



**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN  
KHOA KỸ THUẬT MÁY TÍNH**



**UIT  
TRƯỜNG ĐẠI HỌC  
CÔNG NGHỆ THÔNG TIN**

**Sinh viên:** Trương Thiên Quý

**MSSV:** 23521321

**Lớp:** CE213.P21

**Giảng viên hướng dẫn:** Hồ Ngọc Diễm



## BÀI THỰC HÀNH 3

### THIẾT KẾ MÁY TRẠNG THÁI HỮU HẠN

#### I. Mục tiêu

- Trong bài thực hành này, sinh viên sẽ dùng sử dụng ngôn ngữ Verilog HDL thiết kế mạch tuần tự theo mô hình máy trạng thái hữu hạn (FSM).

#### 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.
- Điểm 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.

**TRƯỜNG ĐẠI HỌC  
CÔNG NGHỆ THÔNG TIN**

**Code thực thi:**

```
1  module sequence_detector (
2      input clk,
3      input x,
4      output reg z
5  );
6      // Định nghĩa các trạng thái dùng parameter
7      parameter S0 = 4'd0,
8          S1 = 4'd1,
9          S2 = 4'd2,
10         S3 = 4'd3,
11         S4 = 4'd4,
12         S5 = 4'd5,
13         S6 = 4'd6,
14         S7 = 4'd7,
15         S8 = 4'd8;
16
17     reg [3:0] current_state, next_state;
18
19     always @ (posedge clk) begin
20         current_state <= next_state;
21     end
22
23     always @ (*) begin
24         z = 0; // mặc định
25         case (current_state)
26             S0: next_state = (x) ? S1 : S5;
27             S1: next_state = (x) ? S2 : S5;
28             S2: next_state = (x) ? S3 : S5;
29             S3: next_state = (x) ? S4 : S5;
30             S4: begin
31                 next_state = (x) ? S4 : S5;
32                 z = 1;
33             end

```

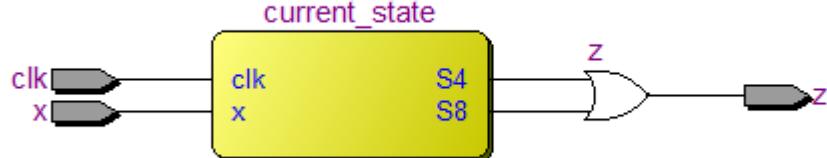


```

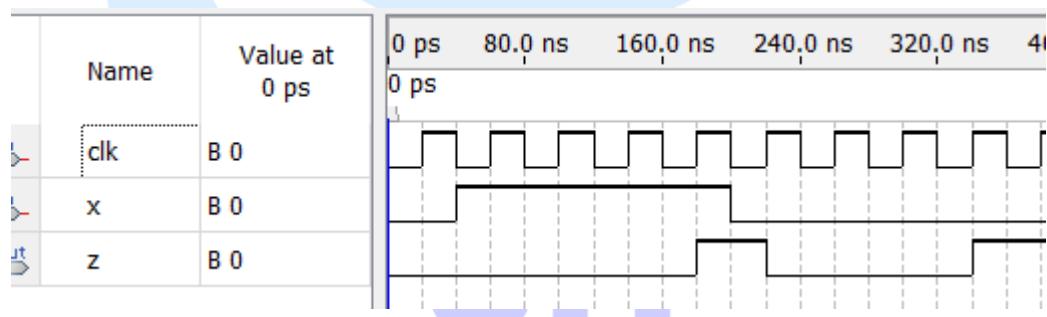
34      S5: next_state = (~x) ? S6 : S1;
35      S6: next_state = (~x) ? S7 : S1;
36      S7: next_state = (~x) ? S8 : S1;
37      S8: begin
38          next_state = (~x) ? S8 : S1;
39          z = 1;
40      end
41      default: next_state = S0;
42  endcase
43 end
44 endmodule

