|  |
| --- |
| ĐẠI HỌC BÁCH KHOA HÀ NỘI  **TRƯỜNG ĐIỆN – ĐIỆN TỬ**  logo_128  **BÁO CÁO GIỮA KỲ**  **Thiết kế, tổng hợp hệ thống số**  **Đề tài: Thiết kế đèn giao thông ở ngã tư**  **sử dụng ngôn ngữ Verilog**  Giáo viên hướng dẫn: Nguyễn Hoàng Dũng  Sinh viên thực hiện: Phạm Trọng Huy Hoàng – 20182546  Hà Nội, 6/2022 |

**MỤC LỤC**

[DANH MỤC BẢNG BIỂU 1](#_Toc106303301)

[DANH MỤC HÌNH VẼ 2](#_Toc106303302)

[CHƯƠNG 1: CƠ SỞ LÝ THUYẾT VỀ FSM 3](#_Toc106303303)

[1.1 Tổng quan về FSM 3](#_Toc106303304)

[1.1.1 Khái niệm 3](#_Toc106303305)

[1.1.2 Phân loại 4](#_Toc106303306)

[1.1.3 So sánh FSM Moore và FSM Mealy 5](#_Toc106303307)

[1.2 Thiết kế FSM 6](#_Toc106303308)

[1.2.1 Yêu cầu 6](#_Toc106303309)

[1.2.2 Phân tích tổng quan 8](#_Toc106303310)

[1.2.3 Phân tích khối FSM 11](#_Toc106303311)

[CHƯƠNG 2 : THIẾT KẾ VÀ MÔ PHỎNG ĐÈN GIAO THÔNG Ở NGÃ TƯ SỬ DỤNG NGÔN NGỮ VERILOG 14](#_Toc106303312)

[2.1 Sơ đồ khối đèn giao thông ở ngã tư 14](#_Toc106303314)

[2.2 Mô phỏng trên Modelsim 23](#_Toc106303315)

[CHƯƠNG 3. References 24](#_Toc106303316)

# DANH MỤC BẢNG BIỂU

[Bảng 1. Bảng chuyển trạng thái đèn giao thông 7](file:///C:\Users\Admin\Desktop\Báo%20cáo%20IC%20số.docx#_Toc106281933)

[Bảng 2. Bảng giá trị ngõ ra bộ điều khiển tương ứng với trạng thái đèn 9](file:///C:\Users\Admin\Desktop\Báo%20cáo%20IC%20số.docx#_Toc106281934)

[Bảng 3. Bảng giá trị ngõ ra FSM 12](file:///C:\Users\Admin\Desktop\Báo%20cáo%20IC%20số.docx#_Toc106281935)

# DANH MỤC HÌNH VẼ

[Hình 1. Mô hình cơ bản của FSM 3](file:///C:\Users\Admin\Desktop\Báo%20cáo%20IC%20số.docx#_Toc106281855)

[Hình 2. FSM Moore 4](file:///C:\Users\Admin\Desktop\Báo%20cáo%20IC%20số.docx#_Toc106281856)

[Hình 3. FSM Mealy 4](file:///C:\Users\Admin\Desktop\Báo%20cáo%20IC%20số.docx#_Toc106281857)

[Hình 4. FSM sử dụng ROM để lưu giá trị ngõ ra và trạng thái kế tiếp 5](file:///C:\Users\Admin\Desktop\Báo%20cáo%20IC%20số.docx#_Toc106281858)

[Hình 5. Sử dụng FSM Moore để tránh tạo mạch bất đồng bộ không mong muốn khi hồi tiếp ngõ ra 5](file:///C:\Users\Admin\Desktop\Báo%20cáo%20IC%20số.docx#_Toc106281859)

[Hình 6. Độ trễ ngõ ra của FSM Mealy phụ thuộc vào độ trễ ngõ vào nhưng FSM Moore thì không 6](file:///C:\Users\Admin\Desktop\Báo%20cáo%20IC%20số.docx#_Toc106281860)

[Hình 7. Sơ đồ ngã tư với đèn giao thông 8](#_Toc106281861)

[Hình 8. Sơ đồ tín hiệu giao tiếp của bộ điều khiển đèn giao thông 9](file:///C:\Users\Admin\Desktop\Báo%20cáo%20IC%20số.docx#_Toc106281862)

[Hình 9. Sơ đồ khối của bộ điều khiển đèn giao thông 10](file:///C:\Users\Admin\Desktop\Báo%20cáo%20IC%20số.docx#_Toc106281863)

