ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KỸ THUẬT MÁY TÍNH

BÁO CÁO BÀI THỰC HÀNH LAB3

HỌ VÀ TÊN: **NGUYỄN QUỐC TRƯỜNG AN – 21521810**

LỚP: **CE213.O11.2**

GIẢNG VIÊN HƯỚNG DẪN:

HỒ NGỌC DIỄM

TP. HỒ CHÍ MINH – Tháng 11 năm 2023

BÀI THỰC HÀNH SỐ 3

I. Mục tiêu

- Hiểu và tự thiết kế được máy trạng thái bất kì (FSM)
- Hiện thực thiết kế bằng ngôn ngữ mô tả phần cứng HDL (Verilog)
- Mô phỏng trên phần mềm Quartus, ModelSim và nạp trên Kit DE2

II. Chuẩn bị thực hành

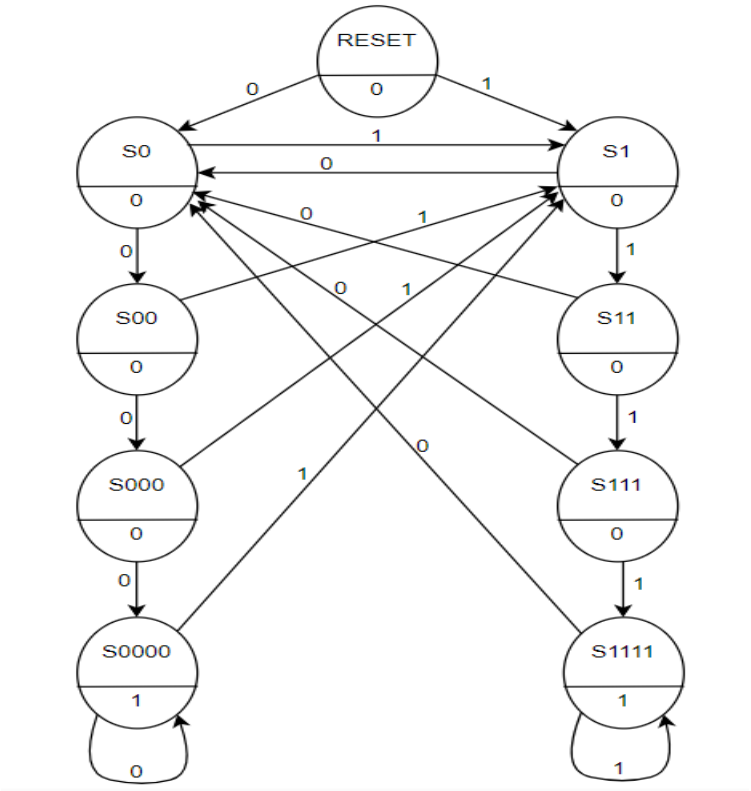
- Sinh viên phải chuẩn bị code Verilog cho tất cả các câu trong phần nội dung thực hành và nộp cho GVHD vào đầu buổi học.
- Sinh viên nào không có bài chuẩn bị được xem là vắng buổi học hôm đó.
- Bài chuẩn bị được tính vào điểm bài báo cáo của Lab.

III. Nội dung thực hành

Câu 1: Sử dụng Verilog HDL thiết kế một mạch tuần tự theo mô hình máy trạng thái có chức năng phát hiện hai chuỗi cụ thể của ngõ vào, cụ thể là bốn số 1 liên tiếp hoặc bốn số 0 liên tiếp. Mạch có một ngõ vào x và một ngõ ra z. Bất cứ khi nào x = 1 hoặc x = 0 trong bốn xung đồng hồ liên tiếp, giá trị của z = 1; mặt khác, z = 0. Cho phép chuỗi ngõ vào được chồng lấp nhau (overlapped), tức là nếu x = 1 trong năm xung clock liên tiếp thì ngõ ra z sẽ bằng 1 sau xung thứ tư và thứ năm.

