ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC – TỰ NHIÊN
KHOA ĐIỆN TỬ – VIỄN THÔNG
BỘ MÔN THỰC HÀNH THIẾT KẾ LOGIC KHẢ TRÌNH
----------o0o----------

THỰC HÀNH THIẾT KẾ LOGIC KHẢ TRÌNH

TP. HỒ CHÍ MINH, THÁNG 4 NĂM 2024

ĐẠI HỌC QUỐC GIA TP.HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC – TỰ NHIÊN
KHOA ĐIỆN TỬ – VIỄN THÔNG
BỘ MÔN THỰC HÀNH LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

----------o0o----------

THỰC HÀNH THIẾT KẾ LOGIC KHẢ TRÌNH

GVHD: CN.Mã Khải Minh

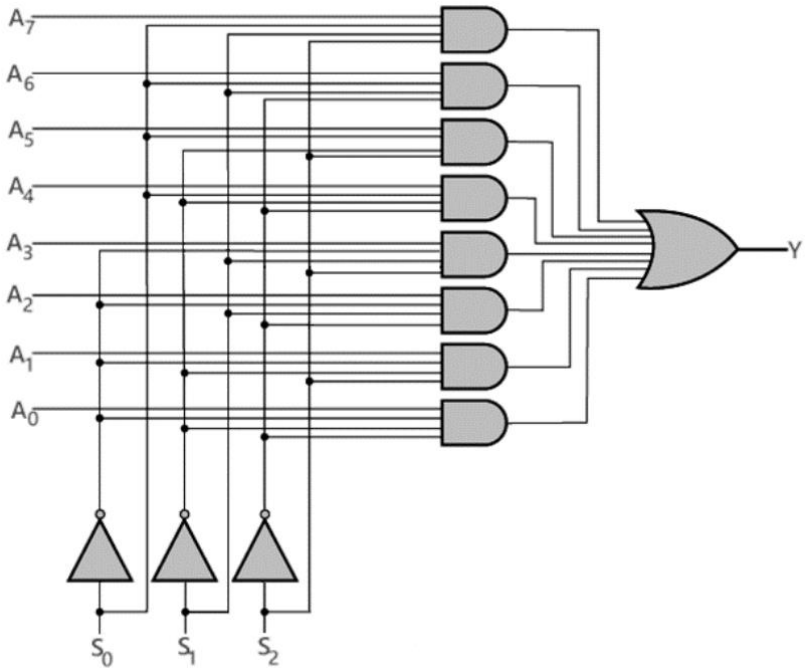
Báo Cáo Cá Nhân

1. Bùi Minh Nhựt – 21207070

TP. HỒ CHÍ MINH, THÁNG 4 NĂM 2024

CHƯƠNG 1: TÌM HIỂU VÀ THIẾT KẾ MẠCH TỔ HỢP

Tìm hiểu và thiết kế mạch tổ hợp như hình ở dưới, sử dụng ngôn ngữ Verilog HDL và lập bảng chân trị (truth table) phù hợp cho mạch này. Cho biết biết S0 → S2 và A0 → A7 là các ngõ vào, và Y ngõ ra. **Hãy cho biết đây là mạch gì và chức năng của nó? Thực hiện mạch trên board FPGA DE10** với S0, S1, S2 được gán với KEY0, KEY1, và KEY2, A0 → A7 được gán với SW0 → SW7, và Y được gán với LED0.



Phần Code:

```
module Cau1(
    input [0:2] KEY,
    input [7:0] SW,
    output LEDR
);
wire C0, c1, c2, c3, c4, c5, c6, c7;
     assign c0 = SW[0] & KEY[0] & KEY[1] & KEY[2];
     assign c1 = SW[1] & KEY[0] & KEY[1] & ~KEY[2];
     assign c2 = SW[2] & KEY[0] & ~KEY[1] & KEY[2];
```

```
assign c3 = SW[3] & KEY[0] & ~KEY[1] & ~KEY[2];
    assign c4 = SW[4] & ~KEY[0] & KEY[1] & KEY[2];
    assign c5 = SW[5] & ~KEY[0] & KEY[1] & ~KEY[2];
    assign c6 = SW[6] & ~KEY[0] & ~KEY[1] & KEY[2];
    assign c7 = SW[7] & ~KEY[0] & ~KEY[1] & ~KEY[2];
    assign LEDR = c0 | c1 | c2 | c3 | c4 | c5 | c6 | c7;
endmodule
```

Đầu vào (input):

- **[0:2] KEY:** Đây là một mảng ba nút nhấn. Mỗi bit của KEY có thể nhận giá trị 0 hoặc 1.
- **[7:0] SW:** Đây là một mảng tám công tắc gạt, với mỗi công tắc có thể ở trạng thái bật (1) hoặc tắt (0).