```

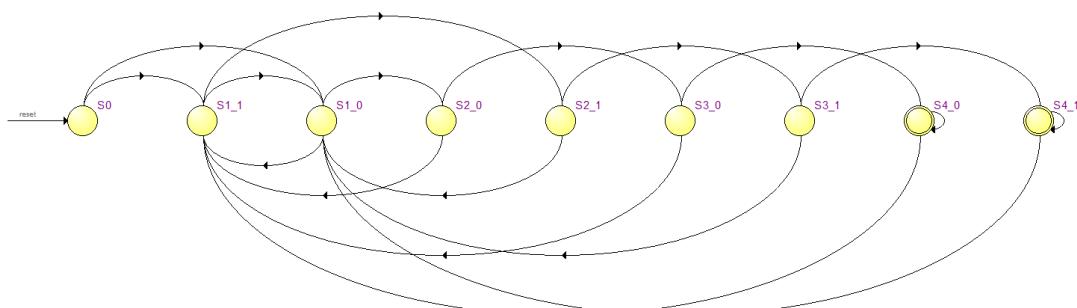
### RTL Viewer:



### Mô phỏng waveform:



### Sơ đồ chuyển trạng thái: TRƯỜNG ĐẠI HỌC





### Giải thích:

1. **Mỗi lần nhấn nút KEY0 (clock),** mạch sẽ:

- **Lấy giá trị hiện tại của SW0** (data đầu vào),
- **Dịch giá trị này vào thanh ghi 4 bit** (shift\_register), giữ lại 4 giá trị gần nhất.

2. **Thanh ghi luôn chứa 4 bit gần nhất:**

- Ví dụ: nếu bạn nhập lần lượt các giá trị 1 1 1 1 thì thanh ghi sẽ chứa 1111.

3. **Sau mỗi lần nhấn,** mạch sẽ kiểm tra:

- Nếu 4 bit hiện tại là 1111 **hoặc** 0000 → mạch phát hiện 4 bit giống nhau →  $z = 1$ .
- Ngược lại →  $z = 0$ .

4. **Mạch cho phép chuỗi chồng lấp (overlap):**

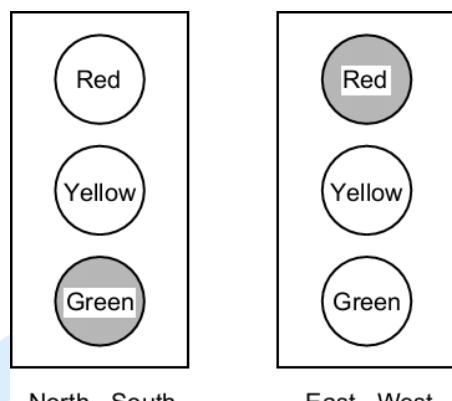
- Ví dụ nếu bạn nhập 1 1 1 1 1, thì  $z = 1$  ở lần thứ 4 và thứ 5.

Mỗi lần nhấn clock, mạch sẽ kiểm tra 4 bit gần nhất từ SW0. Nếu chúng **đều là 0 hoặc đều là 1**, thì mạch sẽ bật LEDG0 để báo hiệu ( $z = 1$ ). Ngược lại thì tắt ( $z = 0$ ).

### Câu 2:

Hiện thực mạch báo đèn giao thông như minh họa 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.

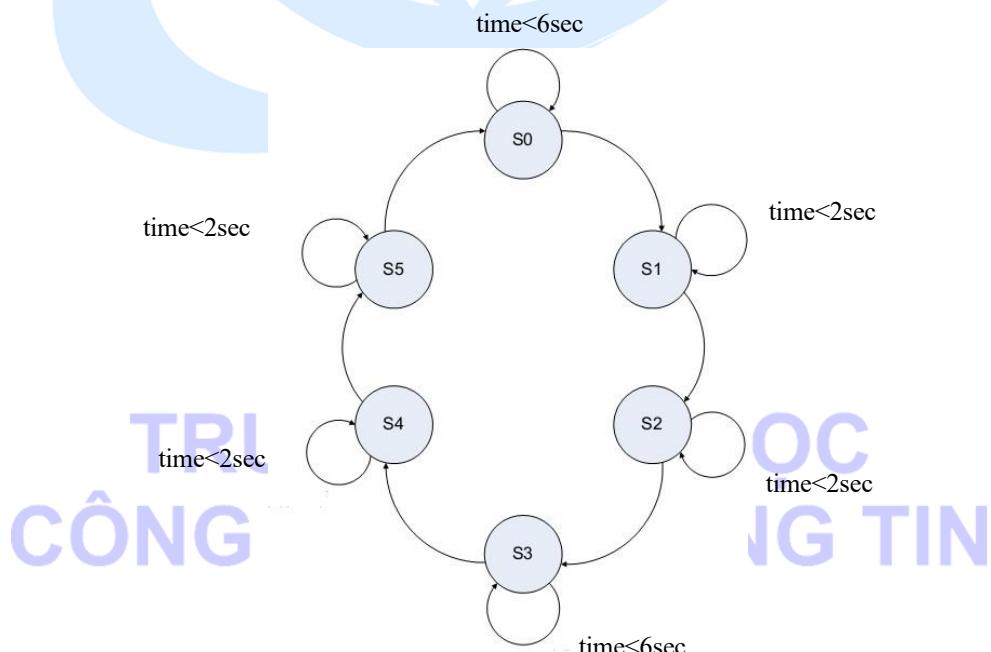
**UIT**  
**TRƯỜNG ĐẠI HỌC**  
**CÔNG NGHỆ THÔNG TIN**