Sử dụng công tắc SW0 trên bo Altera DE2 làm ngõ vào x, LEDG0 làm ngõ ra z và nút ấn KEY0 làm xung clock được áp dụng thủ công. Mô phỏng hoạt động của mạch và kiểm tra chức năng của mạch trên board DE2.

a) State diagram



Ý tưởng: Ta thực hiện máy trạng thái theo kiểu **Moore**, gồm **9 trạng thái** có ý nghĩa như sau:

- **RESET:** Hệ thống sẽ được reset về trạng thái RESET khi có tín hiệu rstn ở mức thấp, z = 0
- **S0:** Ngõ vào x = 0 lần 1, chuỗi liên tiếp lúc này là 0, z = 0
- **S00:** Ngõ vào x = 0 lần 2, chuỗi liên tiếp lúc này là 00, z = 0
- **S000:** Ngõ vào x = 0 lần 3, chuỗi liên tiếp lúc này là 000, z = 0
- **S0000:** Ngõ vào x = 0 lần 4, chuỗi liên tiếp lúc này là 0000, z = 1
- **S1:** Ngõ vào x = 1 lần 1, chuỗi liên tiếp lúc này là 1, z = 0
- **S11:** Ngõ vào x = 1 lần 2, chuỗi liên tiếp lúc này là 11, z = 0
- **S111:** Ngõ vào x = 1 lần 3, chuỗi liên tiếp lúc này là 111, z = 0
- **S1111:** Ngõ vào x = 1 lần 4, chuỗi liên tiếp lúc này là 1111, z = 1

b) Code

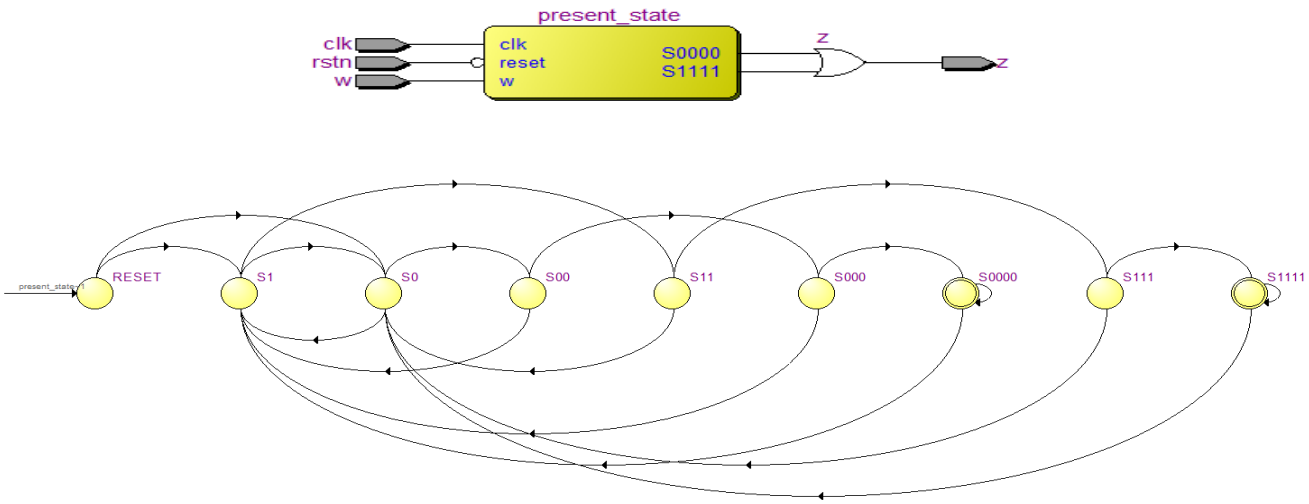
```
module moore_detector(z,
                      w,
                      rstn,
                      clk
                      );

    parameter [3:0] RESET = 0;
    parameter [3:0] S0 = 1;
    parameter [3:0] S00 = 2;
    parameter [3:0] S000 = 3;
    parameter [3:0] S0000 = 4;
    parameter [3:0] S1 = 5;
    parameter [3:0] S11 = 6;
    parameter [3:0] S111 = 7;
    parameter [3:0] S1111 = 8;

    output z;

    input w;
    input rstn;
    input clk;

    reg [3:0] present_state;
    reg [3:0] next_state;

    always @(*) begin

       case (present_state)

          RESET: next_state = w ? S1 : S0;

          S0: next_state = w ? S1 : S00;

          S00: next_state = w ? S1 : S000;