Đầu ra (output):

- **LEDR:** Đèn LED đơn, sẽ sáng lên phụ thuộc vào các điều kiện định trước của các đầu vào KEY và SW.

Trong module, có tám biến trung gian (c0 đến c7) được sử dụng để xác định trạng thái của đèn LED dựa trên sự kết hợp của trạng thái các nút nhấn (KEY) và công tắc gạt (SW). Mỗi biến cX tương ứng với một điều kiện riêng biệt:

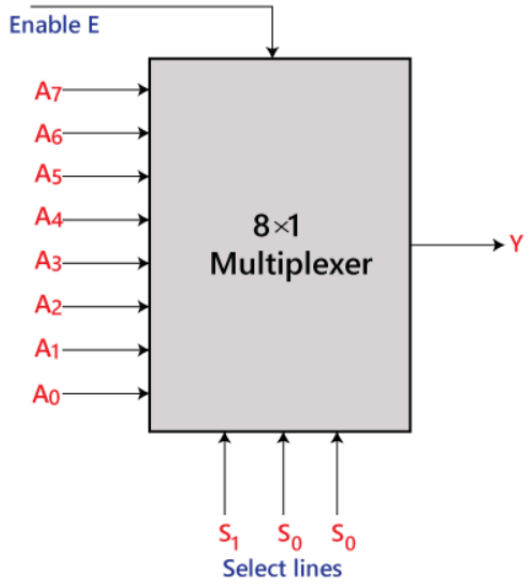
- **c0:** Đèn LED sẽ sáng nếu **SW[0]** và **KEY[0], KEY[1],** và **KEY[2]** đều ở trạng thái bật.
- **c1:** Đèn LED sẽ sáng nếu **SW[1], KEY[0], KEY[1]** ở trạng thái bật và **KEY[2]** ở trạng thái tắt.
- **c2:** Đèn LED sẽ sáng nếu **SW[2], KEY[0], KEY[2]** ở trạng thái bật và **KEY[1]** ở trạng thái tắt.
- **c3:** Đèn LED sẽ sáng nếu **SW[3], KEY[0]** ở trạng thái bật và **KEY[1], KEY[2]** ở trạng thái tắt.
- **c4:** Đèn LED sẽ sáng nếu **SW[4], KEY[1], KEY[2]** ở trạng thái bật và **KEY[0]** ở trạng thái tắt.
- **c5:** Đèn LED sẽ sáng nếu **SW[5], KEY[1]** ở trạng thái bật và **KEY[0], KEY[2]** ở trạng thái tắt.
- **c6:** Đèn LED sẽ sáng nếu **SW[6], KEY[2]** ở trạng thái bật và **KEY[0], KEY[1]** ở trạng thái tắt.

- **c7:** Đèn LED sẽ sáng nếu **SW[7]** ở trạng thái bật và **KEY[0], KEY[1], KEY[2]** đều ở trạng thái tắt.

1.1Sơ Đồ Khối

Bộ ghép kênh là một mạch tổ hợp có tối đa 2^n đầu vào dữ liệu, 'n' đường chọn lọc với một đầu ra duy nhất. Một trong những đầu vào dữ liệu này sẽ được kết nối với đầu ra Y dựa trên các giá trị của các đường chọn lọc. Bộ ghép kênh 8 X 1 có 8 đầu vào dữ liệu D0, D1, D2, D3, D4, D5, D6 và D7, 3 đường chọn lọc S0, S1 và S2 và một đầu ra Y.

