ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH

Môn học: THIẾT KẾ LUẬN LÝ

Lớp : L02

TOPIC : RUNNING LED

GVHD : Huỳnh Hoàng Kha

Group : 7

Sinh viên thực hiện: Nguyễn Thành Chương – 1912800

Bùi Đăng Khoa – 2013489

Lê Thanh Dương – 2012883

MỤC LỤC

I. Phân tích đề tài

➢ **Nhận diện và mô tả các yêu cầu của hệ thống, phân tích các vấn đề cần giải quyết.**

✓ Về thiết kế hệ thống:

- Sử dụng tất cả các LED đơn (18 LED đỏ + 8 LED xanh) để trình diễn các hiệu ứng LED.
- Sử dụng các Switch, các Button trên board Altera DE2i-150 để thay đổi, điều chỉnh hệ thống.
- Sử dụng các LED 7-Segment để hiển thị các thông tin của hệ thống.

✓ Về chức năng hệ thống

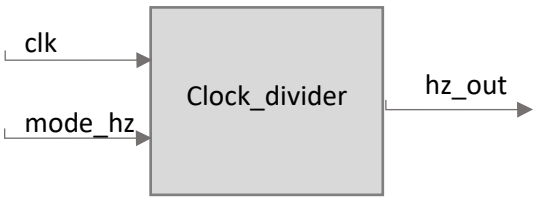
- Hệ thống phải hỗ trợ ít nhất 10 hiệu ứng LED.
- Hệ thống phải hỗ trợ 4 mức tần số để điều chỉnh tốc độ điều khiển hiệu ứng LED: 0.5Hz, 1Hz, 2Hz, và 4Hz
- Hệ thống phải hỗ trợ hiển thị thông tin về hiệu ứng, tần số đang sử dụng trên các LED 7-Segment.
- Hệ thống phải hỗ trợ hai chế độ điều khiển: điều khiển tự động (cho phép các hiệu ứng LED tự động thay đổi) và điều khiển bằng tay (thay đổi hiệu ứng LED bằng tay).

➢ **Sinh viên đề xuất hướng giải quyết cho từng vấn đề.**

- Hệ thống phải hỗ trợ ít nhất 10 hiệu ứng LED: Mỗi hiệu ứng là một case và dùng multiplexer để lựa chọn các hiệu ứng.
- Hệ thống phải hỗ trợ 4 mức tần số để điều chỉnh tốc độ điều khiển hiệu ứng LED: 0.5Hz, 1Hz, 2Hz, và 4Hz: Dùng một module con để chuyển đổi tần số của Board và dùng multiplexer để chọn các tần số đầu ra theo yêu cầu.
- Hệ thống phải hỗ trợ hiển thị thông tin về hiệu ứng, tần số đang sử dụng trên các LED 7-Segment: Dùng một module con để thực hiện giải mã Led 7 đoạn, và dùng các register thích hợp để đưa vào input của module giải mã Led 7 đoạn hiện ra theo yêu cầu.
- Hệ thống phải hỗ trợ hai chế độ điều khiển: điều khiển tự động (cho phép các hiệu ứng LED tự động thay đổi) và điều khiển bằng tay (thay đổi hiệu ứng LED bằng tay): Ở mạch multiplexer chọn hiệu ứng, dùng một register làm biến select, khi điều khiển tự động thì register này sẽ lưu trữ giá trị của tự động thay đổi còn khi điều khiển bằng tay thì sẽ lưu trữ các giá trị nhập vào bằng tay.

II. Thiết kế và hiện thực

1. Module chia tần số (clock_divider):



❖ Ý tưởng:

− Tần số của Board là 50MHz, để chuyển đổi ra các tần số theo yêu cầu thì cần dùng một bộ đếm để thực hiện đếm theo yêu cầu của tần số đầu ra. Ví dụ: Yêu cầu tần số đầu ra là 1 Hz thì ta thực hiện đếm từ 0 – 24,999,999. Mỗi khi đạt giá trị 24,999,999 thì reset lại cho bộ đếm đếm lại từ đầu và khi đó ta đảo ngược trạng thái của hz_out. Tương tự ta có công thức tổng quát cho giá trị cực đại MAX_nHz của bộ đếm ở tần số n Hz là:

$$MAX\_nHz = \frac{50{,}000{,}000}{2n} - 1$$

− Do module có thể có nhiều mức tần số đầu ra nên ta dùng một multiplexer để lựa chọn MAX_nHz theo cầu của tần số đầu ra.

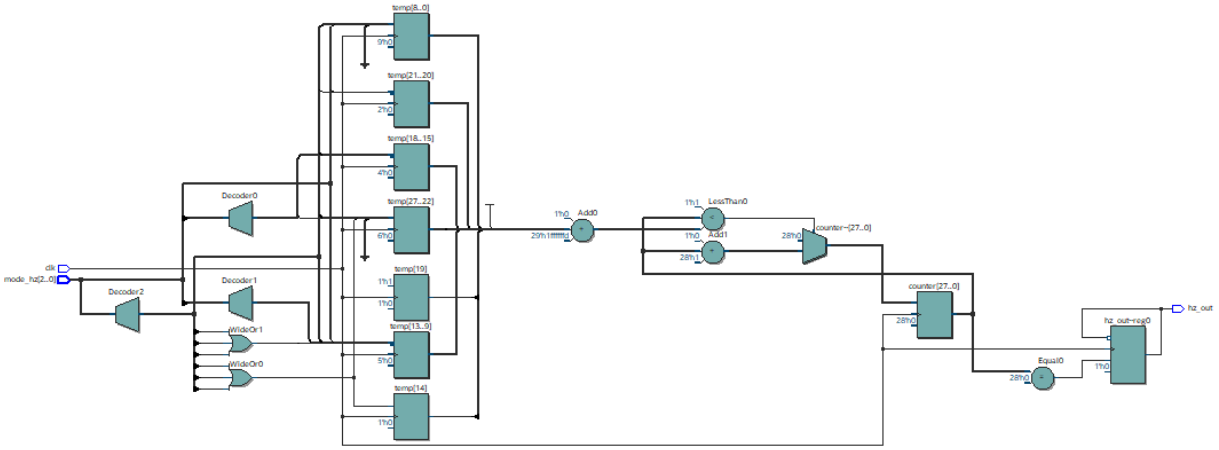
mode_hz	MAX_nHZ	Tần số đầu ra
000	49,999,999	0.5 Hz
001	24,999,999	1 Hz
010	12,499,999	2 Hz
011	6,249,999	4 Hz
100	3,124,999	8 Hz
default	3,124,999	8 Hz

❖ **Hiện thực:**

− Verilog code:

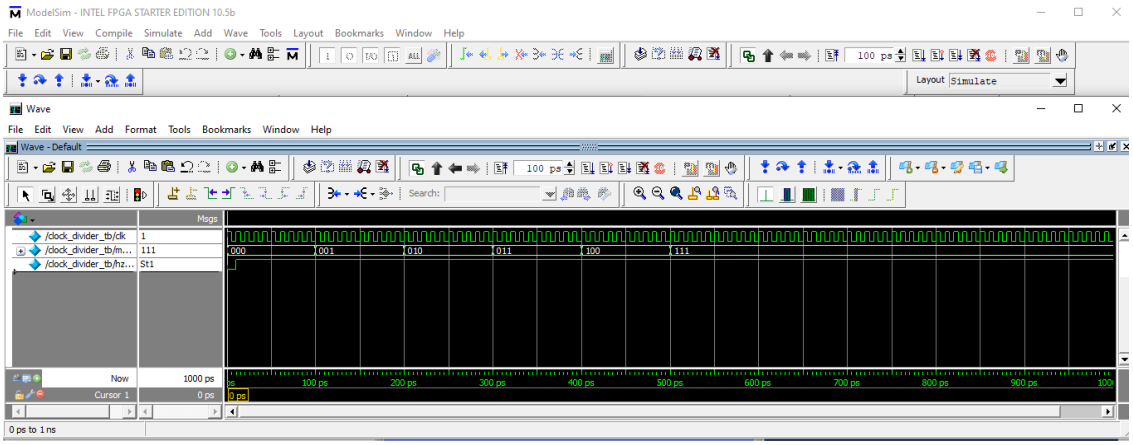
```
module clock_divider(clk, mode_hz, hz_out);
input clk;
input [2:0] mode_hz;
output reg hz_out=1'b0;
reg [27:0] counter;
reg [27:0] temp;
always @(posedge clk) begin
case(mode_hz)
3'b0: begin
temp<=28'd50_000_000*28'd2/28'd2;
if(counter<=temp-28'd1) counter<=counter+28'd1;
else counter<=28'b0;
if(counter==28'd0) hz_out<=~hz_out;
end
3'b001: begin
temp<=28'd50_000_000/28'd2;
if(counter<=temp-28'd1) counter<=counter+28'd1;
else counter<=28'd0;
if(counter==28'd0) hz_out<=~hz_out;
end
3'b010: begin
temp<=28'd50_000_000/28'd2/28'd2;
if(counter<=temp-28'd1) counter<=counter+28'd1;
else counter<=28'd0;
if(counter==28'd0) hz_out<=~hz_out;
end
3'b011: begin
temp<=28'd50_000_000/28'd4/28'd2;
if(counter<=temp-28'd1) counter<=counter+28'd1;
else counter<=28'd0;
if(counter==28'd0) hz_out<=~hz_out;
end
3'b100: begin
temp<=28'd50_000_000/28'd8/28'd2;
if(counter<=temp-28'd1) counter<=counter+28'd1;
else counter<=28'd0;
if(counter==28'd0) hz_out<=~hz_out;
end
default: begin
```

```
temp<=28'd50_000_000/28'd8/28'd2;
if(counter<=temp-28'd1) counter<=counter+28'd1;
else counter<=28'd0;
if(counter==28'd0) hz_out<=~hz_out;
end
endcase
end
endmodule
```

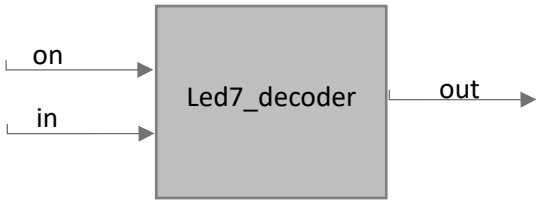


clock_divider.v | clock_divider_tb.v | Compilation Report - BTL

```
1  module clock_divider_tb;
2
3  reg clk;
4  reg [2:0] mode_hz;
5  wire hz_out;
6
7  clock_divider UUT(.clk(clk), .mode_hz(mode_hz), .hz_out(hz_out));
8  always
9  begin
10   clk = 1'b1;
11   #5;
12   clk = 1'b0;
13   #5;
14 end
15
16 initial begin
17   mode_hz = 3'b000;
18   #100;
19   mode_hz = 3'b001;
20   #100;
21   mode_hz = 3'b010;
22   #100;
23   mode_hz = 3'b011;
24   #100;
25   mode_hz = 3'b100;
26   #100;
27   mode_hz = 3'b111;
28 end
29 endmodule
```



2. Module giải mã Led 7 đoạn (led7_decoder):



❖ Ý tưởng:

– Tham số hóa các trạng thái của Led 7 đoạn.