```

```
36          S000: next_state = w ? S1 : S0000;
37
38          S0000: next_state = w ? S1 : S0000;
39
40          S1: next_state = w ? S11 : S0;
41
42          S11: next_state = w ? S111 : S0;
43
44          S111: next_state = w ? S1111 : S0;
45
46          S1111: next_state = w ? S1111 : S0;
47
48       endcase
49
50    end
51
52    always @(posedge clk or negedge rstn) begin
53
54       if(!rstn)
55          present_state <= RESET;
56
57       else
58          present_state <= next_state;
59
60    end
61
62    assign z = ((present_state == S0000) || (present_state == S1111)) ? 1'b1 : 1'b0;
63
64 endmodule
```

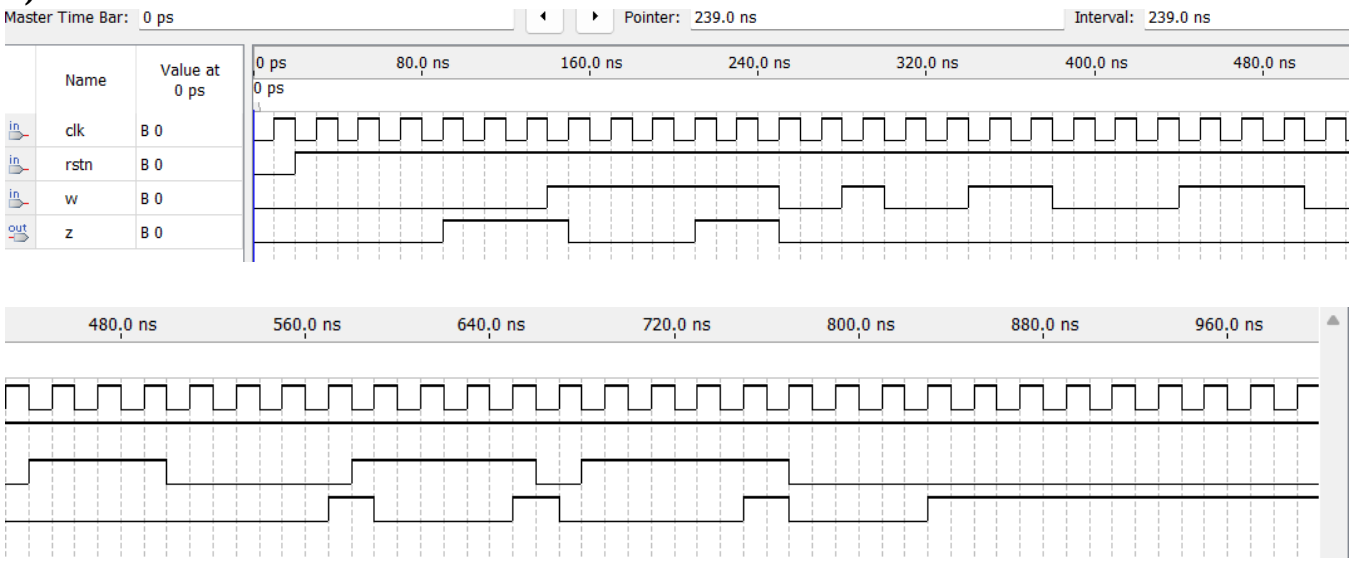
Chương trình code theo kiểu behavior được chia làm 3 phần như sau:

- Phần **Next State**: Từ dòng 26-50
- Phần **Present State**: Từ dòng 52-60
- Phần **Output**: Từ dòng 62

