

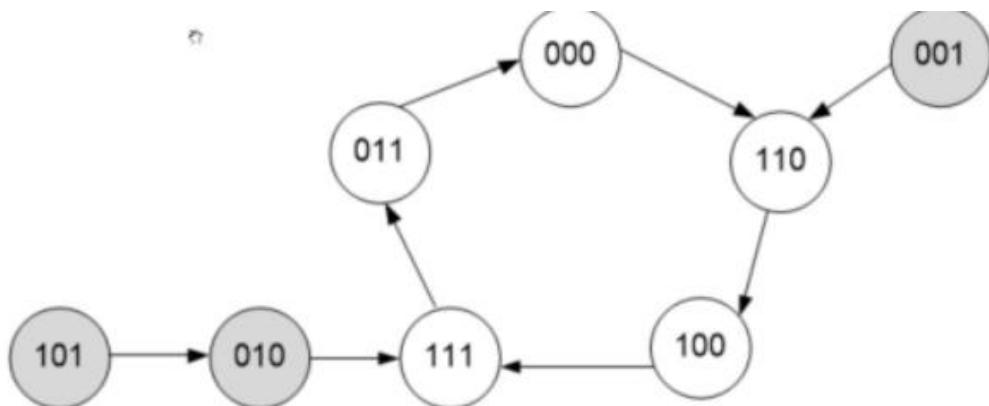
Họ và tên thành viên 1: **Nguyễn Việt Dũng**

MSSV thành viên 1: **23520338**

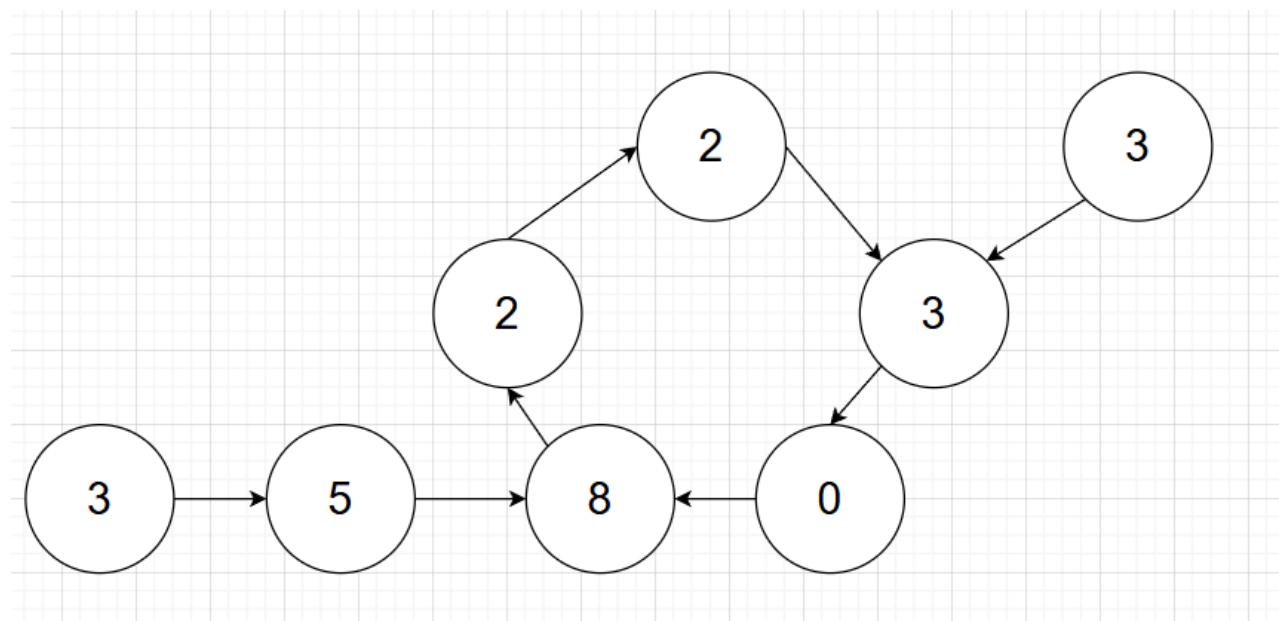
Lớp: **CE213.Q12**

LAB 1

Thiết kế một mạch đếm đồng bộ có chu trình đếm như sơ đồ mã hóa theo mã số sinh viên (23520338)