Sơ đồ khối:



Hình 1.1: Sơ đồ khối của Multiplexer 8:1

Chức năng của Multiplexer:

- Chọn Đầu Vào: Sử dụng ba đầu vào điều khiển (S0, S1, và S2), mạch có thể chọn một trong tám đầu vào (A0 đến A7) để xuất tại đầu ra Y.
- Đơn Giản Hóa Thiết Kế Mạch: Thay vì dùng nhiều cổng logic để điều khiển từng đầu ra, multiplexer cho phép đơn giản hóa thiết kế bằng cách sử dụng chỉ một đường ra.
- Tối Ưu Hóa Đường Truyền: Trong các hệ thống có nhiều tín hiệu cần được truyền đi mà không cần truyền tất cả cùng một lúc, multiplexer cho phép chúng ta chọn một tín hiệu cần thiết ở một thời điểm nhất định.

Trong mô hình FPGA, Multiplexer này có thể được lập trình để kiểm soát việc chọn lựa các tín hiệu được gửi đến các bộ phận khác của mạch, hoặc để đơn giản hóa việc routing tín hiệu trong một thiết kế phức tạp.

1.2 Hoạt Động

Bảng chân trị cho mạch này (truth table) sẽ như sau:

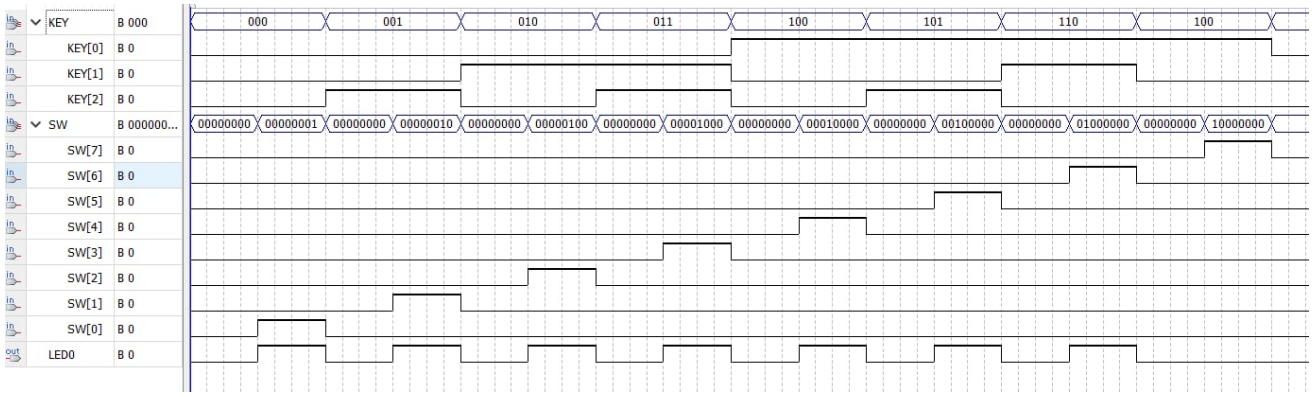
INPUTS			OUTPUT
S2	S1	S0	Y
0	0	0	A0
0	0	1	A1
0	1	0	A2
0	1	1	A3
1	0	0	A4
1	0	1	A5
1	1	0	A6
1	1	1	A7

Biểu thức logic của số hạng Y như sau:

Y=S0'.S1'.S2'.A0+S0.S1'.S2'.A1+S0'.S1.S2'.A2+S0.S1.S2'.A3+S0'.S1'.S2A4+S0.S1'.S2 A5+S0'.S1.S2 .A6+S0.S1.S3.A7

1.3 Phần Mềm Sử Dụng

- MODELSIM: Để lập trình FPGA bằng ngôn ngữ Verilog, chúng tôi đã sử dụng phần mềm Modelsim của Mentor Graphics. Đây là một môi trường đa ngôn ngữ cho phép mô phỏng các ngôn ngữ mô tả phần cứng như VHDL, Verilog và SystemC, bao gồm cả trình gỡ lỗi C tích hợp.
- Phần mềm thay thế như EDA PLAYGROUND có thể được sử dụng cho mô phỏng. Đây là phần mềm mã nguồn mở, ứng dụng web miễn phí cho phép người dùng chỉnh sửa, mô phỏng, tổng *hợp và chia sẻ mã HDL. Mục đích là tăng tốc quá trình học thiết kế và phát triển bảng kiểm tra, hỗ trợ nhiều HDL như System Verilog, VHDL, MyHDL và Migen.*



Hình 1.2: Dạng sóng của mạch tổ hợp

CHƯƠNG 2: THIẾT KẾ MẠCH DỊCH LED

Sử dụng ngôn ngữ Verilog HDL **thiết kế mạch dịch led** sử dụng 10 LED đỏ, và được **điều khiển bởi SW0 và KEY0.** Trong đó:

- LED chạy theo hướng được điều khiển bởi SW0 (tự quy định), khi LED dịch đến cạnh thì quay trở lại từ LED đầu tiên
- LED được dịch với tần số tùy chọn
- Khi nhấn KEY0, led sẽ đứng yên tại chỗ cho đến khi KEY0 được nhấn lần nữa.