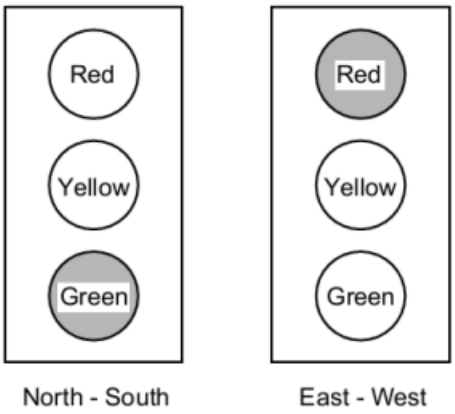
b) Netlist



b) Simulation



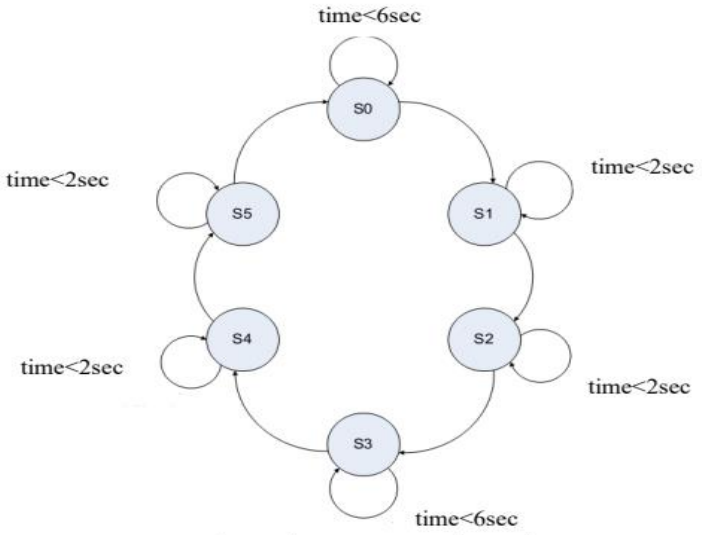
Câu 2: Hiện thực mạch báo đèn giao thông như mình hoạ trong Hình 3.1. Các đèn giao thông được đặt ở ngã tư giao nhau của một trục đường hướng bắc-nam và một trục đường hướng đông-tây.Tập các đèn giao thông được thể hiện trong Bảng 3.1 và giản đồ chuyển trạng thái cho các đèn trên hai trục Bắc-Nam và Đông-Tây được biểu diễn trong Hình 3.2.



Hình 3.1 – Sáu đèn Led thể hiện đèn giao thông trên hai trục đường

Bảng 3.1 – Các trạng thái đèn giao thông

State	North - South	East - West	Delay (sec.)
0	Green	Red	5
1	Yellow	Red	1
2	Red	Red	1
3	Red	Green	5
4	Red	Yellow	1
5	Red	Red	1



Hình 3.2 – Giản đồ chuyển trạng thái điều khiển đèn giao thông

Sử dụng đèn LED trên board DE2 để hiển thị các đèn Đỏ, Xanh, Vàng; CLOCK_50 để kiểm soát thời gian của mạch. Viết chương trình và kiểm tra chức năng của mạch trên board DE2.