Tham số	Ý nghĩa	Giá trị
zero	Đại diện cho giá trị để Led hiện số 0	7'b1111110
one	Đại diện cho giá trị để Led hiện số 1	7'b0110000
two	Đại diện cho giá trị để Led hiện số 2	7'b1101101
three	Đại diện cho giá trị để Led hiện số 3	7'b1111001
four	Đại diện cho giá trị để Led hiện số 4	7'b0110011
five	Đại diện cho giá trị để Led hiện số 5	7'b1011011
six	Đại diện cho giá trị để Led hiện số 6	7'b1011111
seven	Đại diện cho giá trị để Led hiện số 7	7'b1110000
eight	Đại diện cho giá trị để Led hiện số 8	7'b1111111
nine	Đại diện cho giá trị để Led hiện số 9	7'b1111011
c	Đại diện cho giá trị để Led hiện chữ C	7'b1001111
e	Đại diện cho giá trị để Led hiện chữ E	7'b1001110

– Dùng input on để bật, tắt Led.

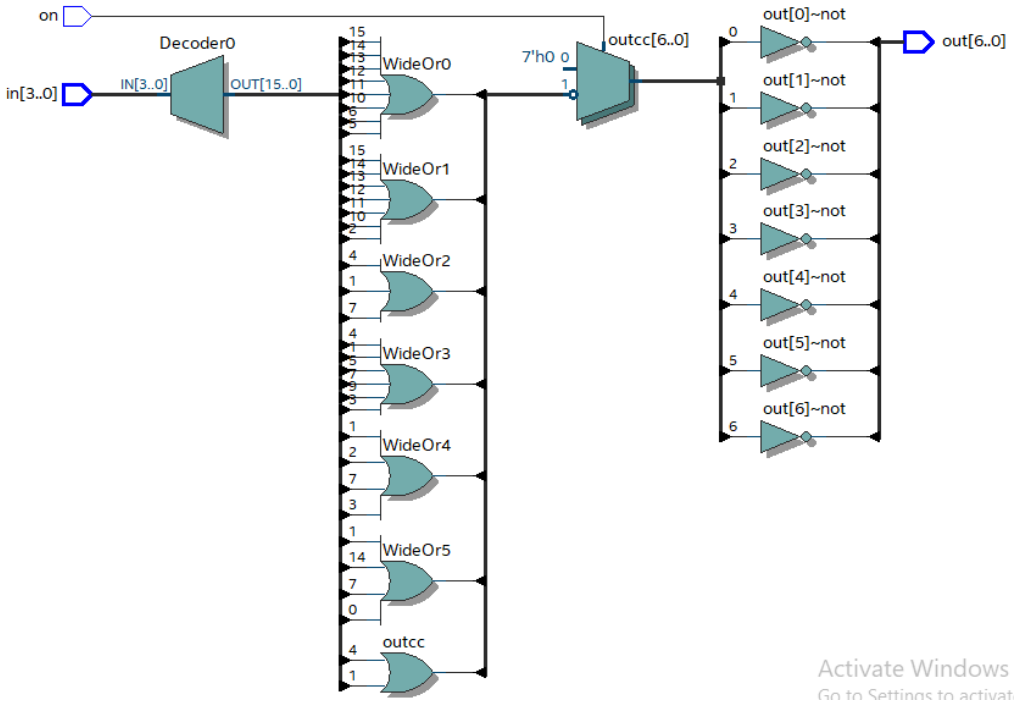
– Dùng multiplexer để ứng với các input in có các trạng thái Led 7 đoạn tương ứng.

in	out
0000	zero
0001	one
0010	two
0011	three
0100	four
0101	five
0110	six
0111	seven
1000	eight
1001	nine
1110	c
1111	e
default	e

❖ **Hiện thực:**

```
module led7_decoder(on, in, out);
input [3:0] in;
input on;
reg [6:0] outcc;
output [6:0] out;
parameter zero=7'b1111110,
one=7'b0110000,
two=7'b1101101,
three=7'b1111001,
four=7'b0110011,
five=7'b1011011,
six=7'b1011111,
seven=7'b1110000,
eight=7'b1111111,
nine=7'b1111011,
e=7'b1001111,
c=7'b1001110;
always @* begin
```

```
 if(on) begin
 case (in)
 4'b0000: outcc = zero ;
 4'b0001: outcc = one ;
 4'b0010: outcc = two ;
 4'b0011: outcc = three ;
 4'b0100: outcc = four ;
 4'b0101: outcc = five ;
 4'b0110: outcc = six ;
 4'b0111: outcc = seven ;
 4'b1000: outcc = eight ;
 4'b1001: outcc = nine ;
 4'b1110: outcc = c ;
4'b1111: outcc = e ;
 default outcc = e ; //E: error
 endcase
end
else outcc=7'b0;
 end
 not a1[6:0] (out,outcc);
 endmodule
```

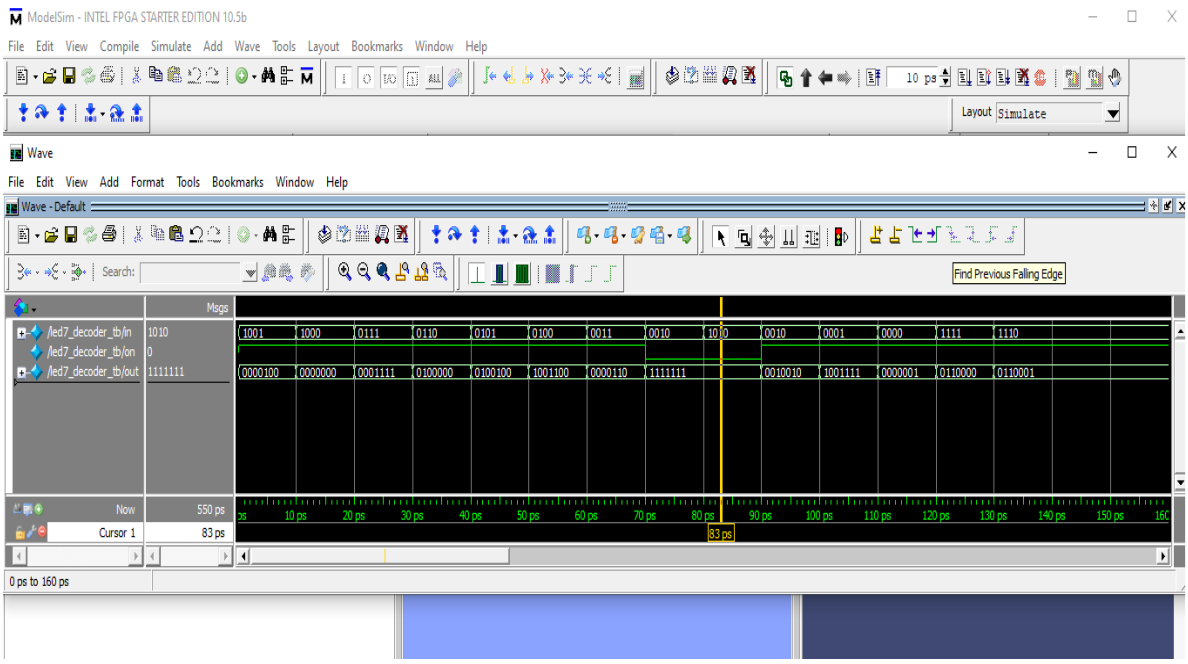


RUNNING LED

```
module led7_decoder_tb;

reg [3:0] in;
reg on;
wire [6:0] out;

led7_decoder UUT(.on(on), .in(in), .out(out));
initial begin
on=1'b1;
in=4'b1001;
#10;
on=1'b1;
in=4'b1000;
#10;
on=1'b1;
in=4'b0111;
#10;
on=1'b1;
in=4'b0110;
#10;
on=1'b1;
in=4'b0101;
#10;
on=1'b1;
in=4'b0100;
#10;
on=1'b1;
in=4'b0011;
#10;
on=1'b0;
in=4'b0010;
#10;
on=1'b0;
in=4'b1010;
#10;
on=1'b1;
in=4'b0010;
#10;
on=1'b1;
in=4'b0001;
#10;
on=1'b1;
in=4'b0000;
#10;
on=1'b1;
in=4'b1111;
#10;
on=1'b1;
in=4'b1110;
#10;
end
endmodule
```