[Hình 10. FSM điều khiển đèn giao thông 11](file:///C:\Users\Admin\Desktop\Báo%20cáo%20IC%20số.docx#_Toc106281864)

[Hình 11. Sơ đồ nguyên lý của FSM 13](file:///C:\Users\Admin\Desktop\Báo%20cáo%20IC%20số.docx#_Toc106281865)

[Hình 12. Hình mô phỏng trên Model sim 23](#_Toc106281866)

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT VỀ FSM

## Tổng quan về FSM

### Khái niệm

Máy trạng thái hữu hạn, viết tắt là FSM, là một thành phần được sử dụng phổ biến trong thiết kế vi mạch số với ưu điểm là dễ kiểm soát quá trình hoạt động của thiết kế và dễ debug hoạt động của thiết kế.

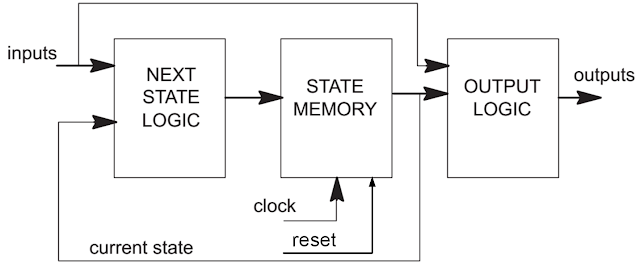
|  |
| --- |
| [https://1.bp.blogspot.com/-M8hoK35DiXI/XcgH3Cj04ZI/AAAAAAAALaY/vVVZwyQ6cWAOxufaS3DztYbX6VL16RmQwCKgBGAsYHg/s640/image.png](https://1.bp.blogspot.com/-M8hoK35DiXI/XcgH3Cj04ZI/AAAAAAAALaY/vVVZwyQ6cWAOxufaS3DztYbX6VL16RmQwCKgBGAsYHg/s1600/image.png)  Hình . Mô hình cơ bản của FSM |
|  |

 FSM gồm có 3 thành phần cơ bản như sau:

* Mạch tạo trạng thái kế tiếp (Next state logic) là mạch tổ hợp phụ thuộc vào ngõ vào FSM và giá trị trạng thái hiện tại lấy từ bộ nhớ trạng thái (state memory)
* Bộ nhớ trạng thái (state memory) là phần tử lưu trạng thái hiện tại của FSM nó có thể là Flip-Flop, Latch, ... lấy ngõ vào từ mạch tạo trạng thái kế tiếp. Bộ nhớ trạng thái thường được sử dụng trong các thiết kế đồng bộ là FF hoạt động theo xung clock. Một tín hiệu reset có thể phải sử dụng để khởi động FSM đến một giá trị ban đầu. Reset không cần sử dụng đối với các FSM luôn hoạt động đúng dù giá trị ban đầu của FF là bao nhiêu.
* Mạch tạo ngõ ra (output logic) là mạch tổ hợp tạo giá trị ngõ ra tương ứng với trạng thái hiện tại của FSM. Mạch này lấy ngõ vào là giá trị trạng thái hiện tại và có thể tổ hợp thêm ngõ vào của FSM

### Phân loại

FSM được chia làm 2 loại:

* FSM Moore là loại có mạch tạo ngõ ra không phụ thuộc trực tiếp vào ngõ vào FSM
* [](https://1.bp.blogspot.com/-px2vqkRXt4w/XcgIBpfhJbI/AAAAAAAALag/ISlRLFtWpzkgmbxHTEB0TFdYhiqy9_a5gCKgBGAsYHg/s1600/image.png)FSM Mealy là loại có mạch tạo ngõ ra phụ thuộc trực tiếp vào ngõ vào FSM

Hình . FSM Moore

|  |
| --- |
|  |
| [https://1.bp.blogspot.com/-dFL2HO8i2jA/XcgH80bsGuI/AAAAAAAALac/rdexjarABa4wT1mg0NOO8pcaSDdg48VVQCKgBGAsYHg/s640/image.png](https://1.bp.blogspot.com/-dFL2HO8i2jA/XcgH80bsGuI/AAAAAAAALac/rdexjarABa4wT1mg0NOO8pcaSDdg48VVQCKgBGAsYHg/s1600/image.png) |

|  |
| --- |
|  |
| Hình . FSM Mealy |

Với FSM Mealy, mạch tạo trạng thái kế tiếp và ngõ ra có thể được thay bằng ROM trong một số FPGA. Thay vì sử dụng mạch logic để tính toán giá trị ngõ ra và trạng thái kế tiếp thì bộ ROM được sử dụng để lưu các giá trị này ứng với mỗi trạng thái hiện tại. Lúc này ngõ vào và giá trị trạng thái hiện tại được sử dụng để điều khiển và tạo địa chỉ truy cập ROM.