Yêu cầu và gợi ý:

- Động tác nhấn KEY0 được tính là nhấn rồi thả, không phải nhấn và giữ
- Có thể thêm tín hiệu reset (tùy chọn/không bắt buộc)

Phần Code:

```
module Cau2(
    input CLOCK_50,
    input [1:0] SW,
    input [1:0] KEY,
    output reg [9:0] LEDR
);

    reg [24:0] counter;
    reg flag = 1'b1;

    always @(posedge CLOCK_50)
        counter <= counter + 1'b1;


    always @(negedge KEY[0])
    begin

                        flag = ~flag;

    end

    always @(posedge counter[24])
    begin
```

```
if(flag) begin
                    if(LEDR == 10'b0)
                        LEDR <= 10'b1;
            else if(SW[0])
                LEDR <= {LEDR[0], LE0DR[9:1]};
            else
                LEDR <= {LEDR[8:0], LEDR[9]};
        end
    end
endmodule
```

Các ngõ vào:

- **CLOCK_50**: Đây là tín hiệu đồng hồ chính, thường là 50 MHz, dùng để điều khiển hoạt động của bộ đếm và các logic khác trong mạch.
- **SW[1:0]:** Hai công tắc (switches) có thể được dùng để điều khiển hướng của việc dịch chuyển các LED.
- **KEY[1:0]:** Hai nút nhấn, trong đó **KEY[0]** được dùng để chuyển đổi trạng thái bật tắt (toggle) của LED (qua biến flag).

Ngõ ra:

- **LEDR[9:0]:** Mảng 10 đèn LED, được điều khiển bởi mạch để hiển thị các trạng thái nhấp nháy hoặc dịch chuyển.

Các biến tạm thời:

- **counter:** Bộ đếm 25-bit, dùng để tạo thời gian trễ dựa trên tín hiệu đồng hồ.

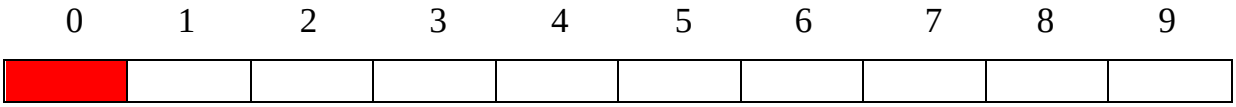
- **flag:** Biến này dùng để kiểm soát có cho phép dịch chuyển hoặc nhấp nháy LED hay không. Nó được đảo giá trị khi có một cạnh xuống của KEY[0].

Cách Hoạt Động:

1. Bộ đếm (Counter): Bộ đếm tăng liên tục với mỗi xung đồng hồ từ CLOCK_50. Khi bộ đếm này tràn (đạt giá trị cao nhất và quay trở lại 0), bit cao nhất (counter[24]) sẽ phát sinh một xung dương, sử dụng điều này để kích hoạt các sự kiện tiếp theo.
2. Chuyển Đổi Trạng Thái (Toggle State): Khi nút KEY[0] được thả ra (cạnh xuống), biến flag được đảo ngược. Nếu flag là 1, các LED sẽ hoạt động; nếu là 0, các LED sẽ không thay đổi trạng thái cho đến khi flag trở lại 1.
3. Điều khiển LED:

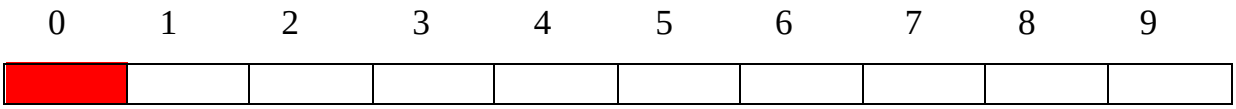
- Khi **flag** là 1 và có xung dương từ **counter[24],** tùy vào trạng thái của công tắc SW[0], mảng LED sẽ dịch trái hoặc dịch phải.
- Nếu LEDR đang ở trạng thái tắt hoàn toàn (giá trị 0), nó sẽ chuyển thành giá trị bật đầu tiên (10'b1).
- Nếu **SW[0]** = 1, LED sẽ dịch **sang trái.**
- Nếu **SW[0]** = 0, LED sẽ dịch **sang phải**