**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.

### Code thực thi:

```

1  module TrafficLightController(
2      input clk_50MHz, reset,
3      output reg [2:0] LEDR,          // LED đỏ trên DE2
4      output reg [2:0] LEDG          // LED xanh, vàng trên DE2
5  );
6      reg [2:0] NS_Light;           // Đèn hướng Bắc - Nam (Red, Yellow, Green)
7      reg [2:0] EW_Light;           // Đèn hướng Đông - Tây (Red, Yellow, Green)
8      // State encoding
9      parameter S0 = 3'b000;        // NS: Green, EW: Red (6s)
10     parameter S1 = 3'b001;        // NS: Yellow, EW: Red (2s)
11     parameter S2 = 3'b010;        // NS: Red, EW: Red (2s)
12     parameter S3 = 3'b011;        // NS: Red, EW: Green (6s)
13     parameter S4 = 3'b100;        // NS: Red, EW: Yellow (2s)
14     parameter S5 = 3'b101;        // NS: Red, EW: Red (2s)
15
16     reg [2:0] state, next_state;
17     reg [25:0] timer;
18
19     // Next state logic
20     always @(*) begin
21         next_state = state; // Default: stay in current state
22
23         case (state)
24             S0: if (timer > 250_000_000) next_state = S1;
25             S1: if (timer > 50_000_000) next_state = S2;
26             S2: if (timer > 50_000_000) next_state = S3;
27             S3: if (timer > 250_000_000) next_state = S4;
28             S4: if (timer > 50_000_000) next_state = S5;
29             S5: if (timer > 50_000_000) next_state = S0;
30             default: next_state = S0;
31         endcase
32     end
33 
```

**CÔNG NGHỆ THÔNG TIN**



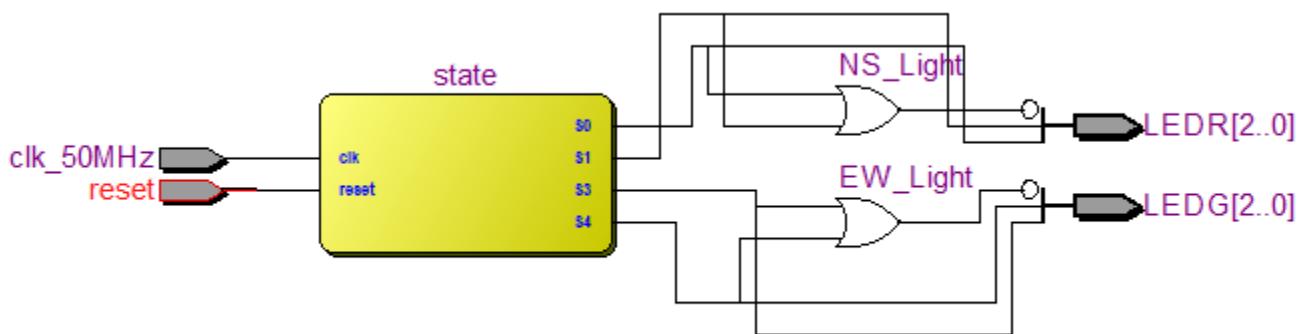
```

33
34     // State and timer sequential logic
35     always @(posedge clk_50MHz or posedge reset) begin
36         if (reset) begin
37             timer <= 0;
38             state <= S0;
39         end else begin
40             if (state != next_state) begin
41                 // Reset timer when state changes
42                 timer <= 0;
43                 state <= next_state;
44             end else begin
45                 // Increment timer when staying in the same state
46                 timer <= timer + 1;
47             end
48         end
49     end
50
51     // Light output logic
52     always @(*) begin
53         case (state)
54             S0: begin NS_Light = 3'b001; EW_Light = 3'b100; end
55             S1: begin NS_Light = 3'b010; EW_Light = 3'b100; end
56             S2: begin NS_Light = 3'b100; EW_Light = 3'b100; end
57             S3: begin NS_Light = 3'b100; EW_Light = 3'b001; end
58             S4: begin NS_Light = 3'b100; EW_Light = 3'b010; end
59             S5: begin NS_Light = 3'b100; EW_Light = 3'b100; end
60             default: begin NS_Light = 3'b100; EW_Light = 3'b100; end
61         endcase
62     end
63
64     // LED mapping logic
65     always @(*) begin
66         LEDR = NS_Light;
67         LEDG = EW_Light;
68     end
69 endmodule