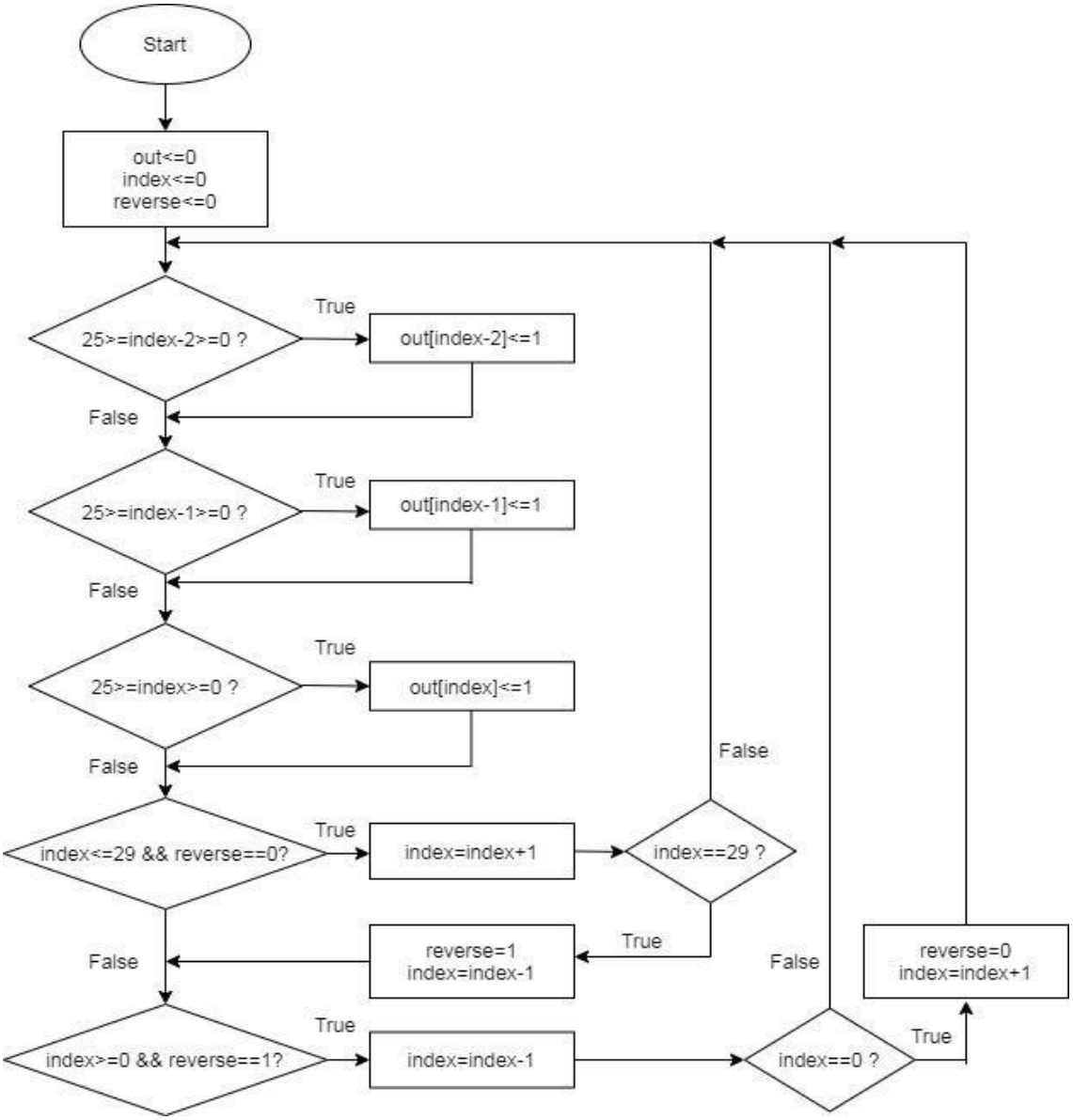
3. Các hiệu ứng:

3.1 **Hiệu ứng 1:** Cho chuỗi 3 LED sáng ở vị trí biên phải, sau đó cho chuỗi LED này di chuyển theo hướng từ phải qua trái. Khi chạm biên trái, chuỗi LED di chuyển ngược lại theo hướng trái qua phải. Quá trình trên được lặp đi lặp lại.

❖ **Ý tưởng:**

– Dùng thuật toán để Led hiển thị theo yêu cầu. –

Flow chart:



❖ **Hiện thực:**

– **Verilog code:**

```
for(i=6'd1; i<=6'd26; i=i+6'd1) begin
if (i==index || i==index-6'd1 || i==index-6'd2) out[i-6'd1]<=1'b1;
else out[i-6'd1]<=1'b0;
end
if(index<=6'd29 && reverse==1'b0) begin
index<=index+6'd1;
if(index==6'd29) begin
 reverse<=1'b1;
```

```
 index<=index-6'd1;
end
end
if(index>=6'd0 && reverse==1'b1) begin
index<=index-6'd1;
if(index==6'd0) begin
reverse<=1'b0;
index<=index+6'd1;
 // Ket thuc hieu ung gan auto_next=1
end
end
```

− **Mô phỏng trên modelSim:**

```
# 00000000000000000000000000
# 00000000000000000000000001
# 00000000000000000000000011
# 00000000000000000000000111
# 00000000000000000000001110
# 00000000000000000000011100
# 00000000000000000000111000
# 00000000000000000001110000
# 00000000000000000011100000
# 00000000000000000111000000
# 00000000000000001110000000
# 00000000000000011100000000
# 00000000000000111000000000
# 00000000000001110000000000
```

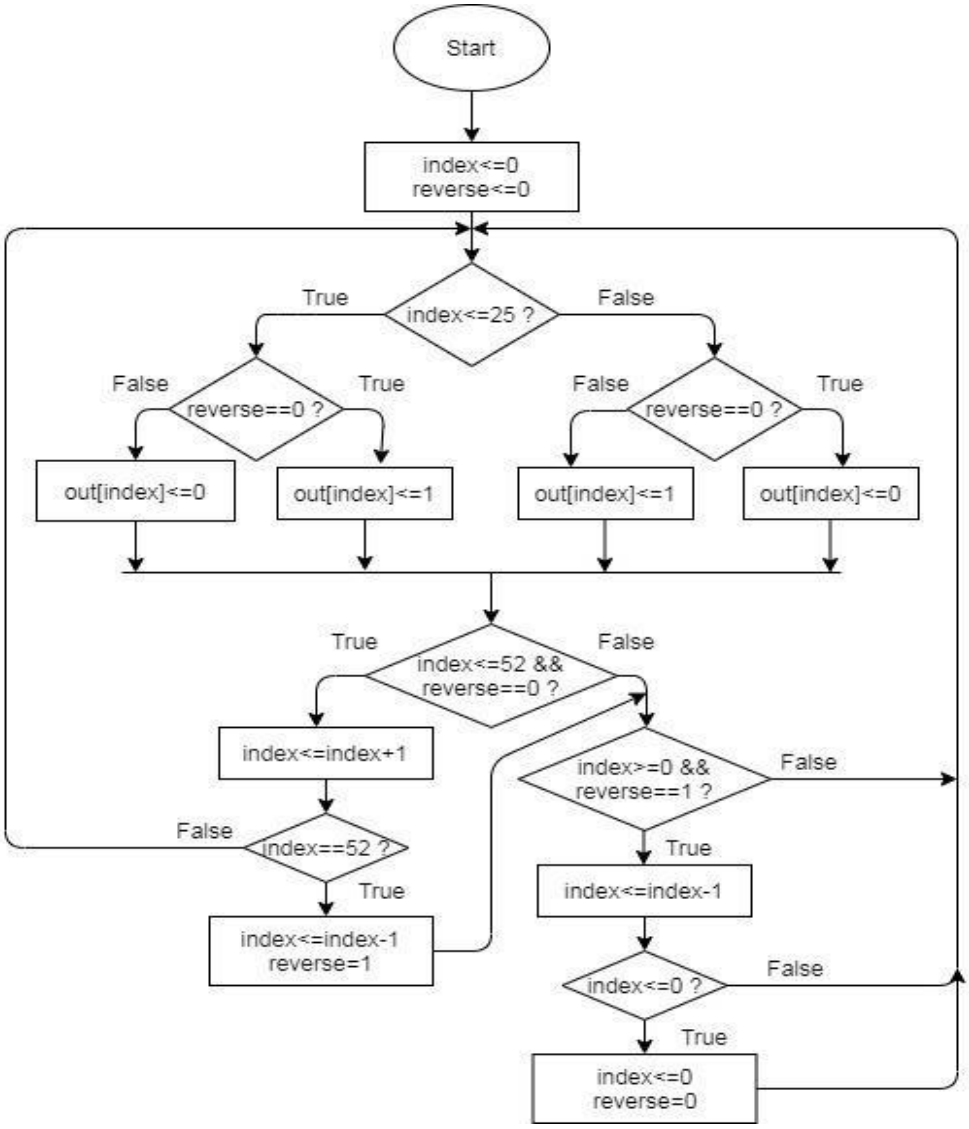
```
# 00000000000011100000000000
# 00000000000111000000000000
# 00000000001110000000000000
# 00000000011100000000000000
# 00000001110000000000000000
# 00000011100000000000000000
# 00000111000000000000000000
# 00001110000000000000000000
# 00011100000000000000000000
# 00111000000000000000000000
# 01110000000000000000000000
# 11100000000000000000000000
# 11000000000000000000000000
```

```
# 10000000000000000000000000
# 00000000000000000000000000
# 10000000000000000000000000
# 11000000000000000000000000
# 11100000000000000000000000
# 01110000000000000000000000
# 00111000000000000000000000
# 00011100000000000000000000
# 00001110000000000000000000
# 00000111000000000000000000
# 00000011100000000000000000
# 00000001110000000000000000
# 00000000111000000000000000
# 00000000011100000000000000
# 00000000001110000000000000
# 00000000000111000000000000
# 00000000000011100000000000
# 00000000000001110000000000
# 00000000000000111000000000
# 00000000000000011100000000
# 00000000000000001110000000
# 00000000000000000111000000
# 00000000000000000011100000
# 00000000000000000001110000
# 00000000000000000000111000
# 00000000000000000000011100
# 00000000000000000000001110
# 00000000000000000000000111
# 00000000000000000000000011
# 00000000000000000000000001
# 00000000000000000000000000
```

3.2 **Hiệu ứng 2:** Cho LED sáng lần lượt từng cái từ phải qua trái. Khi chạm biên trái, LED đổi chiều sáng lần lượt từ trái qua phải. Quá trình trên được lặp đi lặp lại.