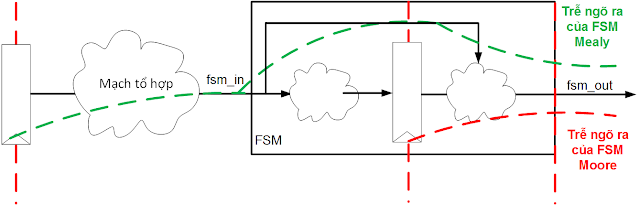
|  |
| --- |
| [https://1.bp.blogspot.com/-tPVDtVq4O18/XcgIH7vIrqI/AAAAAAAALak/nJpdG3zI8DY8a94Jk53GboV9I693ulVagCKgBGAsYHg/s640/image.png](https://1.bp.blogspot.com/-tPVDtVq4O18/XcgIH7vIrqI/AAAAAAAALak/nJpdG3zI8DY8a94Jk53GboV9I693ulVagCKgBGAsYHg/s1600/image.png)  Hình . FSM sử dụng ROM để lưu giá trị ngõ ra và trạng thái kế tiếp |

|  |
| --- |
|  |
|  |

### So sánh FSM Moore và FSM Mealy

FSM Moore có tính an toàn cao hơn FSM Mealy vì ngõ ra FSM được tổ hợp từ giá trị FF nên nếu được kết nối đến các khối khác thì tín hiệu này có thể được sử dụng làm đường hồi tiếp trở lại tổ hợp với các mạch tạo ngõ vào cho FSM. Còn đối với FSM Mealy thì việc nay sẽ tạo ra mạch bất đồng bộ.

|  |
| --- |
| [https://1.bp.blogspot.com/-bMLVj08Jv9Y/XcgIjDE3iwI/AAAAAAAALas/BmqWC5I_Nb8UrPb9YfvL_uH2Frlf6J9mwCKgBGAsYHg/s640/image.png](https://1.bp.blogspot.com/-bMLVj08Jv9Y/XcgIjDE3iwI/AAAAAAAALas/BmqWC5I_Nb8UrPb9YfvL_uH2Frlf6J9mwCKgBGAsYHg/s1600/image.png)  Hình . Sử dụng FSM Moore để tránh tạo mạch bất đồng bộ không mong muốn khi hồi tiếp ngõ ra |
|  |

[](https://1.bp.blogspot.com/-ck14jbCaCdE/XcgIp_JMt2I/AAAAAAAALaw/x1SwKcWcT0cbz60JSaLILr-IuVGjXi7-ACKgBGAsYHg/s1600/image.png)Độ trễ ngõ ra FSM Moore được xác định rõ ràng và giới hạn trong FSM tính từ FF đi qua mạch tổ hợp tạo ngõ ra. Trong khi ngõ ra FSM Mealy phụ thuộc cả vào ngõ vào nên độ trễ của các tầng trước nối đến ngõ vào FSM cũng được tính vào độ trễ ngõ ra FSM và có thể là đường critical path (đường có độ trễ lớn nhất).

Hình . Độ trễ ngõ ra của FSM Mealy phụ thuộc vào độ trễ ngõ vào nhưng FSM Moore thì không

|  |
| --- |
|  |
|  |

Mealy có ưu điểm là số trạng thái ít hơn Moore với cùng một chức năng và ngõ ra đáp ứng ngay theo sự thay đổi của ngõ vào mà không cần chờ đến cạnh lên xung clock. Nếu dùng Mealy thì cần đặc biệt chú ý đến vấn đề hồi tiếp và độ trễ cộng dồn như đã trình bày ở trên.