a) Code

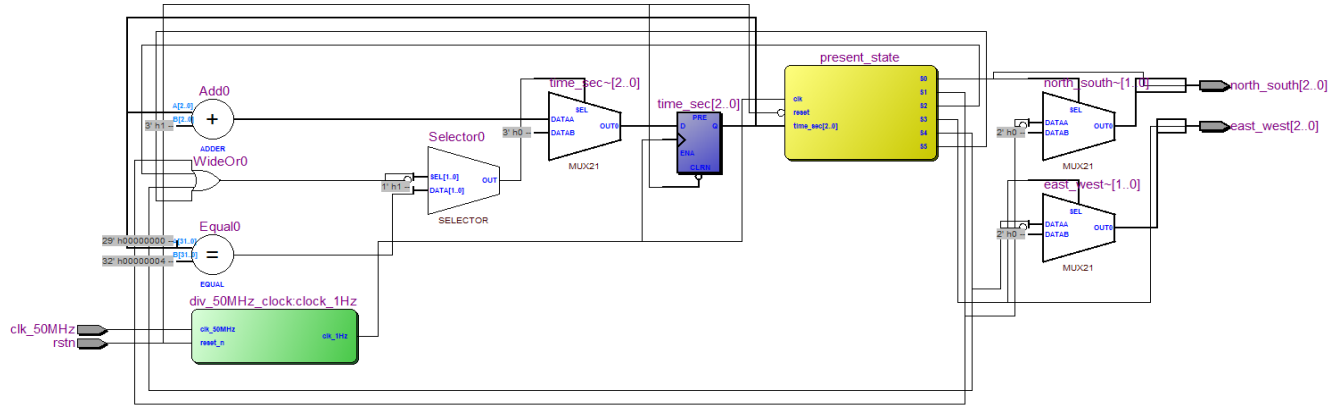
```
module div_50MHz_clock (clk_1Hz,
                        clk_50MHz,
                        reset_n
                        );

    output clk_1Hz;

    input clk_50MHz;
    input reset_n;

    reg clk_1Hz;

    reg [24:0] count;

    always @(posedge clk_50MHz or negedge reset_n) begin

       if(!reset_n) begin
          count <= 0;
          clk_1Hz <= 0;

       end

       else if(count < 25000000) begin

          count <= count + 1;

          end

       else begin

          count <= 0;
          clk_1Hz <= ~clk_1Hz;

       end
```

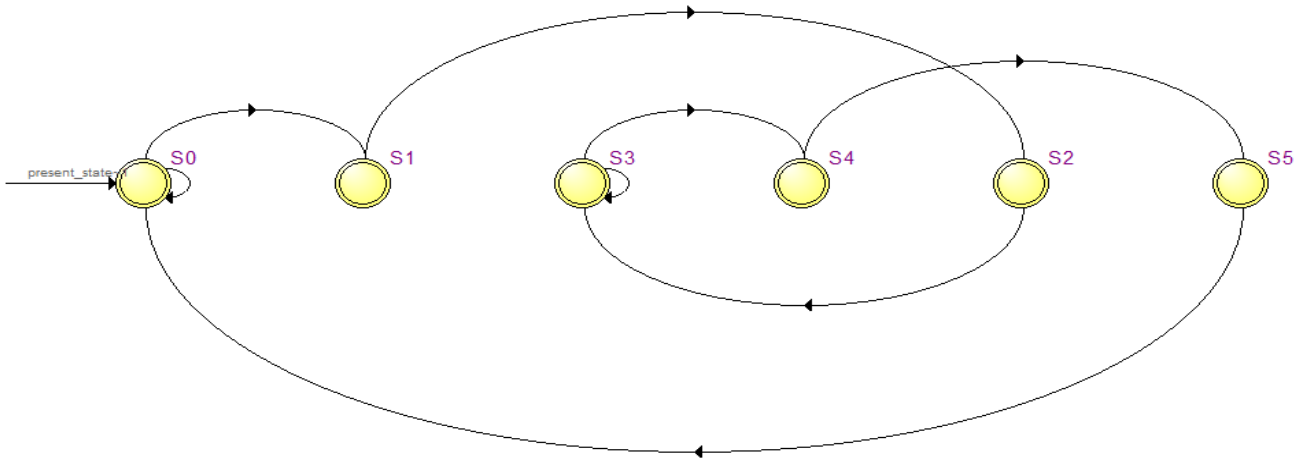
```
35
36      end
37
38  endmodule
39
40
41  module traffic_light (north_south,
42                        east_west,
43                        rstn,
44                        clk_50MHz
45                        );
46
47      parameter S0 = 0;
48      parameter S1 = 1;
49      parameter S2 = 2;
50      parameter S3 = 3;
51      parameter S4 = 4;
52      parameter S5 = 5;
53
54      output [2:0] north_south;
55      output [2:0] east_west;
56
57      input rstn;
58      input clk_50MHz;
59
60      reg [2:0] present_state;
61      reg [2:0] next_state;
62      reg [2:0] time_sec;
63      reg change;
64
65      wire clk_1Hz;
66
67      div_50MHz_clock clock_1Hz(.clk_1Hz(clk_1Hz),
68                                .clk_50MHz(clk_50MHz),
69                                .reset_n(rstn)
70                                );
71
72      always @(*) begin
73
74          case (present_state)
75
76              S0: begin
77                    change = (time_sec == 4) ? 1 : 0;
78                    next_state = (time_sec == 4) ? S1 : S0;
79                  end
80
81              S1: begin
82                    change = 1;
83                    next_state = S2;
84                  end
85
86              S2: begin
87                    change = 1;
88                    next_state = S3;
89                  end
90
91              S3: begin
92                    change = (time_sec == 4) ? 1 : 0;
93                    next_state = (time_sec == 4) ? S4 : S3;
94                  end
95
96              S4: begin
97                    change = 1;
98                    next_state = S5;
99                  end
100
```

```
101         S5: begin
102               change = 1;
103               next_state = S0;
104            end
105
106      endcase
107
108   end
109
110   always @(posedge clk_1Hz or negedge rstn) begin
111
112      if(!rstn) begin
113         present_state <= S0;
114         time_sec <= 0;
115      end
116
117      else begin
118
119         present_state <= next_state;
120
121         if(change)
122            time_sec <= 0;
123
124         else
125            time_sec <= time_sec + 1;
126
127      end
128
129   end
130
131   assign north_south = (present_state == S0) ? 3'b010 :
132                        (present_state == S1) ? 3'b001 : 3'b100;
133
134   assign east_west = (present_state == S3) ? 3'b010 :
135                      (present_state == S4) ? 3'b001 : 3'b100;
136
137endmodule
```

