuần tự theo mô hình máy trạng thái kiểu mealy phát hiện chuỗi (010)

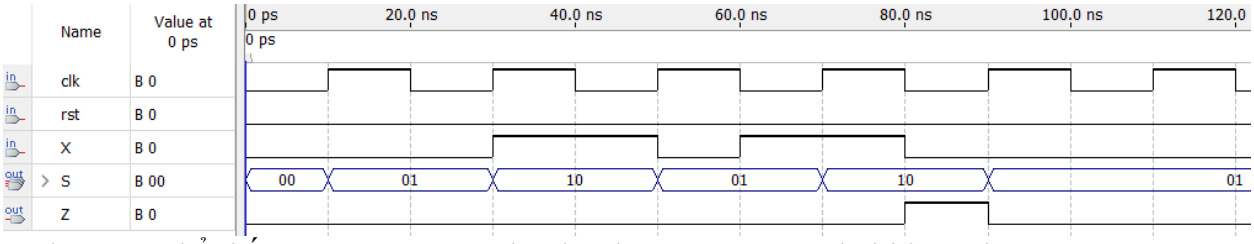
1. Sơ đồ chuyển trạng thái



2. Source Code

```
1  module lab_3_mealy (
2      input wire clk,
3      input wire rst,
4      input wire X,
5      output reg Z,
6      output reg [1:0] S
7  );
8
9      reg [1:0] cur_state, next_state;
10     always @(posedge clk or posedge rst) begin
11         if (rst)
12             cur_state <= 2'b00;
13         else
14             cur_state <= next_state;
15     end
16     always @(*) begin
17         case (cur_state)
18             2'b00: begin
19                 next_state = (X == 1'b0) ? 2'b01 : 2'b00;
20                 Z = 1'b0;
21                 S = cur_state;
22             end
23             2'b01: begin
24                 next_state = (X == 1'b1) ? 2'b10 : 2'b01;
25                 Z = 1'b0;
26                 S = cur_state;
27             end
28             2'b10: begin
29                 next_state = (X == 1'b0) ? 2'b01 : 2'b00;
30                 Z = (X == 1'b0) ? 1'b1 : 1'b0;
31                 S = cur_state;
32             end
33             default: begin
34                 next_state = 2'b00;
35                 Z = 1'b0;
36                 S = cur_state;
37             end
38         endcase
39     end
40
41 endmodule
42
```

3. Waveform



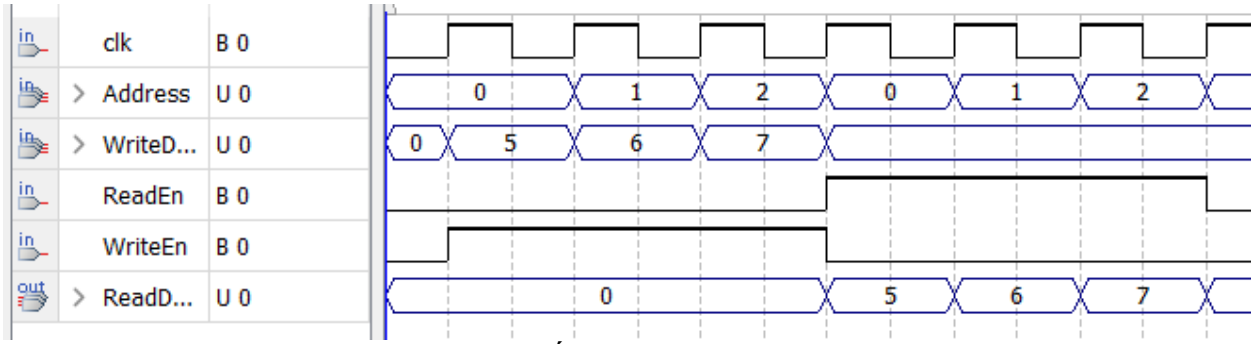
- Như ta có thể thấy, output của mealy phụ thuộc vào trạng thái hiện tại và ngõ vào nên chỉ khi ở state 2 và input = 1 thì Z mới = 1

III. Thiết kế SRAM

1. Source Code

```
1  module sram(
2      input wire clk,
3      input wire WriteEn,
4      input wire ReadEn,
5      input wire [5:0] Address,
6      input wire [31:0] WriteData,
7      output reg [31:0] ReadData
8  );
9
10     reg   [31:0]   MEMORY[0:63];
11
12     always@(posedge clk)
13     begin
14        if(WriteEn)
15           MEMORY[Address] <= WriteData;
16        else if(ReadEn)
17                 ReadData <= MEMORY[Address];
18              else
19                 ReadData <= 32'b0;
20     end
21
22 endmodule
```

2. Waveform



- Khi WriteEnable được bật lên thì sẽ tiến hày lưu data vào trong các địa chỉ được đặt
- Khi ReadEnable được bật lên thì sẽ tiến hành đọc data trong các địa chỉ được chọn