```

**RTL Viewer:**

# TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN



### Code thực thi nếu không có chân Clk 50:

```

1  module TrafficLightControllerWithoutCLK50(
2      input clk, reset,
3      output reg [2:0] LEDR,          // LED đỏ trên DE2
4      output reg [2:0] LEDG          // LED xanh, vàng trên DE2
5  );
6      reg [2:0] NS_Light;           // Đèn hướng Bắc - Nam (Red, Yellow, Green)
7      reg [2:0] EW_Light;           // Đèn hướng Đông - Tây (Red, Yellow, Green)
8      // State encoding
9      parameter S0 = 3'b000;        // NS: Green, EW: Red (6s)
10     parameter S1 = 3'b001;        // NS: Yellow, EW: Red (2s)
11     parameter S2 = 3'b010;        // NS: Red, EW: Red (2s)
12     parameter S3 = 3'b011;        // NS: Red, EW: Green (6s)
13     parameter S4 = 3'b100;        // NS: Red, EW: Yellow (2s)
14     parameter S5 = 3'b101;        // NS: Red, EW: Red (2s)
15
16     reg [2:0] state, next_state;
17     reg [2:0] timer;
18
19     // Next state logic
20     always @(*) begin
21         next_state = state; // Default: stay in current state
22
23         case (state)
24             S0: if (timer > 5) next_state = S1;
25             S1: if (timer > 1) next_state = S2;
26             S2: if (timer > 1) next_state = S3;
27             S3: if (timer > 5) next_state = S4;
28             S4: if (timer > 1) next_state = S5;
29             S5: if (timer > 1) next_state = S0;
30             default: next_state = S0;
31         endcase
32     end
33 
```



```

34      // State and timer sequential logic
35  always @(posedge clk or posedge reset) begin
36    if (reset) begin
37      timer <= 0;
38      state <= S0;
39    end else begin
40      if (state != next_state) begin
41        // Reset timer when state changes
42        timer <= 0;
43        state <= next_state;
44      end else begin
45        // Increment timer when staying in the same state
46        timer <= timer + 1;
47      end
48    end
49  end
50
51  // Light output logic
52  always @(*) begin
53    case (state)
54      S0: begin NS_Light = 3'b001; EW_Light = 3'b100; end
55      S1: begin NS_Light = 3'b010; EW_Light = 3'b100; end
56      S2: begin NS_Light = 3'b100; EW_Light = 3'b100; end
57      S3: begin NS_Light = 3'b100; EW_Light = 3'b001; end
58      S4: begin NS_Light = 3'b100; EW_Light = 3'b010; end
59      S5: begin NS_Light = 3'b100; EW_Light = 3'b100; end
60      default: begin NS_Light = 3'b100; EW_Light = 3'b100; end
61    endcase
62  end
63
64  // LED mapping logic
65  always @(*) begin
66    LEDR = NS_Light;
67    LEDG = EW_Light;
68  end
69 endmodule

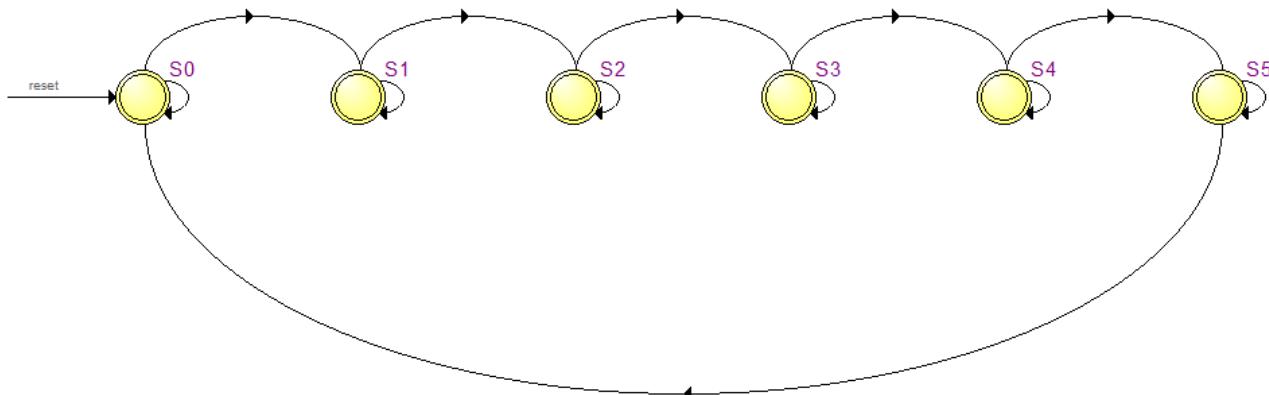
```