Trạng thái LED ban đầu:

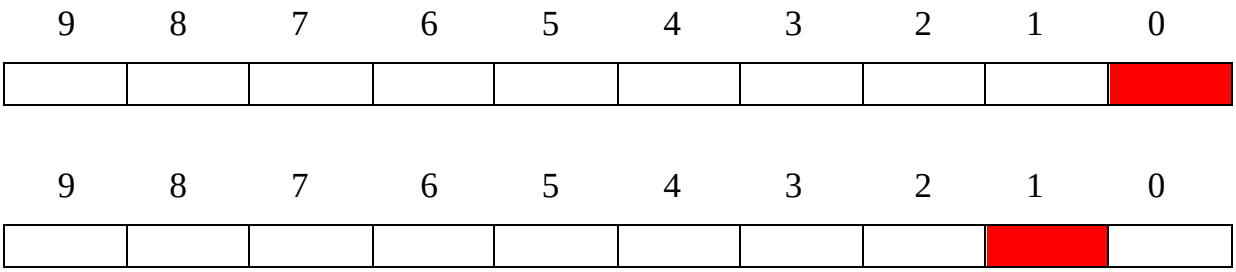


Trạng thái khi SW và KEY thay đổi:

Khi KEY[0] nhấn (toggle): Biến flag sẽ đảo ngược, cho phép hoặc ngăn cản sự thay đổi của LED.

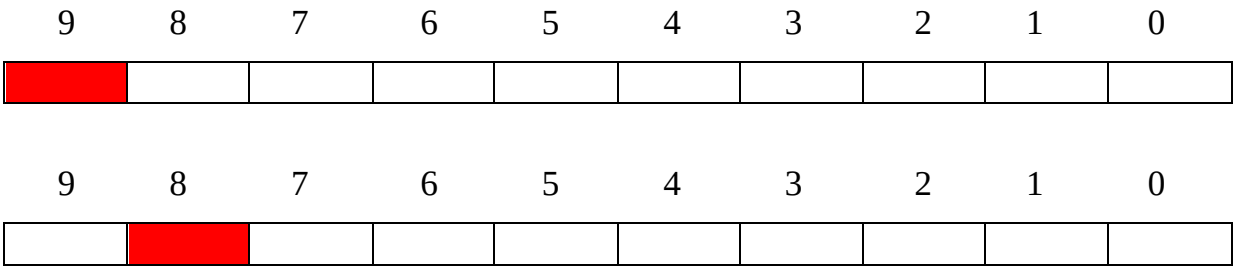


Khi SW[0] gạt lên (1) và flag = 1: LED dịch chuyển sang trái



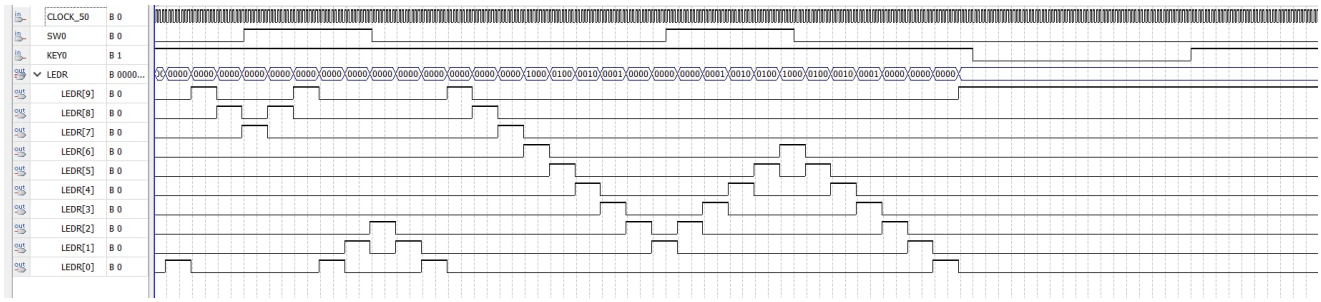
.....