## Thiết kế FSM

### Yêu cầu

Sử dụng FSM để thiết kế một bộ điều khiển đèn giao thông tại một ngã tư đường với yêu cầu:

* Thời gian dừng tại mỗi trạng thái đèn có thể cấu hình được trước khi biên dịch RTL code
* Tần số điều khiển đèn là 1 Hz, tương ứng với chu kỳ 1 s
* Sự chuyển trạng thái đèn được xác định như bảng sau
* Mỗi đường sẽ có 3 tín hiệu điều khiển đèn tương ứng với 3 màu Green/Yellow/Red
* Trạng thái khởi động là trạng thái đèn trên cả STREET A và STREET B đều đỏ (Red)

|  |
| --- |
| [https://1.bp.blogspot.com/-KjC_5S4tLqo/XcgIv84JJkI/AAAAAAAALa0/3Xn65Rh0-qYiiOMg7Evx-Q-BBM18HiZigCKgBGAsYHg/s640/image.png](https://1.bp.blogspot.com/-KjC_5S4tLqo/XcgIv84JJkI/AAAAAAAALa0/3Xn65Rh0-qYiiOMg7Evx-Q-BBM18HiZigCKgBGAsYHg/s1600/image.png)  Bảng . Bảng chuyển trạng thái đèn giao thông |
|  |

|  |
| --- |
| [https://1.bp.blogspot.com/--MN1tqIaKsM/XcgIzyJkbLI/AAAAAAAALa4/mD0sdOVh5vg8u67S8imOr3L1_SoowxAUQCKgBGAsYHg/s640/image.png](https://1.bp.blogspot.com/--MN1tqIaKsM/XcgIzyJkbLI/AAAAAAAALa4/mD0sdOVh5vg8u67S8imOr3L1_SoowxAUQCKgBGAsYHg/s1600/image.png)  Hình . Sơ đồ ngã tư với đèn giao thông |
|  |

### Phân tích tổng quan

Từ yêu cầu ban đầu, chúng ta sẽ có một thiết kế gồm các tín hiệu như sau:

|  |
| --- |
|  |
| [https://1.bp.blogspot.com/-gHpbnj2HeU4/XcgI5qbCMdI/AAAAAAAALa8/SZWF1MlCwuAk-C8NzF_AGzSLobhS07LEgCKgBGAsYHg/s640/image.png](https://1.bp.blogspot.com/-gHpbnj2HeU4/XcgI5qbCMdI/AAAAAAAALa8/SZWF1MlCwuAk-C8NzF_AGzSLobhS07LEgCKgBGAsYHg/s1600/image.png)  Hình . Sơ đồ tín hiệu giao tiếp của bộ điều khiển đèn giao thông |

Trong đó:

* clk là xung clock tần số 1 Hz
* rst\_n là tín hiệu dùng để khởi động bộ điều khiển đến trạng thái ban đầu Red-Red khi nó tích cực mức thấp
* street\_a[2:0] là tín hiệu điều khiển đèn trên đường STREET A. Thứ tự bit từ 0 đến 2 tương ứng với trạng thái Red/Yellow/Green. Tích cực mức 1.
* street\_b[2:0] là tín hiệu điều khiển đèn trên đường STREET B. Thứ tự bit từ 0 đến 2 tương ứng với trạng thái Red/Yellow/Green. Tích cực mức 1.

Bảng giá trị ngõ ra tương ứng với các trạng thái đèn như sau:

|  |
| --- |
| [https://1.bp.blogspot.com/-daKbIV15DRI/XcgI-G5ubSI/AAAAAAAALbA/ILMvjop-6E8jdT7qLw2Z5aA8VcxIHd4kACKgBGAsYHg/s640/image.png](https://1.bp.blogspot.com/-daKbIV15DRI/XcgI-G5ubSI/AAAAAAAALbA/ILMvjop-6E8jdT7qLw2Z5aA8VcxIHd4kACKgBGAsYHg/s1600/image.png)  Bảng . Bảng giá trị ngõ ra bộ điều khiển tương ứng với trạng thái đèn |
|  |

Căn cứ trên hoạt động mà yêu cầu đề ra, sơ đồ khối của bộ điều khiển đèn giao thông như sau:

* TIME\_COUNTER: dùng để đếm xunh clock xác định thời gian duy trì trạng thái của đèn. Đây là thời gian có thể cấu hình được trước khi biên dịch như yêu cầu đặt ra. Ba tín hiệu g\_end, r\_end và y\_end báo thời điểm kết thúc của trạng thái GREEN, RED và YELLOW. Ba tín hiệu fsm\_g, fsm\_r và fsm\_y báo trạng thái GREEN, REG và YELLOW để bộ đếm hoạt động theo giá trị cấu hình phù hợp.
* FSM: là máy trạng thái sẽ tạo ngõ ra street\_a, street\_b và fsm\_g, fsm\_r, fsm\_y