### Mô phỏng WaveForm nếu không có chân Clk\_50:





### Sơ đồ trạng thái:



### Giải thích:

#### Hỗn năng:

Tiều khiển **đèn giao thông 2 chiều** (Bắc-Nam và Đông-Tây) bằng FSM, mỗi chiều có đèn Đỏ–Vàng–Xanh. Tần số sử dụng clk\_50MHz để tạo thời gian và cho hiển thị trên LEDR (NS) và LEDG (EW).

#### Cách hoạt động chi tiết:

##### . Các trạng thái (state):

- S0: NS xanh, EW đỏ – 6 giây
- S1: NS vàng, EW đỏ – 2 giây
- S2: NS đỏ, EW đỏ – 2 giây
- S3: NS đỏ, EW xanh – 6 giây
- S4: NS đỏ, EW vàng – 2 giây
- S5: NS đỏ, EW đỏ – 2 giây  
→ Sau S5 quay lại S0 → tạo chu kỳ 20 giây.

##### . Chuyển trạng thái:

- Dựa vào bộ đếm timer và thời gian định sẵn:
  - 6 giây = 300\_000\_000 chu kỳ (50 MHz)
  - 2 giây = 100\_000\_000 chu kỳ

Khi timer đạt ngưỡng, chuyển sang trạng thái kế tiếp và reset timer.



### . Cập nhật timer và trạng thái:

- Tại mỗi cạnh lên của clk\_50MHz, nếu không reset:
  - Nếu có chuyển trạng thái → reset timer.
  - Nếu chưa chuyển → timer tăng.

### . Hiển thị ra đèn LED:

- NS\_Light → LEDR
- EW\_Light → LEDG  
→ Mỗi chiều dùng 3 bit để hiển thị:
  - 3'b001 = xanh
  - 3'b010 = vàng
  - 3'b100 = đỏ

#### Ví dụ:

↳ S0:

- NS: 3'b001 (xanh) → LEDR = xanh
- EW: 3'b100 (đỏ) → LEDG = đỏ

au 6 giây → sang S1 (NS vàng, EW đỏ) → và tiếp tục theo chu kỳ.