Khi SW[0] gạt xuống (0) và flag = 1: LED sẽ dịch chuyển sang phải.



. . . . . .

2.1 Mô Phỏng Dạng Sóng



Hình 1.3: Dạng sóng của mạch tổ hợp mạch dịch led

CHƯƠNG 3:THIẾT KẾ ĐỒNG HỒ ĐẾM THỜI GIAN

Thiết kế mạch đồng hồ đếm thời gian gồm phút và giây hiển thị lên HEX3, HEX2, HEX1, HEX0 và được điều khiển bởi các KEY trên board FPGA DE10, sử dụng ngôn ngữ Verilog HDL. Trong đó:

- **KEY0** dùng để reset đồng hồ về "00:00"
- **KEY1** dùng để bắt đầu hoặc tạm dừng đếm thời gian (đồng hồ tiếp tục đếm khi KEY1 được nhấn lần nữa)

Yêu cầu và gợi ý:

- Dùng một bộ đếm và thực hiện so sánh để tạo xung clock chính xác 1Hz từ clock 50MHz (để tạo ra 1 giây thời gian thực chính xác nhất); bộ đếm có thể được thiết kế riêng thành module riêng và gọi vào thiết kế (instantiate)
- Các chức năng giải mã hiển thị ra HEX được thiết kế thành module riêng

Phần Code

```
module Cau3(input CLOCK_50, input [1:0] KEY, output [0:6] HEX0, output [0:6] HEX1, output [0:6] HEX2, output [0:6] HEX3);

reg [25:0] counter, led_time;

reg [3:0] giay_donvi, giay_chuc, phut_donvi, phut_chuc;

reg flag, KEY0, KEY1;

initial led_time = 26'd50000000;

    always @(posedge CLOCK_50)

    begin

        KEY0 <= KEY[0];

        KEY1 <= KEY[1];

        if (KEY0 && ~KEY[0])
```

```
                                    phut_chuc <= phut_chuc +
                               1'b1;
                                    phut_donvi <= 4'b0;
                               end


                               else
                                    phut_donvi <= phut_donvi +
1'b1;
                               giay_chuc <= 4'b0;
                          end


                          else
                               giay_chuc <= giay_chuc + 1'b1;


                               giay_donvi <= 4'b0;
                     end


                     else
                          giay_donvi <= giay_donvi + 1'b1;
               end
          end
          else
               counter <= counter + 1'b1;
     end
     HEX_display a0(.a(giay_donvi), .b(HEX0));
     HEX_display a1(.a(giay_chuc), .b(HEX1));
     HEX_display a2(.a(phut_donvi), .b(HEX2));
     HEX_display a3(.a(phut_chuc), .b(HEX3));
endmodule
```

Module phụ trong đoạn mã Verilog mà bạn cung cấp là **HEX_display**. Đây là module được sử dụng để giải mã giá trị số (thập phân) từ đầu vào 4-bit a và biểu diễn nó dưới dạng mã đèn LED 7 đoạn trên đầu ra 7-bit b. Mỗi bit của b tương ứng với một segmen của một đèn LED 7 đoạn.

```
module HEX_display(input [3:0] a, output reg [0:6] b);
     always @(*)
     begin
          b = (a == 4'b0000) ? 7'b0000001 : // 0
                    (a == 4'b0001) ? 7'b1001111 : // 1
                    (a == 4'b0010) ? 7'b0010010 : // 2
                    (a == 4'b0011) ? 7'b0000110 : // 3
                    (a == 4'b0100) ? 7'b1001100 : // 4
                    (a == 4'b0101) ? 7'b0100100 : // 5
                    (a == 4'b0110) ? 7'b0100000 : // 6
                    (a == 4'b0111) ? 7'b0001111 : // 7
                    (a == 4'b1000) ? 7'b0000000 : // 8
                    (a == 4'b1001) ? 7'b0000100 : // 9
                    (a == 4'b1010) ? 7'b0001000 : // A
                    (a == 4'b1011) ? 7'b1100000 : // B
                    (a == 4'b1100) ? 7'b0110001 : // C
                    (a == 4'b1101) ? 7'b1000010 : // D
                    (a == 4'b1110) ? 7'b0110000 : // E
                   7'b0111000;  // F
     end
endmodule
```

Khi chưa nhấn nút:

- counter sẽ tăng với mỗi xung đồng hồ CLOCK_50.
- Nếu không nhấn bất kỳ nút nào (KEY[0] hoặc KEY[1]), counter tiếp tục tăng cho đến khi nó đạt giá trị led_time (giả sử 50 triệu để đại diện cho một giây nếu CLOCK_50 là 50MHz).
- Cờ flag vẫn giữ nguyên trạng thái, nghĩa là nếu nó được set là 1 từ trước, đồng hồ sẽ tiếp tục đếm; nếu là 0, đồng hồ dừng lại.

```
always @(posedge CLOCK_50) begin
    // Counter tăng liên tục để đếm thời gian
    if (counter < led_time) begin
        counter <= counter + 1'b1;
    end
    // Các hoạt động khác liên quan đến KEY và flag
end
```

Khi nhấn nút KEY[0]:

Khi nút KEY[0] được nhấn, một sự kiện cạnh xuống được phát hiện (KEY0 và ~KEY[0]), điều này làm reset các giá trị thời gian giây và phút về 0.

```
// Khi nhấn nút KEY[0]:
always @(posedge CLOCK_50) begin
    KEY0 <= KEY[0];  // Lưu trạng thái trước của KEY[0]
    if (KEY0 && ~KEY[0]) begin  // Phát hiện cạnh xuống
        // Reset thời gian khi nút KEY[0] được nhấn
```

```
giay_donvi <= 4'b0;

        giay_chuc <= 4'b0;

        phut_donvi <= 4'b0;

        phut_chuc <= 4'b0;

    end
```

Khi thả nút KEY[0]:

Sau khi thả nút KEY[0], không có sự kiện cạnh xuống mới nào, vì vậy không có thay đổi gì xảy ra cho đến khi nút được nhấn lại.

Khi nhấn nút KEY[1]:

Khi nút KEY[1] được nhấn, cũng phát hiện một sự kiện cạnh xuống (KEY1 và ~KEY[1]), nó sẽ toggle trạng thái của flag.

Nếu flag là 1 (đồng hồ đang chạy), nó sẽ chuyển thành 0 và ngược lại. Điều này cho phép bắt đầu hoặc dừng đồng hồ.

```
// Khi nhấn nút KEY[1]:

always @(posedge CLOCK_50) begin

    KEY1 <= KEY[1];  // Lưu trạng thái trước của
KEY[1]

    if (KEY1 && ~KEY[1]) begin  // Phát hiện cạnh
xuống

        flag <= ~flag;  // Toggle trạng thái của flag

    end
```

Khi thả nút KEY[1]:

Thả nút KEY[1] không gây ra bất kỳ sự kiện quan trọng nào cho đến khi nút được nhấn lại.

Khi nhấn và thả nút KEY[1] lần nữa:

Quá trình giống như khi nhấn và thả lần đầu tiên, flag sẽ được toggle lại.

Tạo xung chính xác 1 giây:

$$f = \frac{1}{T} = \frac{1}{1} = 1$$

Với f : Tần số (Hz)

T : Chu kì (s)

led_time được set để phù hợp với chu kỳ của đồng hồ 50MHz. Mỗi giây, có 50 triệu xung đồng hồ, nên led_time phải đếm đến 50 triệu trước khi reset counter và tăng thời gian giây/phút.

Công thức: led_time = Tần số của CLOCK_50 x Chu kỳ mong muốn (1 giây).

Ví dụ, nếu CLOCK_50 là 50MHz, thì led_time = 50MHz x 1 giây = 50,000,000.

Module HEX_display:

Module này không phụ thuộc vào trạng thái của nút nhấn. Nó đơn giản chỉ là một module giải mã nhận vào một số thập phân 4-bit và chuyển đổi nó thành mã tương ứng cho đèn LED 7 đoạn để hiển thị. Đối với mỗi đầu vào từ 0 đến 9 (và A đến F nếu cần), module sẽ xuất ra mã đèn LED 7 đoạn tương ứng. Mỗi bit 0 trong giá trị 7-bit của b sẽ bật segmen tương ứng của đèn LED, và bit 1 sẽ tắt segmen đó.

3.1 Thực hành trực tiếp trên mạch

https://drive.google.com/drive/folders/11YUPynWgATWBbdBW31ySELs2HTJvEYVS?usp=drive_link