❖ **Ý tưởng:**

- Dùng thuật toán để Led hiển thị theo yêu cầu.
- Flow chart:



❖ **Hiện thực:**

– **Verilog code:**

```
if(index<=6'd25) begin
if(reverse==1'b0) begin
out[index]<=1'b1;
end
else begin
out[index]<=1'b0;
end
end
```

```
if(index>6'd25) begin
if(reverse==1'b0) begin
out[index-6'd26]<=1'b0;
end
else begin
out[index-6'd26]<=1'b1;
end
end
if(index<=6'd52 && reverse==1'b0) begin
index<=index+6'd1;
if(index==6'd52) begin
reverse<=1'b1;
index<=index-6'd1;
end
end
if(index>=6'd0 && reverse==1'b1) begin
index<=index-6'd1;
if(index<=6'd0) begin
reverse<=1'b0;
index<=6'd0;
// Ket thuc hieu ung gan auto_next=2
end
end
```

− **Mô phỏng trên ModelSim:**

#00000000000000000000000000
#00000000000000000000000001
00000000000000000000000011
00000000000000000000000111
#00000000000000000000001111
#00000000000000000000011111
00000000000000000000111111
00000000000000000001111111
#00000000000000000011111111
#00000000000000000111111111
00000000000000001111111111
00000000000000011111111111
#00000000000000111111111111
#00000000000001111111111111
00000000000011111111111111
00000000000111111111111111
#00000000001111111111111111
#00000000011111111111111111

#00000000111111111111111111
#00000001111111111111111111
#00000011111111111111111111
#00000111111111111111111111
#00001111111111111111111111
#00011111111111111111111111
#00111111111111111111111111
#01111111111111111111111111
#11111111111111111111111111
#11111111111111111111111110
#11111111111111111111111100
#11111111111111111111111000
#11111111111111111111110000
#11111111111111111111100000
#11111111111111111111000000
#11111111111111111110000000
#11111111111111111100000000
#11111111111111111000000000

```
# 11111111111111110000000000
# 11111111111111100000000000
# 11111111111111000000000000
# 11111111111110000000000000
# 11111111111100000000000000
# 11111111111000000000000000
# 11111111110000000000000000
# 11111111100000000000000000
# 11111111000000000000000000
# 11111110000000000000000000
# 11111100000000000000000000
# 11111000000000000000000000
# 11110000000000000000000000
# 11100000000000000000000000
# 11000000000000000000000000
# 10000000000000000000000000
# 00000000000000000000000000
# 10000000000000000000000000
# 11000000000000000000000000
# 11100000000000000000000000
# 11110000000000000000000000
# 11111000000000000000000000
# 11111100000000000000000000
# 11111110000000000000000000
# 11111111000000000000000000
# 11111111100000000000000000
# 11111111110000000000000000
# 11111111111000000000000000
# 11111111111100000000000000
# 11111111111110000000000000
# 11111111111111000000000000
# 11111111111111100000000000
# 11111111111111110000000000
```

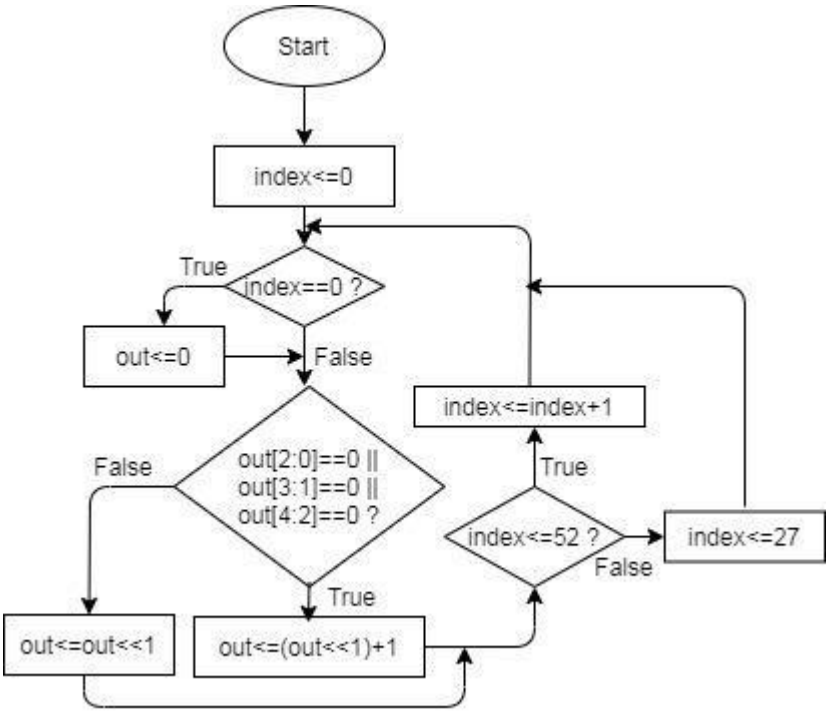
```
#11111111111111111110000000
#11111111111111111111000000
#11111111111111111111100000
#11111111111111111111110000
#11111111111111111111111000
#11111111111111111111111100
#11111111111111111111111110
#11111111111111111111111111
#01111111111111111111111111
#00111111111111111111111111
#00011111111111111111111111
#00001111111111111111111111
#00000111111111111111111111
#00000011111111111111111111
#00000001111111111111111111
#00000000111111111111111111
#00000000011111111111111111
#00000000001111111111111111
#00000000000111111111111111
#00000000000011111111111111
#00000000000001111111111111
#00000000000000111111111111
#00000000000000011111111111
#00000000000000001111111111
#00000000000000000111111111
#00000000000000000011111111
#00000000000000000001111111
#00000000000000000000111111
#00000000000000000000011111
#00000000000000000000001111
#00000000000000000000000111
#00000000000000000000000011
#00000000000000000000000001
#00000000000000000000000000
```

3.3 **Hiệu ứng 3:** Cho các chuỗi 3 LED sáng, hai chuỗi cách nhau bởi 3 LED tắt. Cho các chuỗi LED di chuyển vòng từ phải sáng trái.

❖ **Ý tưởng:**

− Dùng thuật toán để Led hiển thị theo yêu cầu. −

Flow chart:



❖ **Hiện thực:**

– **Verilog code:**

```
if(index==6'd0) out<=26'd0;

if(out[2:0]==3'd0 || out[3:1]==3'd0 ||
out[4:2]==3'd0)

out<=(out<<1)+26'd1;

else out<=out<<1;

if(index<=52) index<=index+6'd1;

else begin

index<=6'd27;

 // Ket thuc hieu ung gan auto_next=3

end
```

– **Mô phỏng trên ModelSim:**

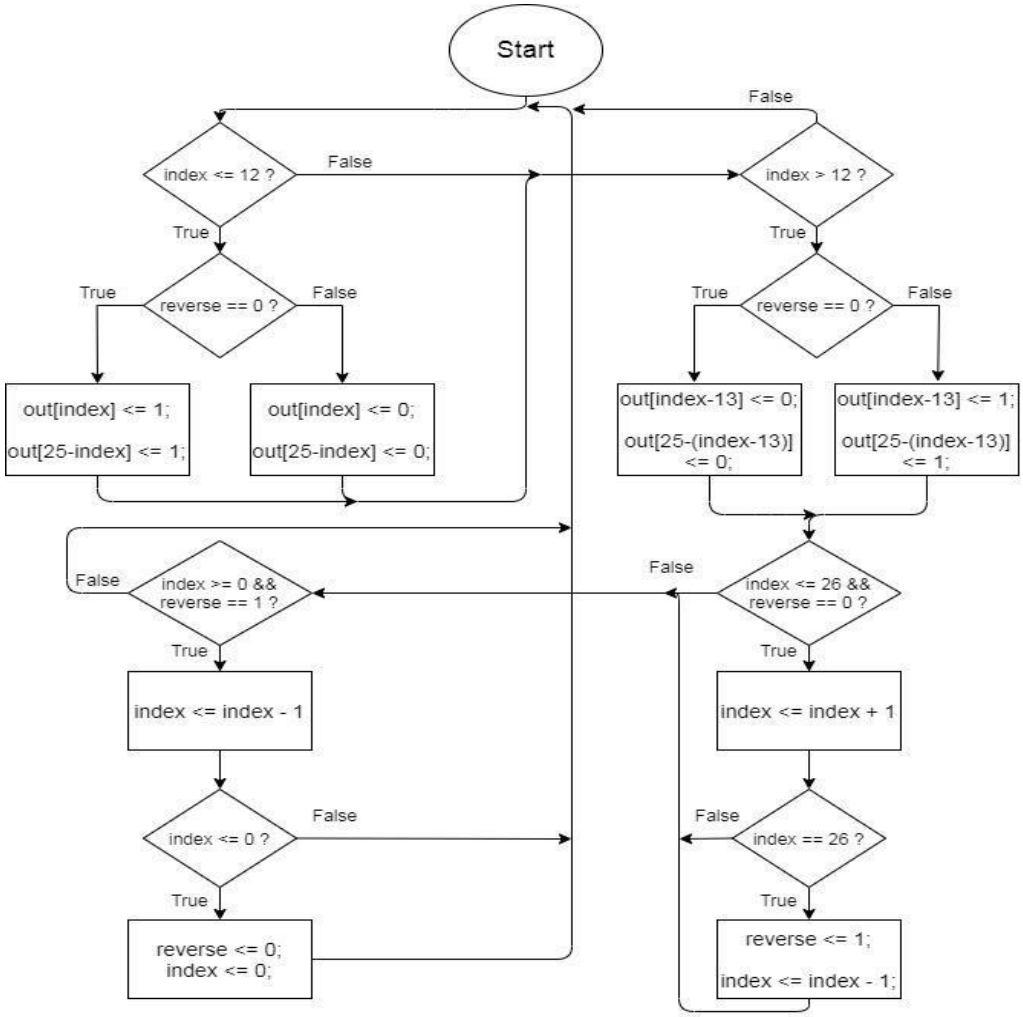
# 00000000000000000000000000	#00000000000000000000111000
# 00000000000000000000000001	#00000000000000000001110001
# 00000000000000000000000011	#00000000000000000011100011
# 00000000000000000000000111	#00000000000000000111000111
# 00000000000000000000001110	#00000000000000001110001110
# 00000000000000000000011100	#00000000000000011100011100
# 00000000000000111000111000	#11000111000111000111000111
# 00000000000001110001110001	#10001110001110001110001110
# 00000000000011100011100011	#00011100011100011100011100
# 00000000000111000111000111	#00111000111000111000111000
# 00000000001110001110001110	#01110001110001110001110001
# 00000000011100011100011100	#11100011100011100011100011
# 00000000111000111000111000	#11000111000111000111000111
# 00000001110001110001110001	#10001110001110001110001110
# 00000011100011100011100011	#00011100011100011100011100
# 00000111000111000111000111	#00111000111000111000111000
# 00001110001110001110001110	#01110001110001110001110001
# 00011100011100011100011100	#11100011100011100011100011
# 00111000111000111000111000	#11000111000111000111000111
# 01110001110001110001110001	#10001110001110001110001110
# 11100011100011100011100011	#00011100011100011100011100

3.4 **Hiệu ứng 4:** Cho LED sáng dần từ hai biên vào trong, khi tấc cả các LED sáng hết thì thì các LED tắt dần từ 2 biên vào trong, khi tấc cả các LED tắt hết thì các LED sẽ sáng lại từ trong ra ngoài, Khi tấc cả các LED sáng hết thì các LED sẽ tắt dần từ trong ra ngoài đến khi tắt hết thì quá trình lại bắt đầu lại từ đầu.