### Câu 3: (Tuỳ chọn)

Mã Morse là mã có dạng các mẫu xung ngắn và dài để thể hiện một thông điệp. Mỗi chữ cái được biểu diễn dưới dạng một chuỗi các dấu chấm (xung ngắn) và dấu gạch ngang (xung dài) như Hình 3.3. Ví dụ: tám chữ cái đầu tiên của bảng chữ cái có cách biểu diễn sau:

	UIT	TRƯỜNG ĐẠI HỌC	CÔNG NGHỆ THÔNG TIN
A	•—		
B	—•••		
C	—•—•		
D	—••		
E	•		
F	••—•		
G	——•		
H	••••		

Hình 3.3 - Minh họa mã Morse



Thiết kế và hiện thực mạch mã hóa mã Morse trên bảng FSM. Ngõ vào của mạch là một trong tám chữ cái đầu tiên của bảng chữ cái và ngõ ra là mã Morse tương ứng được hiển thị trên đèn LED. Sử dụng công tắc SW2-0 và nút bấm KEY1-0 làm đầu vào. Khi người dùng nhấn KEY1, mạch sẽ hiển thị mã Morse cho một chữ cái được chỉ định bởi SW2-0 (000 cho A, 001 cho B, v.v.), sử dụng các xung 0,5 giây để biểu thị các dấu chấm và các xung 1,5 giây để biểu thị dấu gạch ngang. Ngoài ra mạch có chức năng reset bất đồng bộ thông qua ngõ KEY0.

### Code thực thi:

```

1  module Morse(
2    input CLK,
3    input [2:0] SW,
4    input [1:0] KEY,
5    output reg [3:0] LEDR
6  );
7    reg [27:0] timer;
8    reg [3:0] morse_code;
9    reg start;
10   reg prev_key1;
11   reg [27:0] max_time;
12
13 // Phát hiện cạnh lên KEY[1]
14 wire key1_rising = ~prev_key1 & KEY[1];
15
16 // Bộ timer và điều khiển start
17 always @ (posedge CLK or negedge KEY[0]) begin
18   if (!KEY[0]) begin
19     timer <= 0;
20     start <= 0;
21     prev_key1 <= 0;
22   end else begin
23     prev_key1 <= KEY[1];
24
25     if (key1_rising) begin
26       start <= 1;
27       timer <= 0;
28     end else if (start) begin
29       if (timer < max_time)
30         timer <= timer + 1;
31       else
32         start <= 0;
33     end

```

**UIT**  
**TRƯỜNG ĐẠI HỌC**  
**CÔNG NGHỆ THÔNG TIN**



```

34         end
35     end
36
37     // Gán max_time theo ký tự Morse
38     always @(*) begin
39         case (SW)
40             3'b000: max_time = 100_000_000; // A
41             3'b001: max_time = 150_000_000; // B
42             3'b010: max_time = 200_000_000; // C
43             3'b011: max_time = 125_000_000; // D
44             3'b100: max_time = 25_000_000; // E
45             3'b101: max_time = 150_000_000; // F
46             3'b110: max_time = 175_000_000; // G
47             3'b111: max_time = 125_000_000; // H
48             default: max_time = 0;
49         endcase
50     end
51
52     // Gán mã Morse tương ứng
53     always @(*) begin
54         morse_code = 4'b0000;
55         case (SW)
56             3'b000: begin // A: ..
57                 if (timer < 25_000_000) morse_code = 4'b1000;
58                 else if (timer < 100_000_000) morse_code = 4'b0100;
59             end
60             3'b001: begin // B: -...
61                 if (timer < 75_000_000) morse_code = 4'b1000;
62                 else if (timer < 100_000_000) morse_code = 4'b0100;
63                 else if (timer < 125_000_000) morse_code = 4'b0010;
64                 else if (timer < 150_000_000) morse_code = 4'b0001;
65             end
66             3'b010: begin // C: -..