|  |
| --- |
| [https://1.bp.blogspot.com/-5HN9cMpXfGM/XcgJDctlkdI/AAAAAAAALbE/regbQ0sTXT47co1rW-ZgjN6UY9A11Q8wACKgBGAsYHg/s640/image.png](https://1.bp.blogspot.com/-5HN9cMpXfGM/XcgJDctlkdI/AAAAAAAALbE/regbQ0sTXT47co1rW-ZgjN6UY9A11Q8wACKgBGAsYHg/s1600/image.png)  Hình . Sơ đồ khối của bộ điều khiển đèn giao thông |

|  |
| --- |
|  |

### Phân tích khối FSM

Ứng với 6 cặp trạng thái nên chúng ta có FSM 6 trạng thái như sau:

|  |
| --- |
| [https://1.bp.blogspot.com/-LicvlkNRHCU/XcgJJKEjyEI/AAAAAAAALbI/SmSRvXBQdo0GCdHVj8FI7SfdYmjuek7PwCKgBGAsYHg/s640/image.png](https://1.bp.blogspot.com/-LicvlkNRHCU/XcgJJKEjyEI/AAAAAAAALbI/SmSRvXBQdo0GCdHVj8FI7SfdYmjuek7PwCKgBGAsYHg/s1600/image.png)  Hình . FSM điều khiển đèn giao thông |
|  |

Trong đó, ký hiệu A và B ứng với STREET A và STREET B. G là GREEN, R là RED và Y là YELLOW. Ví dụ, AG\_BR nghĩa là trạng thái STREET A đang GREEN, STREET B đang RED.

Yêu cầu đặt ra, sau khi reset thì cả hai đèn phải RED-RED nên bạn có thể chọn AR\_BR1 hoặc AR\_BR2. Ở đây, trạng thái AR\_BR1 được chọn làm trạng thái khởi động.

Việc chuyển trạng thái chỉ xảy ra khi tín hiệu báo độ trễ tương ứng của trạng thái đó tích cực. Ví dụ, khi đèn trên STREET A hoặc STREET B đang xanh thì g\_end phải tích cực thì mới chuyển sang trạng thái tiếp theo.

|  |
| --- |
| [https://1.bp.blogspot.com/-vM6e8iFwWpI/XcgJOxpgQFI/AAAAAAAALbM/RtV7Ze9UgNEBqppcqPkqipFq5PddAx2qwCKgBGAsYHg/s640/image.png](https://1.bp.blogspot.com/-vM6e8iFwWpI/XcgJOxpgQFI/AAAAAAAALbM/RtV7Ze9UgNEBqppcqPkqipFq5PddAx2qwCKgBGAsYHg/s1600/image.png)  Bảng . Bảng giá trị ngõ ra FSM |

|  |
| --- |
|  |

Sơ đồ nguyên lý tương ứng của FSM như sau:

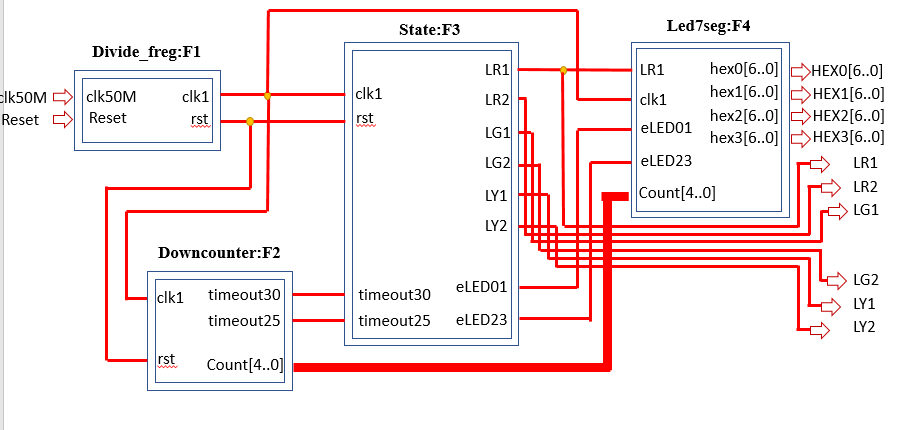
|  |
| --- |
| [https://1.bp.blogspot.com/-WqmWjlyywBg/XcgJZpQy-JI/AAAAAAAALbQ/d5qjKtjaOUMG-YHNZxqpzOsdKg2vmBalQCKgBGAsYHg/s640/image.png](https://1.bp.blogspot.com/-WqmWjlyywBg/XcgJZpQy-JI/AAAAAAAALbQ/d5qjKtjaOUMG-YHNZxqpzOsdKg2vmBalQCKgBGAsYHg/s1600/image.png)  Hình . Sơ đồ nguyên lý của FSM |
|  |

|  |
| --- |
|  |

CHƯƠNG 2 : THIẾT KẾ VÀ MÔ PHỎNG ĐÈN GIAO THÔNG Ở NGÃ TƯ SỬ DỤNG NGÔN NGỮ VERILOG



## Sơ đồ khối đèn giao thông ở ngã tư



Code minh họa cho từng khối:

* Khối Divide\_freg:

`timescale 1ns / 1ps

module Divide\_freg(

    input clk50M,

    input Reset,

    output reg clk1,

    output reg rst

    );

localparam divide\_count = 25000000;

reg [31:0]count;

always@(posedge clk50M , posedge Reset)

begin

    if(Reset)

        begin

        clk1 <= 1'b0;

        count <= 32'b0;

        end

    else if(count == divide\_count - 1)

        begin

        clk1 <= ~ clk1;

        count <= 32'b0;

        end

    else

       begin

       clk1 <= clk1;

       count <= count + 1 ;

       end

end

always@(\*)

begin

    if(Reset)

        begin

        rst = 1;

        end

    else

        begin

        rst = 0;

        end

end

endmodule

* Khối Downcounter:

module Downcounter(

    input clk1,

    input rst,

    output  reg timeout30,

    output  reg timeout25,

    output reg [4:0] Count

    );

localparam count\_timeout25 = 25 ;

localparam count\_timeout30 = 30 ;

always@(posedge clk1, posedge rst)

begin

    if(rst)

        begin

        Count <= 5'b0;

        end

    else if (Count == count\_timeout30 - 1)

        begin

        Count <= 5'b0;

        end

    else

        begin

        Count <= Count + 1;

        end

end

always@(\*)

begin

     if (Count == count\_timeout25 - 1)

        begin

        timeout25 = 1;

        end

    else

        begin

         timeout25 = 0;

        end

end

always@(\*)

begin

     if (Count == count\_timeout30 - 1)

        begin

        timeout30 = 1;

        end

    else

        begin

         timeout30 = 0;

        end

end

endmodule

* Khối Led7seg:

module Led7seg(

    input LR1,

    input clk1,

    input eLED01,

    input eLED23,

    input [4:0] Count,

    output reg [6:0] hex0,

    output reg [6:0] hex1,

    output reg [6:0] hex2,

    output reg [6:0] hex3

    );

reg [4:0] countled;

reg [3:0] countled1;

reg [3:0] countled2;

always@(posedge clk1)

begin

    countled  = Count%25;

    countled1 = countled/10;

    countled2 = countled%10;

end

// gia su led 0 sang

always @(\*)

  begin

  if(eLED01) begin

    case(countled2)

      0       :  hex0 <= 7'b0000001;

      1       :  hex0 <= 7'b1001111;

      2       :  hex0 <= 7'b0010010;

      3       :  hex0 <= 7'b0000110;

      4       :  hex0 <= 7'b1001100;

      5       :  hex0 <= 7'b0100100;

      6       :  hex0 <= 7'b0100000;

      7       :  hex0 <= 7'b0001111;

      8       :  hex0 <= 7'b0000000;

      default :  hex0 <= 7'b0000100;

    endcase

    case(countled1)

      0       :  hex1 <= 7'b0000001;

      1       :  hex1 <= 7'b1001111;

      2       :  hex1 <= 7'b0010010;

      3       :  hex1 <= 7'b0000110;

      4       :  hex1 <= 7'b1001100;

      5       :  hex1 <= 7'b0100100;

      6       :  hex1 <= 7'b0100000;

      7       :  hex1 <= 7'b0001111;

      8       :  hex1 <= 7'b0000000;

      default :  hex1 <= 7'b0000100;

    endcase

  end

  end

  always @(\*)

  begin

  if(eLED23) begin

    case(countled2)

      0       :  hex2 <= 7'b0000001;

      1       :  hex2 <= 7'b1001111;

      2       :  hex2 <= 7'b0010010;

      3       :  hex2 <= 7'b0000110;

      4       :  hex2 <= 7'b1001100;

      5       :  hex2 <= 7'b0100100;

      6       :  hex2 <= 7'b0100000;

      7       :  hex2 <= 7'b0001111;