Hình 2-1 Sơ đồ chuyển trạng thái của bộ đếm



I. Các Module của mạch :

- lab1 (module chính xử lý mạch đếm)

1. Khai báo biến

```
1 module lab1 (in, cnt, load_en, clk, rst, out);
2     input      [3:0] in;
3     input      [1:0] cnt;
4     input          load_en, clk, rst;
5     output reg [3:0] out;
6
7     reg      [2:0] state, n_state;
```

- in : 1 số trong mã số sinh viên (ví dụ 2, 3, 5,)
- cnt : vị trí của số trong mã số nếu có trùng
- load_en : tín hiệu cho phép nhập
- clk : Clock chạy của mạch
- rst : reset mức thấp
- out : 1 số trong mã số sinh viên
- state: trạng thái hiện tại
- n_state: trạng thái kế tiếp

2. Định nghĩa các trạng thái

```
8
9     parameter state0 = 3'b000;
10    parameter state1 = 3'b001;
11    parameter state2 = 3'b010;
12    parameter state3 = 3'b011;
13    parameter state4 = 3'b100;
14    parameter state5 = 3'b101;
15    parameter state6 = 3'b110;
16    parameter state7 = 3'b111;
17
```

3. Định nghĩa các số trong mã số sinh viên

```
18    parameter mssv0  = 4'b0010;      // 2
19    parameter mssv1  = 4'b0011;      // 3
20    parameter mssv2  = 4'b0101;      // 5
21    parameter mssv3  = 4'b0010;      // 2
22    parameter mssv4  = 4'b0000;      // 0
23    parameter mssv5  = 4'b0011;      // 3
24    parameter mssv6  = 4'b0011;      // 3
25    parameter mssv7  = 4'b1000;      // 8
26
```

4. Đoạn mã này thực hiện việc chuyển đổi trạng thái của hệ thống

```
27  always @(posedge clk or negedge rst) begin
28      if (!rst)
29          state <= state0;
30      else
31          state <= n_state;
32  end
33 
```

5. Đoạn mã chuyển các trạng thái theo chu trình và theo input nhập đồng bộ

```
34  always @(*) begin
35      if (load_en)
36          case (in)
37              4'd2: begin
38                  if (cnt == 2'd0)
39                      n_state = state0;
40                  else
41                      n_state = state3;
42              end
43              4'd3: begin
44                  if (cnt == 2'd0)
45                      n_state = state1;
46                  else if (cnt == 2'd1)
47                      n_state = state5;
48                  else
49                      n_state = state6;
50              end
51              4'd5: n_state = state2;
52              4'd0: n_state = state4;
53              4'd8: n_state = state7;
54              default: n_state = state0;
55          endcase

```

```

56      |     else
57      |     |     case (state)
58      |     |     state0: n_state = state6;
59      |     |     state1: n_state = state6;
60      |     |     state2: n_state = state7;
61      |     |     state3: n_state = state0;
62      |     |     state4: n_state = state7;
63      |     |     state5: n_state = state2;
64      |     |     state6: n_state = state4;
65      |     |     state7: n_state = state3;
66      |     |     default: n_state = state0;
67      |     |     endcase
68      |     end

```

6. Đoạn mã giải mã từ state sang số trong mã số sinh viên

```

70      |     always @(*) begin
71      |     |     case(state)
72      |     |     state0: out = 4'd2;
73      |     |     state1: out = 4'd3;
74      |     |     state2: out = 4'd5;
75      |     |     state3: out = 4'd2;
76      |     |     state4: out = 4'd0;
77      |     |     state5: out = 4'd3;
78      |     |     state6: out = 4'd3;
79      |     |     state7: out = 4'd8;
80      |     |     default: out = 4'd2;
81      |     |     endcase
82      |     end
83      |     endmodule