```



**TRƯỜNG ĐẠI HỌC  
CÔNG NGHỆ THÔNG TIN**



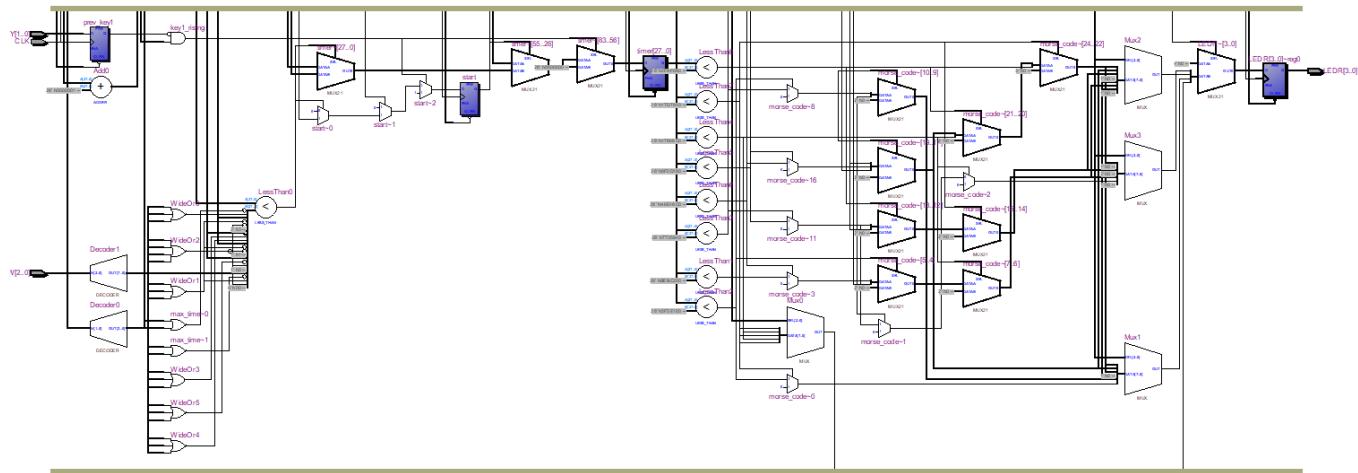
```

67             if (timer < 75_000_000) morse_code = 4'b1000;
68             else if (timer < 100_000_000) morse_code = 4'b0100;
69             else if (timer < 175_000_000) morse_code = 4'b0010;
70             else if (timer < 200_000_000) morse_code = 4'b0001;
71         end
72     3'b011: begin // D: ...
73         if (timer < 75_000_000) morse_code = 4'b1000;
74         else if (timer < 100_000_000) morse_code = 4'b0100;
75         else if (timer < 125_000_000) morse_code = 4'b0010;
76     end
77     3'b100: begin // E: .
78         if (timer < 25_000_000) morse_code = 4'b1000;
79     end
80     3'b101: begin // F: ... .
81         if (timer < 25_000_000) morse_code = 4'b1000;
82         else if (timer < 50_000_000) morse_code = 4'b0100;
83         else if (timer < 125_000_000) morse_code = 4'b0010;
84         else if (timer < 150_000_000) morse_code = 4'b0001;
85     end
86     3'b110: begin // G: ---.
87         if (timer < 75_000_000) morse_code = 4'b1000;
88         else if (timer < 150_000_000) morse_code = 4'b0100;
89         else if (timer < 175_000_000) morse_code = 4'b0010;
90     end
91     3'b111: begin // H: ....
92         if (timer < 25_000_000) morse_code = 4'b1000;
93         else if (timer < 50_000_000) morse_code = 4'b0100;
94         else if (timer < 75_000_000) morse_code = 4'b0010;
95         else if (timer < 125_000_000) morse_code = 4'b0001;
96     end
97 endcase
98 end
99
// Cập nhật LED
.00
.01 always @(posedge CLK or negedge KEY[0]) begin
.02     if (!KEY[0])
.03         LEDR <= 4'b0000;
.04     else if (start)
.05         LEDR <= morse_code;
.06     else
.07         LEDR <= 4'b0000;
.08 end
.09
.10 endmodule

```



### RTL Viewer:



### Code Thực thi nếu không có chân Clk 50::

```

1  module MorseWithoutCLK50(
2      input CLK,
3      input [2:0] SW,
4      input [1:0] KEY,
5      output reg [3:0] LEDR
6  );
7
8      reg [2:0] timer;
9      reg [2:0] max_time;
10     reg [3:0] morse_code;
11     reg start;
12     reg prev_key1;
13
14     wire key1_rising = ~prev_key1 & KEY[1];
15
16     // Timer và phát hiện cạnh lên KEY[1]
17     always @ (posedge CLK or negedge KEY[0]) begin
18         if (!KEY[0]) begin
19             timer <= 0;
20             start <= 0;
21             prev_key1 <= 0;
22         end else begin
23             prev_key1 <= KEY[1];
24
25             if (key1_rising) begin
26                 start <= 1;
27                 timer <= 0;
28             end else if (start) begin
29                 if (timer < max_time)
30                     timer <= timer + 1;
31                 else
32                     start <= 0; // Dừng khi hết thời gian Morse
33         end