❖ **Ý tưởng:**

− Dùng thuật toán để Led hiển thị theo yêu cầu.

− Flow chart:



❖ **Hiện thực:**

– **Verilog code:**

```
 if(index<=6'd12) begin
if(reverse==1'b0) begin
 out[index]<=1'b1;
 out[6'd25-index]<=1'b1;
end
else begin
 out[index]<=1'b0;
 out[6'd25-index]<=1'b0;
end
end
if(index>6'd12) begin
if(reverse==1'b0) begin
 out[index-6'd13]<=1'b0;
```

```
 out[6'd25-(index-6'd13)]<=1'b0;
end
else begin
 out[index-6'd13]<=1'b1;
 out[6'd25-(index-6'd13)]<=1'b1;
end
end
if(index<=6'd26 && reverse==1'b0) begin
index<=index+6'd1;
if(index==6'd26) begin
 reverse<=1'b1;
 index<=index-6'd1;
end
end
if(index>=6'd0 && reverse==1'b1) begin
index<=index-6'd1;
if(index<=6'd0) begin
 reverse<=1'b0;
 index<=6'd0;
 // Ket thuc hieu ung gan auto_next=4
end
end
```

– **Mô phỏng trên ModelSim:**

```
# 00000000000000000000000000                #01111111111111111111111110
# 10000000000000000000000001                #00111111111111111111111100
# 11000000000000000000000011                #00011111111111111111111000
# 11100000000000000000000111                #00001111111111111111110000
# 11110000000000000000001111                #00000111111111111111100000
# 11111000000000000000011111                #00000011111111111111000000
# 11111100000000000000111111                #00000001111111111110000000
# 11111110000000000001111111                #00000000111111111100000000
# 11111111000000000011111111                #00000000011111111000000000
# 11111111100000000111111111                #00000000001111110000000000
# 11111111110000001111111111                #00000000000111100000000000
# 11111111111000011111111111                #00000000000011000000000000
# 11111111111100111111111111                #00000000000000000000000000
# 11111111111111111111111111                #00000000000011000000000000
```

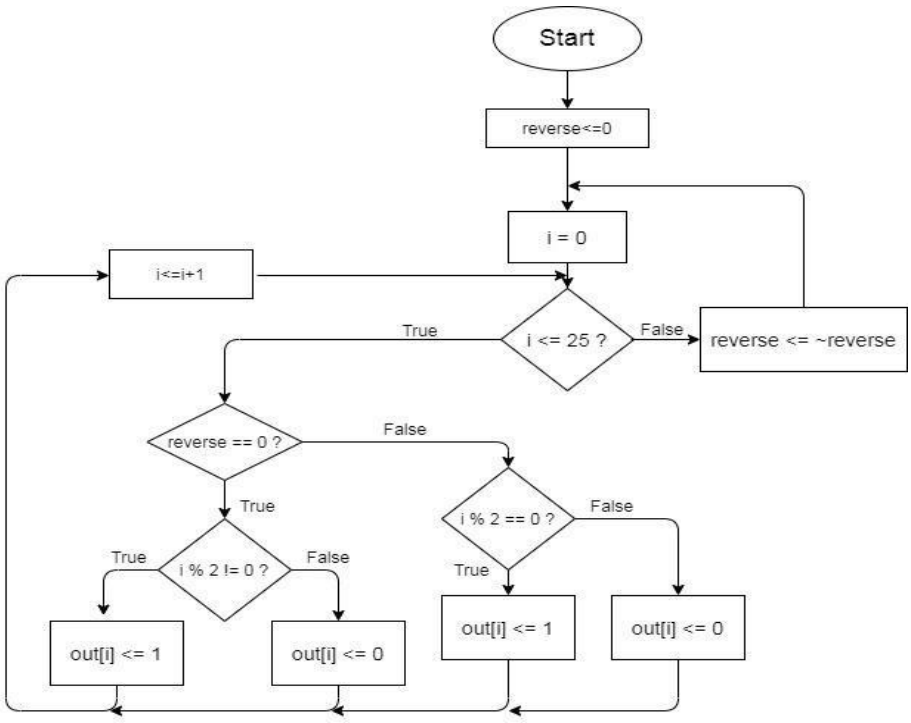
```
# 00000000000111100000000000      #11111111111000011111111111
# 00000000001111110000000000      #11111111110000001111111111
# 00000000011111111000000000      #11111111100000000111111111
# 00000001111111111100000000      #11111111000000000011111111
# 00000011111111111110000000      #11111100000000000001111111
# 00000111111111111111000000      #11111000000000000000111111
# 00001111111111111111100000      #11110000000000000000011111
# 00011111111111111111110000      #11100000000000000000001111
# 00111111111111111111111000      #11000000000000000000000111
# 01111111111111111111111100      #10000000000000000000000011
# 11111111111111111111111110      #00000000000000000000000001
# 11111111111100111111111111      #00000000000000000000000000
```

3.5 **Hiệu ứng 5:** Sáng chẵn lẻ. Khi các LED có chỉ số chẵn sáng thì các LED có chỉ số lẻ tắt và ngược lại.

❖ **Ý tưởng:**

− Dùng thuật toán để Led hiển thị theo yêu cầu. −

Flow chart: −



❖ **Hiện thực:**

− **Verilog code:**

```
for(i=6'd0;i<=6'd25;i=i+6'd1) begin
if(reverse==1'b0) begin
 if(i%2!=6'd0) out[i]<=1'b1;
else out[i]<=1'b0;
end
if(reverse==1'b1) begin
if(i%2==6'd0) out[i]<=1'b1;
else out[i]<=1'b0;
end
end
reverse<=~reverse;
// phan nay dung cho auto_next
if(index<=6'd25) index<=index+6'd1;
else begin
index<=6'd27;
 // Ket thuc hieu ung gan auto_next=5
auto_next<=4'd5;
end
```

– **Mô phỏng trên ModelSim:**

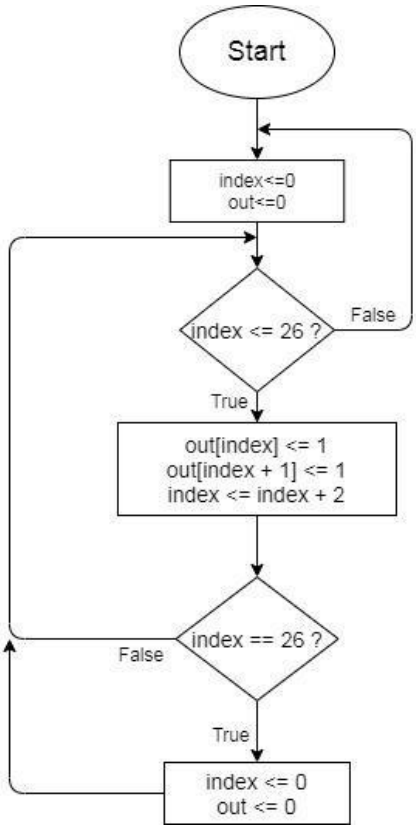
```
# 00000000000000000000000000
# 10101010101010101010101010
# 01010101010101010101010101
# 10101010101010101010101010
# 01010101010101010101010101
# 10101010101010101010101010
# 01010101010101010101010101
# 10101010101010101010101010
```

3.6 **Hiệu ứng 6** (Hiệu ứng xi nhan trái) Chuỗi 2 LED sẽ sáng dần từ trái qua phải, khi các LED sáng hết thì trong xung Clock tiếp theo các LED sẽ tăt hết và quay lại từ chuỗi 2 LED sáng dần từ trái qua phải.

❖ **Ý tưởng:**

− Dùng thuật toán để Led hiển thị theo yêu cầu. −

Flow chart:



❖ **Hiện thực:**

− **Verilog code:**

```
if(index<=6'd26) begin
out[index]<=1'b1;
out[index+6'd1]<=1'b1;
index<=index+6'd2;
if(index==6'd26) begin
index<=6'd0;
out<=26'd0;
 // Ket thuc hieu ung gan auto_next=6
end
end
```

− **Mô phỏng trên ModelSim:**

```
# 00000000000000000000000000
# 00000000000000000000000011
```

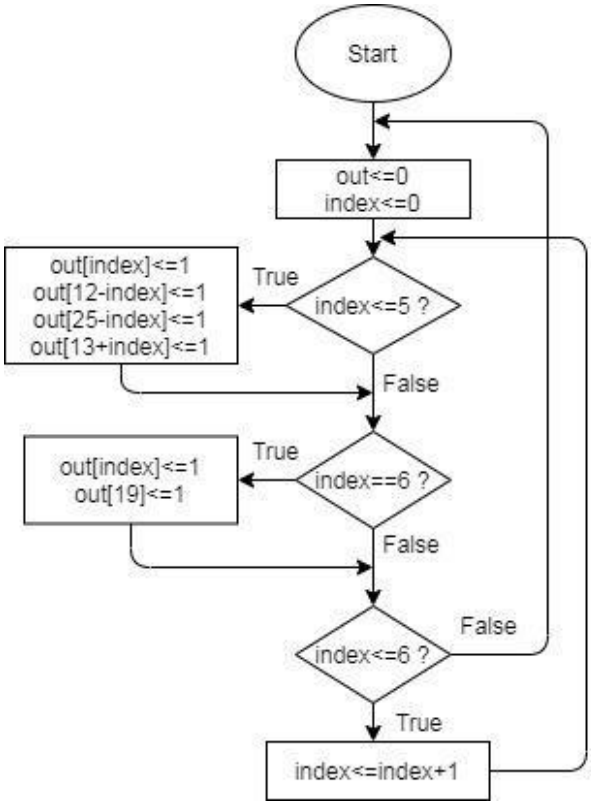
00000000000000000000001111
00000000000000000000111111
00000000000000000011111111
00000000000000001111111111
00000000000000111111111111
00000000000011111111111111
00000000001111111111111111
00000001111111111111111111
00000111111111111111111111
00011111111111111111111111
00111111111111111111111111
11111111111111111111111111
00000000000000000000000000

3.7 **Hiệu ứng 7:** Các LED sẽ sáng từ hai biên vào trong và từ trong ra ngoài, khi tấc cả các LED sáng hết thì các LED sẽ tắt hết và bắt đầu lại từ đầu.