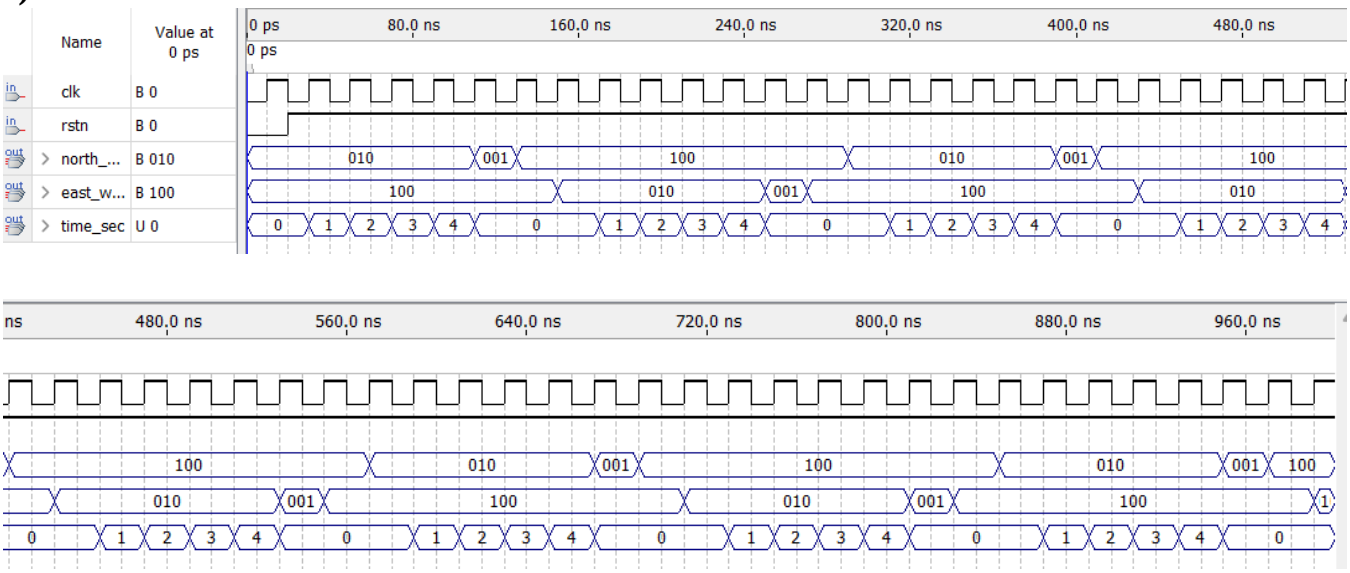
Ý tưởng: Ta thực hiện code theo kiểu behavior, để tạo ra xung clock có tần số 1Hz từ xung clock có tần số 50MHz, ta thực hiện qua bộ chia tần số. Ta dùng biến đếm từ 0-24.999.999, ứng với 25.000.000 thì ta thực hiện đảo clk_1Hz 1 lần sẽ tạo ra được clock có tần số 1Hz. Tiếp đến, để thực hiện delay 1s và 5s, ta lần lượt cho biến đếm qua 1 xung và 5 xung clock thì mới chuyển trạng thái, từ đó có được thời gian delay giữa các trạng thái mong muốn.