```

- clk_div_1hz (module chia tần số 50Mhz về 1Hz)

THIẾT KẾ HỆ THỐNG SỐ VỚI HDL

```
1 module clk_div_1hz (clk_50m, rst_n, clk_lhz);
2     input wire clk_50m;
3     input wire rst_n;
4     output reg clk_lhz;
5
6     reg [25:0] cnt;
7
8     always @(posedge clk_50m or negedge rst_n) begin
9         if (!rst_n)
10             {cnt, clk_lhz} <= 27'd0;
11         else if (cnt == 26'd24_999_999)
12             {cnt, clk_lhz} <= {26'd0, ~clk_lhz};
13         else
14             cnt <= cnt + 1'b1;
15     end
16
17 endmodule
```

- **seg7_decoder** (module chuyển số hex sang các giá trị của led 7 đoạn)

```
1 module seg7_decoder (bin ,seg);
2     input [3:0] bin;
3     output reg [6:0] seg; // {g,f,e,d,c,b,a}
4
5     always @(*) begin
6         case (bin)
7             4'd0: seg = 7'b1000000;
8             4'd1: seg = 7'b1111001;
9             4'd2: seg = 7'b0100100;
10            4'd3: seg = 7'b0110000;
11            4'd4: seg = 7'b0011001;
12            4'd5: seg = 7'b0010010;
13            4'd6: seg = 7'b0000010;
14            4'd7: seg = 7'b1111000;
15            4'd8: seg = 7'b0000000;
16            4'd9: seg = 7'b0010000;
17            default: seg = 7'bl111111;
18        endcase
19    end
20 endmodule
```

- **final** (module kết nối 3 module lại)

THIẾT KẾ HỆ THỐNG SỐ VỚI HDL

```
1  module final (clk_50m, rst, load_en, in, cnt, hex0);
2      input          clk_50m;
3      input          rst;
4      input          load_en;
5      input [3:0]    in;
6      input [1:0]    cnt;
7      output [6:0]   hex0;
8
9      wire clk_lhz;
10     wire [3:0] out;
11
12    // Clock divider
13    clk_div_lhz u_clk (
14        .clk_50m (clk_50m),
15        .rst_n   (rst),
16        .clk_lhz (clk_lhz)
17    );
18
19    // FSM
20    fsm u_fsm (
21        .in      (in),
22        .cnt    (cnt),
23        .load_en (load_en),
24        .clk     (clk_lhz),
25        .rst     (rst),
26        .out     (out)
27    );
28
29    // 7-seg
30    seg7_decoder u_hex (
31        .bin (out),
32        .seg (hex0)
33    );
34 endmodule
```

- **top_module** (module để gán chân cho board de2kit)

THIẾT KẾ HỆ THỐNG SỐ VỚI HDL

```
1  module top_module ( CLOCK_50, KEY, SW, HEX0, LEDR);
2      input      CLOCK_50;
3      input [3:0] KEY;
4      input [17:0] SW;
5      output [6:0] HEX0;
6      output [17:0] LEDR;
7
8      wire clk_50m;
9      wire rst;
10     wire load_en;
11     wire [3:0] in;
12     wire [1:0] cnt;
13     wire [6:0] hex0;
14
15     assign clk_50m = CLOCK_50;
16     assign rst = KEY[0];
17     assign load_en = SW[6];
18     assign in = SW[3:0];
19     assign cnt = SW[5:4];
20     assign HEX0[6:0] = hex0;
21
22     final u_final (
23         .clk_50m (clk_50m),
24         .rst      (rst),
25         .load_en (load_en),
26         .in       (in),
27         .cnt      (cnt),
28         .hex0    (hex0)
29     );
30
31 endmodule
```