❖ **Ý tưởng:**

− Dùng thuật toán để Led hiển thị theo yêu cầu. −

Flow chart:



❖ **Hiện thực:**

− **Verilog code:**

```
if(index<=6'd5) begin
out[index]<=1'b1;
out[6'd12-index]<=1'b1;
out[6'd25-index]<=1'b1;
out[6'd13+index]<=1'b1;
end
if(index==6'd6) begin
out[index]<=1'b1;
out[6'd25-6'd6]<=1'b1;
end
if(index<=6'd6) begin
index<=index+6'd1;
end
else begin
index<=6'd0;
out<=26'd0;
 // Ket thuc hieu ung gan auto_next=7
end
```

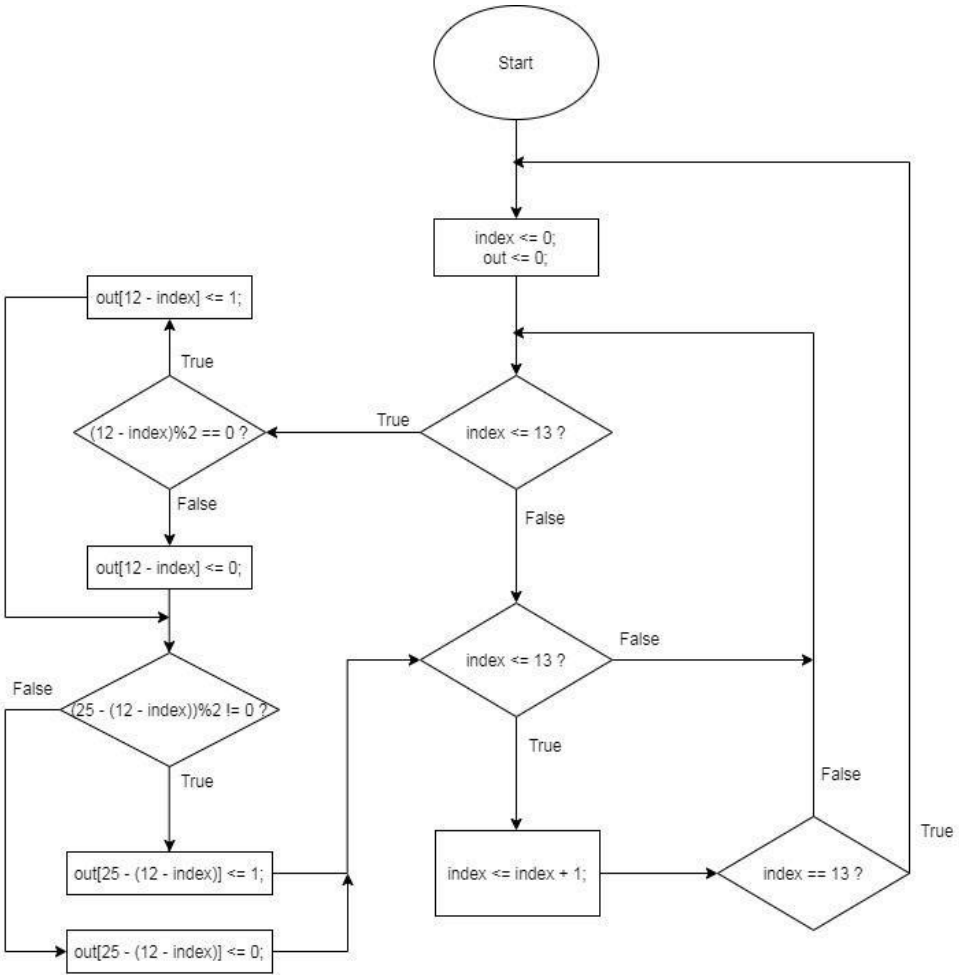
- **Mô phỏng trên ModelSim:**

```
# 00000000000000000000000000
# 10000000000011000000000001
# 11000000000111100000000011
# 11100000001111110000000111
# 11110000011111111000001111
# 11111000111111111100011111
# 11111101111111111110111111
# 11111111111111111111111111
# 00000000000000000000000000
```

3.8 **Hiệu ứng 8:** Các LED sẽ sáng từ trong ra ngoài và xen kẽ với nhau (Ngoại trừ 2 LED ở giữ), khi chạm biên các LED sẽ tắt hết và bắt đầu lại từ đầu.

✞ **Ý tưởng:**

- Dùng thuật toán để Led hiển thị theo yêu cầu.
- Flow chart:



✞ **Hiện thực:**

– **Verilog code:**

```
if(index<=6'd13) begin
if((6'd12-index)%2==6'd0) out[6'd12-index]<=1'b1;
else out[6'd12-index]<=1'b0;
if((6'd25-(6'd12-index))%2!=6'd0) out[6'd25-(6'd12-index)]<=1'b1;
else out[6'd25-(6'd12-index)]<=1'b0;
end
if(index<=6'd13) begin
index<=index+6'd1;
 if(index==6'd13) begin
index<=6'd0;
out<=26'd0;
```

```
 // Ket thuc hieu ung gan auto_next=8
end
end
```

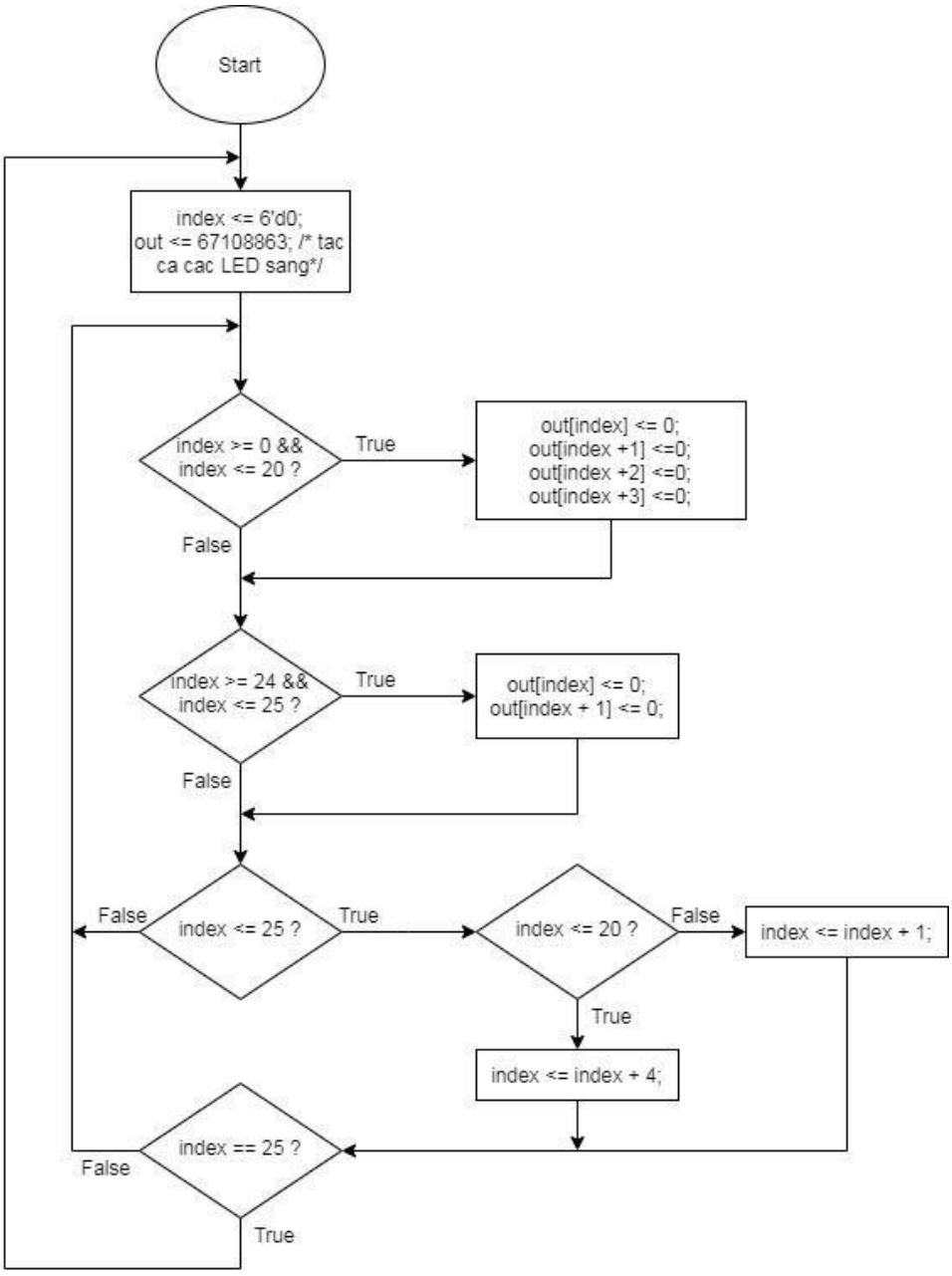
− **Mô phỏng trên ModelSim:**

```
# 00000000000000000000000000
# 00000000000011000000000000
# 00000000001011010000000000
# 00000000101011010100000000
# 00000010101011010101000000
# 00001010101011010101010000
# 00101010101011010101010100
# 10101010101011010101010101
# 00000000000000000000000000
```