b) Netlist



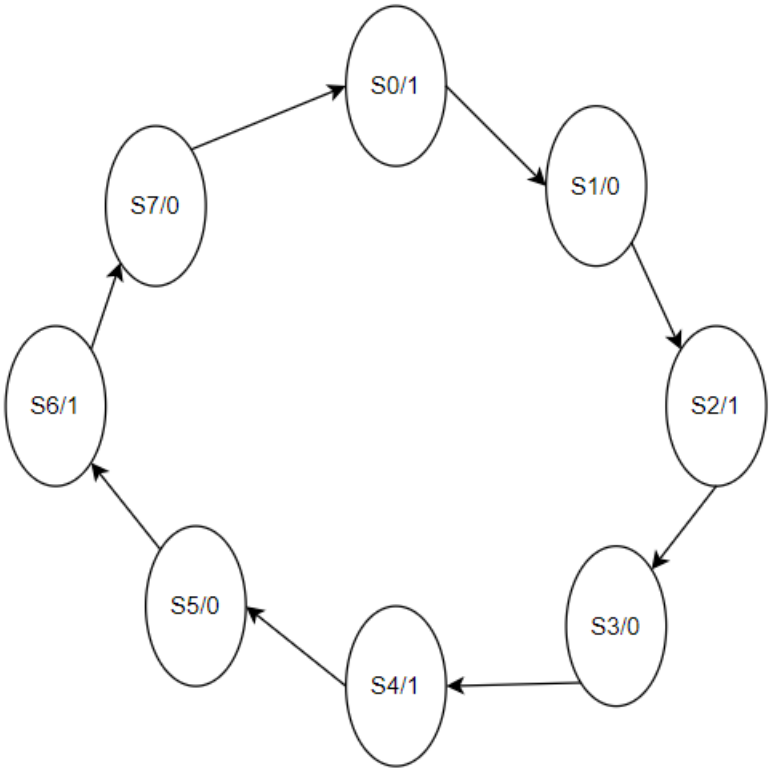


c) Simulation



Câu 3:

a) State diagram



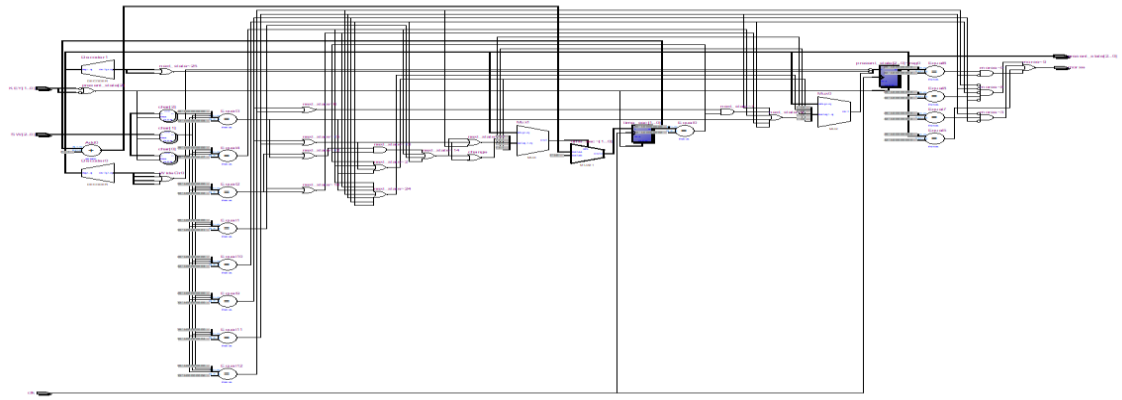
- Các trạng thái S0, S2, S4, S6 ứng với 4 trạng thái của ký tự thuộc 2 kiểu (xung ngắn hoặc xung dài)
- Các trạng thái còn lại ứng với khoảng nghỉ giữa 2 xung (xung ngắn hoặc xung dài)