      8       :  hex2 <= 7'b0000000;

      default :  hex2 <= 7'b0000100;

    endcase

    case(countled1)

      0       :  hex3 <= 7'b0000001;

      1       :  hex3 <= 7'b1001111;

      2       :  hex3 <= 7'b0010010;

      3       :  hex3 <= 7'b0000110;

      4       :  hex3 <= 7'b1001100;

      5       :  hex3 <= 7'b0100100;

      6       :  hex3 <= 7'b0100000;

      7       :  hex3 <= 7'b0001111;

      8       :  hex3 <= 7'b0000000;

      default :  hex3 <= 7'b0000100;

    endcase

  end

  end

endmodule

* Khối State:

module State(

    input clk1,

    input rst,

    input timeout30,

    input timeout25,

    output reg LR1,

    output reg LR2,

    output reg LG1,

    output reg LG2,

    output reg LY1,

    output reg LY2,

    output reg eLED01,

    output reg eLED23

    );

localparam [1:0]

    NS\_go = 2'b00,

    EW\_go = 2'b01,

    WaitA  = 2'b10,

    WaitB   =2'b11;

reg[1:0] state\_reg, state\_next;

always@(posedge clk1, posedge rst)

begin

    if(rst)

        begin

        state\_reg <= NS\_go;

        end

    else

        begin

        state\_reg <= state\_next;

        end

end

always@(state\_reg,timeout25, timeout30)

begin

    state\_next = state\_reg;

    LR1=0;LR2=0;LG1=0;LG2=0;LY1=0;LY2=0;

    case(state\_reg)

        NS\_go: begin

            LR1=1;LR2=0;LG1=0;LG2=1;LY1=0;LY2=0;

            if(timeout25) begin

                state\_next = WaitA;

                end

            else begin

                state\_next = NS\_go;

                end

        end

        EW\_go: begin

            LR1=0;LR2=1;LG1=1;LG2=0;LY1=0;LY2=0;

            if(timeout25) begin

                state\_next = WaitB;

            end

            else begin

                state\_next = EW\_go;

            end

        end

        WaitA: begin

            LR1=0;LR2=0;LG1=0;LG2=0;LY1=1;LY2=1;

            if(timeout30) begin

                state\_next = EW\_go;

            end

            else begin

                state\_next = WaitA;

            end

        end

        WaitB: begin

            LR1=0;LR2=0;LG1=0;LG2=0;LY1=1;LY2=1;

            if(timeout30) begin

                state\_next = NS\_go;

            end

            else begin

                state\_next = WaitB;

            end

        end

    endcase

end

always@(\*)begin

    if(rst) begin

        eLED01 = 1'b0;

        eLED23 = 1'b0;

        end

    else begin

        eLED01 = 1'b1;

        eLED23 = 1'b1;

    end

end

endmodule

* Traffic\_light module:

module Traffic\_Light(

    input clk50M,

    input Reset,

    output [6:0] HEX0,

    output [6:0] HEX1,

    output [6:0] HEX2,

    output [6:0] HEX3,

    output LR1,

    output LR2,

    output LG1,

    output LG2,

    output LY1,

    output LY2

    );

wire clk1,rst,timeout25,timeout30,eLED01,eLED23;

wire[4:0] Count;

Divide\_freg F1(.clk50M(clk50M),.Reset(Reset),.clk1(clk1),.rst(rst));

Downcounter F2(.clk1(clk1),.rst(rst),.timeout25(timeout25),.timeout30(timeout30),.Count(Count));

State F3(.clk1(clk1),.rst(rst),.timeout25(timeout25),.timeout30(timeout30),.LR1(LR1),.LR2(LR2),.LY1(LY1),.LY2(LY2),.LG1(LG1),.LG2(LG2),.eLED01(eLED01),.eLED23(eLED23));

Led7seg F4(.LR1(LR1),.clk1(clk1),.eLED01(eLED01),.eLED23(eLED23),.Count(Count),.hex0(HEX0),.hex1(HEX1),.hex2(HEX2),.hex3(HEX3));

endmodule

* Testbench:

`timescale 1ns / 1ps

module TestbenchTrafficLight();

reg clk50M,Reset;

wire LR1,LR2,LG1,LG2,LY1,LY2;

wire [6:0] HEX0,HEX1,HEX2,HEX3;