3.9 **Hiệu ứng 9:** Ban đầu các LED sẽ sáng hết, sau đó lần lượt chuỗi 4 LED sẽ tắt dần từ phải qua trái đến khi tắt hết thì quá trình bắt đầu lại từ đầu

❖ **Ý tưởng:**

− Dùng thuật toán để Led hiển thị theo yêu cầu.
− Flow chart:



❖ **Hiện thực:**

− **Verilog code:**

```
if(index>=6'd0 && index<=6'd20) begin
out[index]<=1'b0;
out[index+6'd1]<=1'b0;
out[index+6'd2]<=1'b0;
out[index+6'd3]<=1'b0;
```

```
end
f(index>=6'd24 && index<=6'd25) begin
out[index]<=1'b0;
out[index+6'd1]<=1'b0;
end
if(index<=25) begin
if(index<=6'd20) index<=index+6'd4;
else index<=index+6'd1;
if(index==6'd25) begin
index<=6'd0;
for(i=6'd0; i<=6'd25; i=i+6'd1) out[i]<=1'b1;
 // Ket thuc hieu ung gan auto_next=9
end
end
```

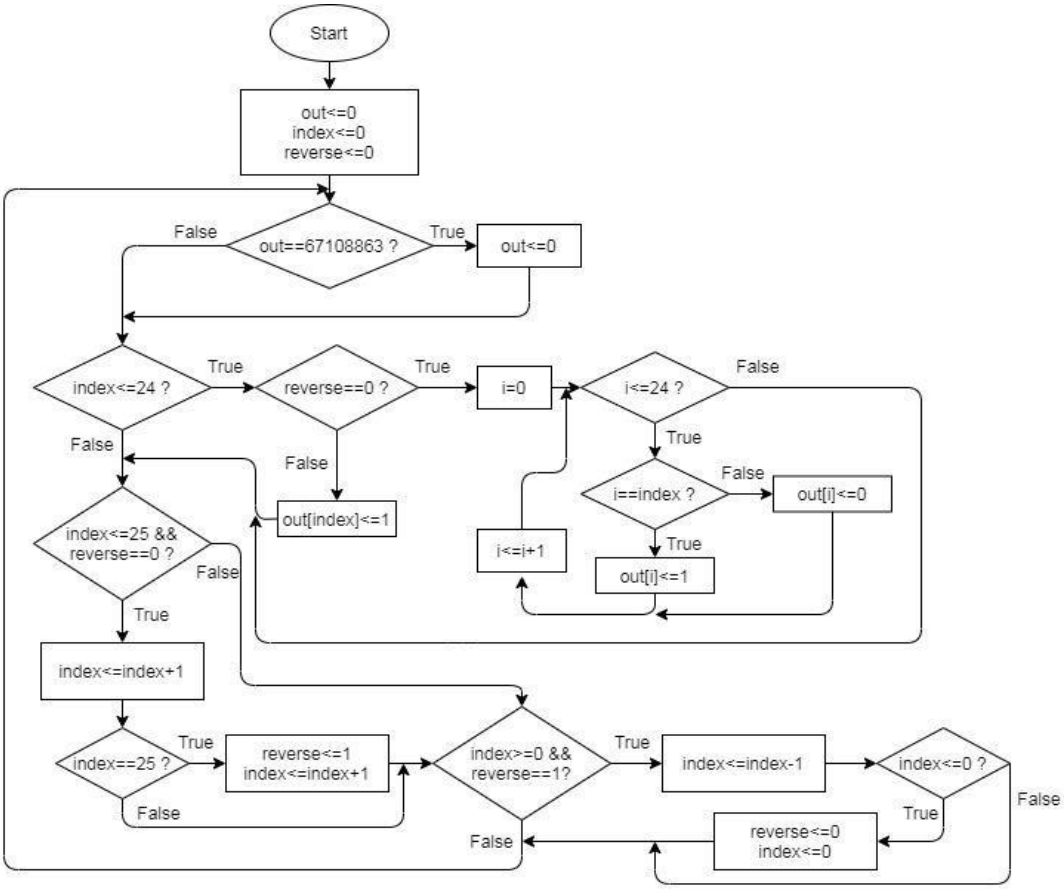
− **Mô phỏng trên ModelSim:**

```
# 11111111111111111111111111
# 11111111111111111111110000
# 11111111111111111100000000
# 11111111111111000000000000
# 11111111110000000000000000
# 11111100000000000000000000
# 11000000000000000000000000
# 00000000000000000000000000
# 11111111111111111111111111
```

3.10 **Hiệu ứng 10:** (Hiệu ứng viên đạn) LED biên trái luôn sáng, chuỗi một LED chuyển di chuyển dần đến LED biên trái, khi chạm LED biến trái thì các LED sẽ sáng dần từ vị trí chạm đến khi tấc cả các LED sáng hết, thì tắt tấc cả các LED và quá trình bắt đầu lại từ đầu.

❖ **Ý tưởng:**

− Dùng thuật toán để Led hiển thị theo yêu cầu.
− Flow chart:



❖ **Hiện thực:**

– **Verilog code:**

```
 if(out==26'b11111111111111111111111111) out<=26'd0;
else begin
 out[25]<=1'b1;
 if(index<=6'd24) begin
if(reverse==1'b0) begin
for(i=0; i<=6'd24; i=i+6'd1) begin
if(i==index) out[i]<=1'b1;
else out[i]<=1'b0;
end
end
else out[index]<=1'b1;
 end
 if(index<=6'd25 && reverse==1'b0) begin
index<=index+6'd1
f(index==6'd25) begin
```

```
reverse<=1'b1;
index<=index-6'd1;
end
 end
 if(index>=6'd0 && reverse==1'b1) begin
index<=index-6'd1;
if(index<=6'd0) begin
reverse<=1'b0;
index<=6'd0; // Ket thuc hieu ung gan auto_next=0
end
 end
 end
```

- **Mô phỏng trên ModelSim:**

```
# 00000000000000000000000000
# 10000000000000000000000001
# 10000000000000000000000010
# 10000000000000000000000100
# 10000000000000000000001000
# 10000000000000000000010000
# 10000000000000000000100000
# 10000000000000000001000000
# 10000000000000000010000000
# 10000000000000000100000000
# 10000000000000001000000000
# 10000000000000010000000000
# 10000000000000100000000000
# 10000000000001000000000000
# 10000000000010000000000000
# 10000000000100000000000000
# 10000000001000000000000000
# 10000000010000000000000000
# 10000001000000000000000000
# 10000010000000000000000000
# 10001000000000000000000000
# 10010000000000000000000000
# 10100000000000000000000000
# 11000000000000000000000000
# 11100000000000000000000000
# 11110000000000000000000000
# 11111000000000000000000000
# 11111100000000000000000000
# 11111110000000000000000000
# 11111111000000000000000000
# 11111111100000000000000000
# 11111111110000000000000000
# 11111111111000000000000000
# 11111111111100000000000000
# 11111111111110000000000000
# 11111111111111000000000000
# 11111111111111100000000000
# 11111111111111110000000000
# 11111111111111111000000000
# 11111111111111111100000000
# 11111111111111111110000000
# 11111111111111111111000000
# 11111111111111111111100000
# 11111111111111111111110000
# 11111111111111111111111000
# 11111111111111111111111100
# 11111111111111111111111110
# 11111111111111111111111111
# 00000000000000000000000000
```

4. Hiển thị thông tin mode ra LED 7 đoạn:

❖ Ý tưởng:

- Dùng 2 LED 7 đoạn 6 và 7 để biểu diễn.
- Dùng một biến tên là controlled76 10 bits để điều khiển 2 LED. 5 bits có chỉ số thấp nhất điều khiển LED 6 và 5 bits còn lại điều khiển LED 7, trong 5 bits này gồm 1 bit để bật tắt LED, và 4 bits dùng làm input cho LED (4 bits có chỉ số thấp nhất).
- Để truyền vào input của LED 6 ta lấy phần dư của mode_in khi chia cho 10 (là biến lưu trữ giá trị của hiệu ứng), tương tự ta lấy phần nguyên của mode_in khi chia cho 10 làm input cho LED 7.
- Khi input của LED 7 là 0 thì ta tắt không cho hiện thị (vì 0 lúc này vô nghĩa), còn LED 6 thì luôn bậc.

❖ Hiện thực:

- Verilog code:

```
output [6:0] led7, led6;

 wire [9:0] controlled76;

 assign controlled76[8:5]=mode_in/4'd10;

assign controlled76[3:0]=mode_in%4'd10;

assign controlled76[9]=(controlled76[8:5]==4'd0)? 1'b0:1'b1;

assign controlled76[4]=1'b1;

led7_decoder l7(controlled76[9], controlled76[8:5], led7);

led7_decoder l6(controlled76[4], controlled76[3:0], led6);
```

5. Hiển thị thông tin tần số ra LED 7 đoạn:

❖ Ý tưởng:

- Dùng 2 LED 7 đoạn 4, 5 để hiển thị thông tin tần số.
- Dùng một biến tên là controlled54 10 bits để điều khiển 2 LED. 5 bits có chỉ số thấp nhất điều khiển LED 4 và 5 bits còn lại điều khiển LED 5, trong 5 bits này gồm 1 bit để bật tắt LED, và 4 bits dùng làm input cho LED (4 bits có chỉ số thấp nhất).
- Để truyền giá trị cho controlled54 ta dùng multiplexer với biến select là mode_hz (biến lưu trữ giá trị của tần số). Cụ thể như sau:

mode_hz	controlled54	Ý nghĩa	Tần số
000	10'b1000010101	Bật LED 5 và cho hiển thị số 0, bật LED 4 và cho hiển thị số 5	0.5 Hz
001	10'b0000010001	Tắt LED 5, bật LED 4 và cho hiển thị số 1	1 Hz
010	10'b0000010010	Tắt LED 5, bật LED 4 và cho hiển thị số 2	2 Hz
011	10'b0000010100	Tắt LED 5, bật LED 4 và cho hiển thị số 4	4 Hz
100	10'b0000011000	Tắt LED 5, bật LED 4 và cho hiển thị số 8	8 Hz
Default	10'b0000011000	Tắt LED 5, bật LED 4 và cho hiển thị số 8	8 Hz

❖ **Hiện thực:**

− Verilog code:

```
output [6:0] led5, led4;
wire [9:0] controlled54;
assign controlled54=(mode_hz==3'd0)? 10'b1000010101:
                    (mode_hz==3'd1)? 10'b0000010001:
                    (mode_hz==3'd2)? 10'b0000010010:
                    (mode_hz==3'd3)? 10'b0000010100:
                    (mode_hz==3'd4)? 10'b0000011000:10'b0000011000;
led7_decoder l5(controlled54[9], controlled54[8:5], led5);
led7_decoder l4(controlled54[4], controlled54[3:0], led4);
```

6. Hệ thống phải hỗ trợ hai chế độ điều khiển: điều khiển tự động (cho phép các hiệu ứng LED tự động thay đổi) và điều khiển bằng tay (thay đổi hiệu ứng LED bằng tay)

❖ **Ý tưởng:**

− Dùng input auto 1 bit để lựa chọn chế độ điều khiển, khi auto=1 thì điều khiển tự động khi auto=0 thì điều khiển bằng tay.