II. Waveform chạy mô phỏng trên ModelSim (Pre Simulation)

- Testbench

1. Khai báo tín hiệu

```
1  `timescale 1ns/1ps
2
3  module tb_lab1;
4
5      reg [3:0] in;
6      reg [1:0] cnt;
7      reg      load_en;
8      reg      clk;
9      reg      rst;
10     wire [3:0] out;
```

2. Khởi tạo instance modul

THIẾT KẾ HỆ THỐNG SỐ VỚI HDL

```
12 //=====
13 // DUT
14 //=====
15   labl dut (
16     .in(in),
17     .cnt(cnt),
18     .load_en(load_en),
19     .clk(clk),
20     .rst(rst),
21     .out(out)
22   );
23 
```

3. Khởi tạo xung clock (10ns)

```
24 //=====
25 // CLOCK 10ns
26 //=====
27   always #5 clk = ~clk;
```

4. Task load_and_step để load giá trị đầu vào

```
29 //=====
30 // TASK: LOAD + STEP
31 //=====
32   task load_and_step;
33     input [3:0] t_in;
34     input [2:0] t_cnt;
35     begin
36       load_en = 1;
37       in = t_in;
38       cnt = t_cnt;
39       #10;           // 1 clock
40       load_en = 0;
41     end
42   endtask
43 
```

5. Test các trường hợp

THIẾT KẾ HỆ THỐNG SỐ VỚI HDL

```
44      //=====
45      // TEST SEQUENCE
46      //=====
47      initial begin
48          // INIT
49          clk      = 0;
50          rst      = 0;
51          load_en = 0;
52          in       = 0;
53          cnt      = 0;
54
55          //-----
56          // RESET
57          //-----
58          #10 rst = 1;
59
60          //-----
61          // TEST SỐ 2
62          //-----
63          load_and_step(4'd2, 3'd0);    // 2 lần 1
64          #20;
65          load_and_step(4'd2, 3'd1);    // 2 lần 2
66          #20;
67
68          //-----
69          // TEST SỐ 3
70          //-----
71          load_and_step(4'd3, 3'd0);    // 3 lần 1
72          #20;
73          load_and_step(4'd3, 3'd1);    // 3 lần 2
74          #20;
75          load_and_step(4'd3, 3'd2);    // 3 lần 3
76          #20;
77
78          //-----
79          // TEST SỐ 5
80          //-----
81          load_and_step(4'd5, 3'd0);
82          #20;
```

THIẾT KẾ HỆ THỐNG SỐ VỚI HDL

```
83 //-----  
84 // TEST SỐ 0  
85 //-----  
86 load_and_step(4'd0, 3'd0);  
87 #20;  
88  
89 //-----  
90 // TEST SỐ 8  
91 //-----  
92 load_and_step(4'd8, 3'd0);  
93 #20;  
94  
95 //-----  
96 // TEST SỐ 4  
97 //-----  
98 load_and_step(4'd4, 3'd0);  
99 #20;  
100 load_and_step(4'd4, 3'd1);  
101 #20;  
102  
103 //-----  
104 // TEST SỐ 5 nếu cnt = 1  
105 //-----  
106 load_and_step(4'd5, 3'd1);  
107 #20;  
108  
109 //-----  
110 // CHẠY FSM BÌNH THƯỜNG  
111 //-----  
112 #10;  
113 repeat (8) #10;  
114  
115 //-----  
116 // KẾT THÚC  
117 //-----  
118 $display("== END OF TEST ==");  
119 #20;  
120 $finish;  
121  
122 end
```

6. Hiện thị ra màn hình

```
124 //=====  
125 // MONITOR  
126 //=====  
127 initial begin  
128     $display(" time | rst load in cnt | state | out");  
129     $monitor("%4t | %b %b %d %d | %d %d",  
130             $time, rst, load_en, in, cnt, dut.state, out);  
131 end
```

- Waveform

THIẾT KẾ HỆ THỐNG SỐ VỚI HDL



III. Source code

Đường dẫn: [Link github](#)

IV. Video mô phỏng

Đường dẫn: [Link mô phỏng](#)