```



```

34         end
35     end
36
37 // Xác định mã Morse và thời gian hiển thị tối đa
38 always @(*) begin
39     morse_code = 4'b0000;
40     case (SW)
41         3'b000: begin max_time = 4;
42             if (timer == 1) morse_code = 4'b1000;
43             else if (timer == 2 || timer == 3) morse_code = 4'b0100;
44         end
45         3'b001: begin max_time = 6;
46             if (timer == 1 || timer == 2) morse_code = 4'b1000;
47             else if (timer == 3) morse_code = 4'b0100;
48             else if (timer == 4) morse_code = 4'b0010;
49             else if (timer == 5) morse_code = 4'b0001;
50         end
51         3'b010: begin max_time = 7;
52             if (timer == 1 || timer == 2) morse_code = 4'b1000;
53             else if (timer == 3) morse_code = 4'b0100;
54             else if (timer == 4 || timer == 5) morse_code = 4'b0010;
55             else if (timer == 6) morse_code = 4'b0001;
56         end
57         3'b011: begin max_time = 5;
58             if (timer == 1 || timer == 2) morse_code = 4'b1000;
59             else if (timer == 3) morse_code = 4'b0100;
60             else if (timer == 4) morse_code = 4'b0010;
61         end
62         3'b100: begin max_time = 2;
63             if (timer == 1) morse_code = 4'b1000;
64         end
65         3'b101: begin max_time = 6;
66             if (timer == 1) morse_code = 4'b1000;

```

**UIT**  
**TRƯỜNG ĐẠI HỌC**  
**CÔNG NGHỆ THÔNG TIN**

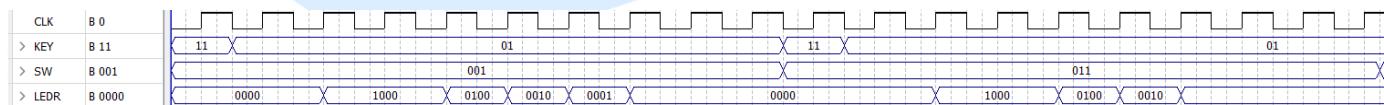


```

67         else if (timer == 2) morse_code = 4'b0100;
68         else if (timer == 3 || timer == 4) morse_code = 4'b0010;
69         else if (timer == 5) morse_code = 4'b0001;
70     end
71     3'b110: begin max_time = 6;
72         if (timer == 1 || timer == 2) morse_code = 4'b1000;
73         else if (timer == 3 || timer == 4) morse_code = 4'b0100;
74         else if (timer == 5) morse_code = 4'b0010;
75     end
76     3'b111: begin max_time = 5;
77         if (timer == 1) morse_code = 4'b1000;
78         else if (timer == 2) morse_code = 4'b0100;
79         else if (timer == 3) morse_code = 4'b0010;
80         else if (timer == 4) morse_code = 4'b0001;
81     end
82     default: begin max_time = 0; end
83   endcase
84 end
85
86 // Hiển thị LED
87 always @ (posedge CLK or negedge KEY[0]) begin
88     if (!KEY[0])
89         LEDR <= 4'b0000;
90     else if (start)
91         LEDR <= morse_code;
92     else
93         LEDR <= 4'b0000;
94 end
95
96 endmodule

```

### Mô phỏng WareForm nếu không có chân Clk 50::



### Giải thích:

**Hỗn năng:** Hiển thị mã Morse cho 8 chữ cái đầu (A–H) qua LEDR[3:0] khi nhấn KEY[1]. Reset bằng KEY[0].

### Cách hoạt động:

## TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

1. Mỗi lần nhấn KEY[1], bộ đếm thời gian (timer) tăng theo clock.
2. Dựa trên giá trị SW, mạch chọn chữ cái cần hiển thị.
3. Trong khoảng thời gian xác định (0.5s là dấu chấm, 1.5s là dấu gạch), mạch **bật từng LED tương ứng** với ký hiệu Morse.



4. Khi hết chuỗi → tắt LED.
5. Nhấn KEY[0] → reset timer và tắt tất cả LED.



**UIT**  
**TRƯỜNG ĐẠI HỌC**  
**CÔNG NGHỆ THÔNG TIN**