Traffic\_Light UUT(.clk50M(clk50M),.Reset(Reset),.LR1(LR1),.LR2(LR2),.LY1(LY1),.LY2(LY2),.LG1(LG1),.LG2(LG2),.HEX0(HEX0),.HEX1(HEX1),.HEX2(HEX2),.HEX3(HEX3));

initial

    begin

    clk50M =0;

    Reset = 0;

    #10;  Reset = 1;

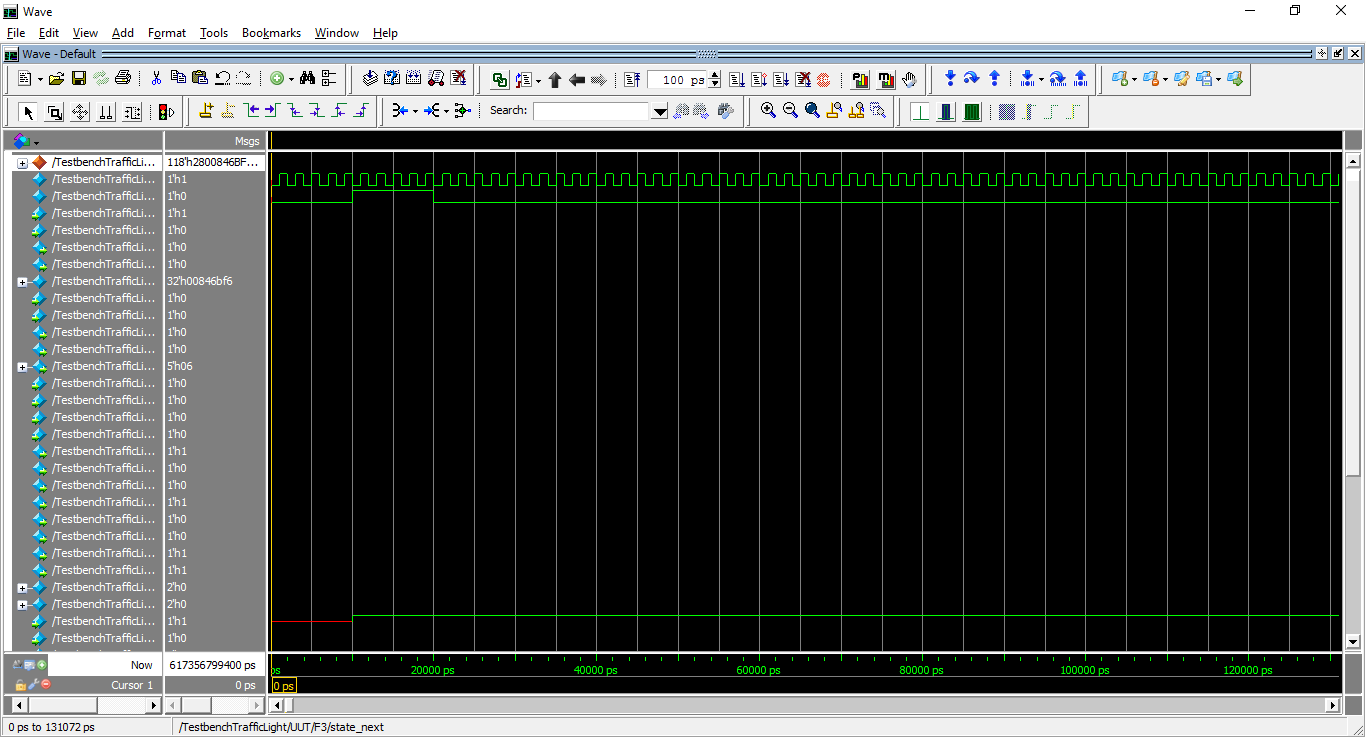
    #10;  Reset = 0;

    end

always #1 clk50M = ~clk50M;

endmodule

## Mô phỏng trên Modelsim



Hình . Hình mô phỏng trên Model sim

# References

Đạt, N. T. (20202). *Report Traffic Light Controller.* Nguyễn Tiến Đạt.

Hưng, Đ. V. (2016, 11 15). *123doc*. Retrieved from 123doc: https://123docz.net/document/4740231-bao-cao-thiet-ke-dieu-khien-den-giao-thong-bang-ngon-ngu-verilog-co-code.htm

Quân, N. (2019, 11 10). *[Verilog][System Verilog] Máy trạng thái hữu hạn FSM*. Retrieved from VLSI TECHNOLOGY: https://nguyenquanicd.blogspot.com/2017/08/verilogsystem-verilog-may-trang-thai.html