b) Code

```
module morse_encoder (morse,
                      clk,
                      SW,
                      KEY,
                      present_state
                      );

    parameter S0 = 0;
    parameter S1 = 1;
    parameter S2 = 2;
    parameter S3 = 3;
    parameter S4 = 4;
    parameter S5 = 5;
    parameter S6 = 6;
    parameter S7 = 7;
    parameter A = 0;
    parameter B = 1;
    parameter C = 2;
    parameter D = 3;
    parameter E = 4;
    parameter F = 5;
    parameter G = 6;
    parameter H = 7;
    parameter LONG = 3;
    parameter SHORT = 1;
    parameter SNAP = 1;

    output morse;
    output [2:0] present_state;

    input clk;
    input [2:0] SW;
    input [1:0] KEY;

reg [2:0] present_state;
reg [2:0] next_state;
reg [2:0] char;
reg [1:0] time_sec;
reg change;

always @(*) begin

   case(present_state)

      S0: begin
            next_state = ((char == A || char == E || char == F || char == H) || (time_sec == 2)) ? S1 : S0;
            change = ((char == A || char == E || char == F || char == H) || (time_sec == 2)) ? 1 : 0;
         end

      S1: begin
            next_state = S2;
            change = 1;
         end

      S2: begin
            next_state = ((char == B || char == C || char == D || char == F || char == H)
                        || ((char == A  || char == G) && time_sec == 2)
                        || (char == E)) ? S3 : S2;
            change = ((char == B || char == C || char == D || char == F || char == H)
                     || ((char == A || char == G) && time_sec == 2)
                     || (char == E)) ? 1 : 0;
         end

      S3: begin
            next_state = S4;
            change = 1;
         end
```

```
69          S4: begin
70                next_state = ((char == B || char == D || char == H)
71                            || ((char == C || char == F) && time_sec == 2)
72                            || (char == E || char == A || char == G)) ? S5 : S4;
73                change = ((char == B || char == D || char == H)
74                         || ((char == C || char == F) && time_sec == 2)
75                         || (char == E || char == A || char == G)) ? 1 : 0;
76             end
77
78          S5: begin
79                next_state = S6;
80                change = 1;
81             end
82
83          S6: begin
84                next_state = ((char == B || char == C || char == F || char == H) || (char == A || char == D || char == E || char == G)) ? S7 : S6;
85                change = ((char == B || char == D || char == H)
86                         || ((char == C || char == F) && time_sec == 2)
87                         || (char == E || char == A)) ? 1 : 0;
88             end
89
90          S7: begin
91                next_state = S0;
92                change = 1;
93             end
94
95          endcase
96
97       end
98
99       always @(posedge clk or negedge KEY[0] or negedge KEY[1]) begin
100
101         if(!KEY[0]) begin
102
103            time_sec <= 0;
104            present_state <= S0;
105            char <= SW;
106
107         end
108         else if(!KEY[1]) begin
109
110            char <= SW;
111            time_sec <= 0;
112            present_state <= S0;
113
114         end
115         else begin
116
117            present_state <= next_state;
118
119            if (change)
120               time_sec <= 0;
121
122            else
123               time_sec <= time_sec + 1;
124
125         end
126
127      end
128
129      assign morse = (present_state == S0 || present_state == S2 && char != E || present_state == S4 && char != A && char != E || present_state == S
    S6 && char != A && char != D && char != E && char != G) ? 1 : 0;
130
131   endmodule
```

c) Netlist



d) Simulation