− Dùng một LED đơn để thông báo, khi auto=1 thì LED này sáng, khi auto=0 thì LED này tắt. LED đơn này ứng với output out_auto.

− Dùng một register mode_in 4 bits để lưu trữ giá trị của hiệu ứng. Khi auto=0 thì mode_in sẽ lưu trữ giá trị của mode (là input 4 bits người dùng nhập vào để lựa chọn hiệu ứng), khi auto=1 thì mode_in sẽ lưu trữ giá trị của auto_next.

− auto_next: là một biến tự động thay đổi giá trị khi kết thúc một hiệu ứng. Dấu hiệu kết thúc một hiệu ứng có ghi trong từng hiệu ứng ở trên.

❖ **Hiện thực:**

− Verilog code:

```
output out_auto;
input auto;
reg [3:0] auto_next=4'd0;
```

```
    reg [3:0]
    auto_next=4'd0;
always @(posedge
    hz_out) begin
 if (auto==0)
    mode_in<=mode;
 else
    mode_in<=auto_next;
end
```

*Chú thích: Trong module chính có thể đặt ở vị trí khác nhưng bản chất nó vẫn như vậy.

7. Hiệu ứng CE20 nhập nháy : Ở cụm 4 LED 7 đoạn 0-3 ban đầu các LED tắt hết, sau đó lần lượt chữ C, E, 2, 0 hiện liên từ trái qua phải, khi hiện lên hết thì dừng một lúc rồi tắt hết, quá trình lặp lại từ đầu.

❖ Ý tưởng:

− Dùng biến onled3210 4 bits để bật tắt các LED. onled3210[0] ứng với điều khiển LED 0, tương tự cho các LED khác.

− Sử dụng module chia tần số để điều khiển tốc độ nhấp nháy. Trong hệ thống này nhóm chọn tần số là 2 Hz. Dùng biến hz_led để làm xung clock đầu ra của module chia tần số này.

− Dùng một bộ đếm countled 3 bits để điều khiển bật tắt các LED. Khi countled=0 thì LED 3 hiện chữ C, countled=1 thì LED 2 hiện chữ E, countled=2 thì LED 3 hiện số 2, countled=3 thì LED 3 hiện số 0, sau đó bộ đếm đếm tới 6, khi đạt giá trị này thì countled quay lại giá trị 0, lúc này tắt tấc cả các LED. Quá trình lặp lại từ đầu.

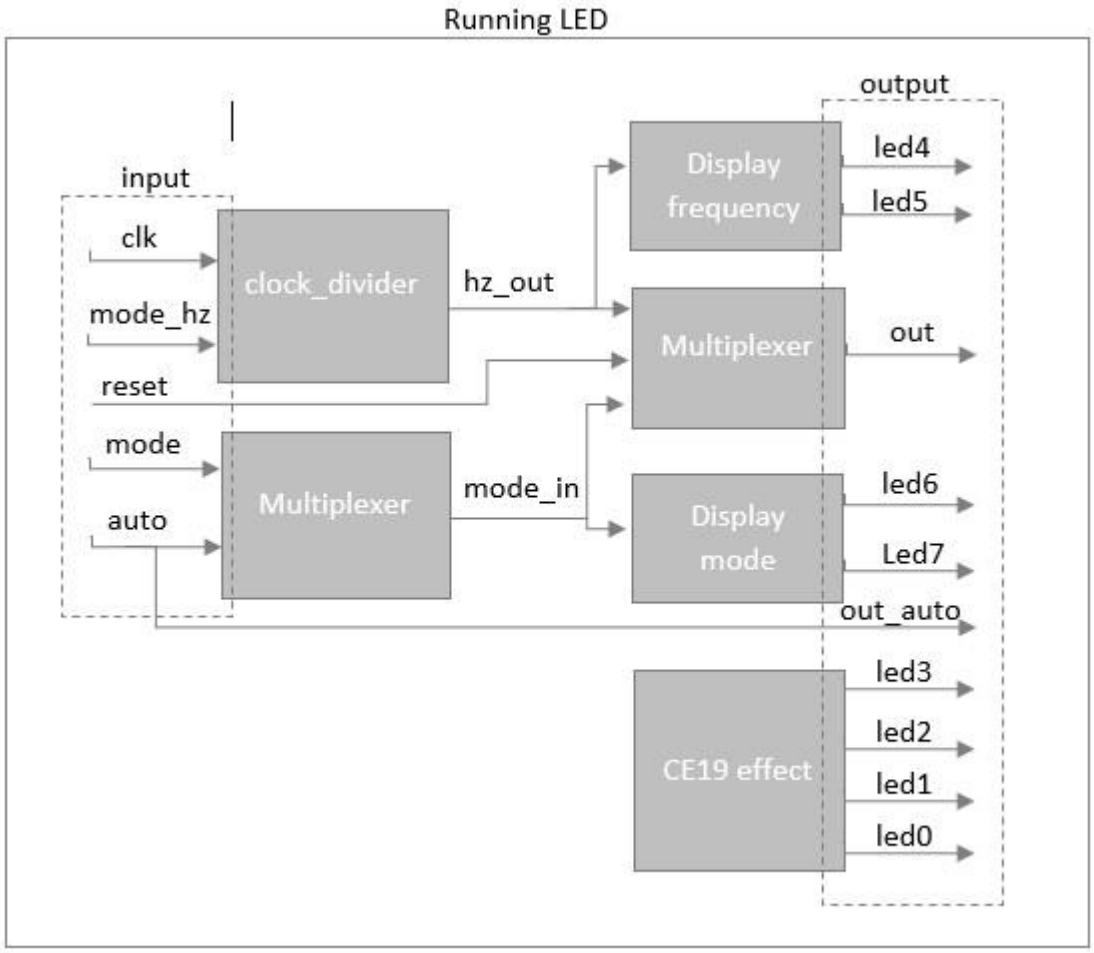
❖ Hiện thực:

− Verilog code:

```
 output [6:0] led3, led2, led1, led0;
 wire hz_led;
 reg [2:0] countled=3'd0;
 reg [3:0] onled3210;
clock_divider c_d2(clk, 3'd2, hz_led);
always @(posedge hz_led) begin
 if(countled<=3'd5) begin
 countled<=countled+3'd1;
 if(countled>=3'd0&&countled<=3'd3)
onled3210[countled]<=1'b1;
 end
 if(countled==3'd6) begin
 countled<=3'd0;
 onled3210<=4'd0;
end
```

```
end
led7_decoder l3(onled3210[0], 4'b1110, led3);
led7_decoder l2(onled3210[1], 4'b1111, led2);
led7_decoder l1(onled3210[2], 4'b0010, led1);
led7_decoder l0(onled3210[3], 4'b0000, led0);
```

8. Sơ đồ khối của hệ thống:



III. Đánh giá kết quả:

- Ứng dụng của đề tài vào thực tế.
  - Ứng dụng trong lĩnh vực trang trí, ta có thể dùng đề tài này để thiết kế các bảng hiệu, bảng quảng cáo, trang trí nội thất ,….
  - Ứng dụng trong giao thông: hiệu ứng báo rẽ, …
- Đề xuất hướng phát triển.
  - Ngoài thiết kế hiệu ứng chỉ trên các LED đơn, ta có thể làm thêm LED 7 đoạn

hoặc thậm chí là cả LCD để đề tài thêm đa dạng cũng như bắt mắt.

- Tạo thêm một số hiệu ứng đẹp hơn nữa.

➢ Ưu và nhược điểm của hệ thống:

- Ưu điểm:
  - Thiết kế ngắn gọn.
  - Hiệu ứng đẹp mắt.
  - Đảm bảo các yêu của cầu đề tài.
- Nhược điểm:
  - Do dùng thuật toán nên thời gian hoàn thành một hiệu ứng khá lâu.
  - Hệ thống khó hiểu, khó debug và fixbug.
  - Do dùng chung các biến nên khi chuyển giữa các chế độ thì sẽ gặp lỗi, dẫn đến chế độ auto chưa hoàn thiện.

IV. Kế hoạch làm việc

Công việc được chia phù hợp với khả năng, ưu điểm của từng cá nhân trong nhóm:

Lê Thanh Dương:

- Chịu tránh nhiệm chính phần thiết kế module, coding, debug, lên ý tưởng các hiệu ứng, viết báo cáo.

Bùi Đăng Khoa:

- Chịu trách nhiệm chính phần viết báo cáo, thuyết trình, lên ý tưởng các hiệu ứng.

Nguyễn Thành Chương:

- Hỗ trợ các thành viên khác trong nhóm, thiết kế module, coding, lên ý tưởng các hiệu ứng.

V. Tài liệu tham khảo

- Các slide bài giảng thiết kế luận lí trên BKElearning.
- Các video bài giảng trên BKElearning.
- https://www.fpga4student.com/2017/08/verilog-code-for-clock-divider-on-fpga.html.

The End