



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

**BÀI BÁO CÁO ASSIGNMENT LAB02
THIẾT KẾ HỆ THỐNG SỐ VỚI HDL**

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 SỐ 3

I. Mục tiêu

- Trong bài thực hành này, sinh viên sẽ dùng procedural assignment để thiết kế các mạch đếm (Counter) và mạch định thời (Timer).
- Thực hành sử dụng **LPM** (Library of Parameterized Modules) của Altera

http://quartushelp.altera.com/14.1/master.htm#mergedProjects/reference/glossary/def_lpm.htm

http://quartushelp.altera.com/14.1/master.htm#mergedProjects/hdl/mega/mega_list_mega_lpm.htm

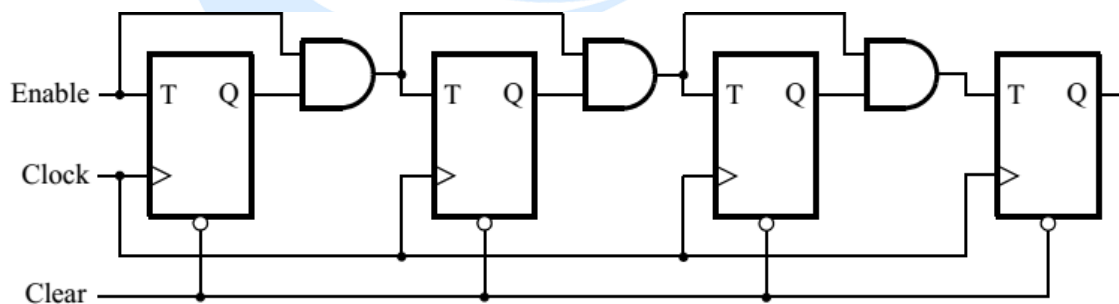
II. Chuẩn bị thực hành

- 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.
- Bài chuẩn bị được tính vào điểm bài báo cáo của Lab.

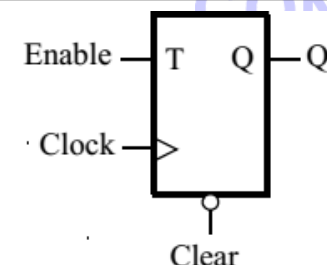
III. Nội dung thực hành

Câu 1.

1.1. Thiết kế bộ đếm 4-bit như hình dưới:



Bảng sự thật của T-FF



Clear	Enable	Clock	Q+
0	x	x	0
1	0	↓	Q
1	1	↓	Q'

Hướng dẫn:

- Tín hiệu Clear trong T-FF trên là Clear bất đồng bộ và tích cực mức thấp

Verilog code:

```
always @(posedge Clock or negedge Clear)
    if (Clear == 1'b0)
```

....

- Clock = ~KEY[0]; Clear = SW[1];
- Enable = SW[0]; LEDR[0] = Enable;
- LEDG [3:0] = Q[3:0]

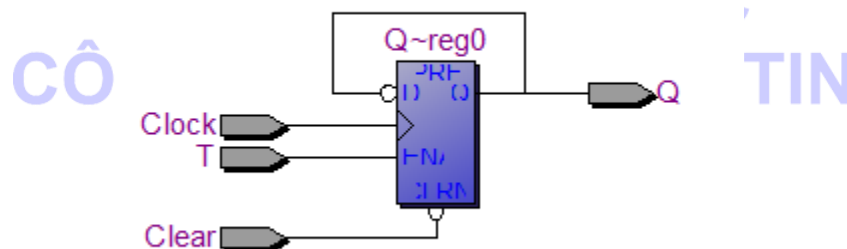
Yêu cầu: SV thực hiện mô phỏng VectorWaveform và nạp KIT demo cho mạch đếm.

Đầu tiên, thiết kế 1 bộ T FF:

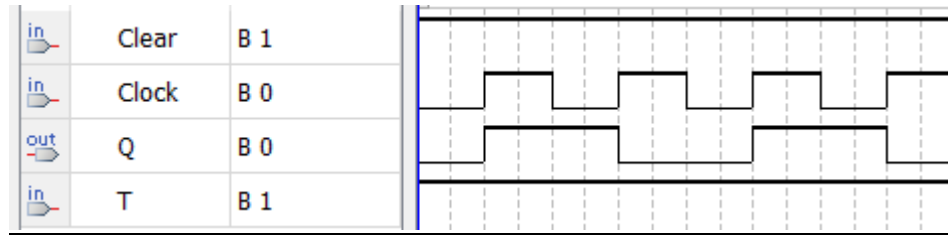
Code Thực Thi:

```
1  module T_FF(T, Q, Clock, Clear);
2      input T, Clock, Clear;
3      output reg Q;
4      always @(posedge Clock or negedge Clear) begin
5          if (Clear == 1'b0) begin
6              Q <= 1'b0;
7          end else begin
8              if (T) begin
9                  Q <= ~Q;
10             end
11         end
12     end
13 endmodule
```

RTL Viewer:



Mô phỏng Waveform:



Giải thích:

- Reset (Clear = 0)
 - Khi tín hiệu Clear ở mức thấp (0), đầu ra Q bị đặt về 0 (bất kể giá trị của Clock hay T).
 - Đây là tính năng reset chủ động mức thấp (active-low reset)
- Hoạt động bình thường (Clear = 1)
 - Khi Clear = 1, Flip-Flop hoạt động như một T Flip-Flop chuẩn:
 - Nếu T = 0, giá trị của Q giữ nguyên.
 - Nếu T = 1, giá trị của Q sẽ đảo trạng thái (toggle) vào cạnh lên của Clock.

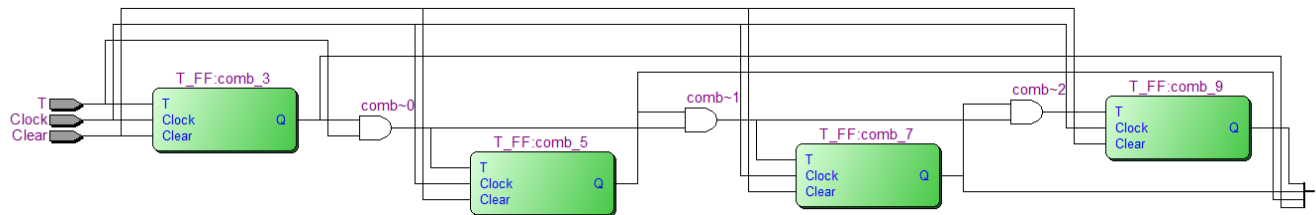
Tiếp theo đó, chúng ta tiến hành thiết kế bộ đếm 4 bit:

Code thực thi:

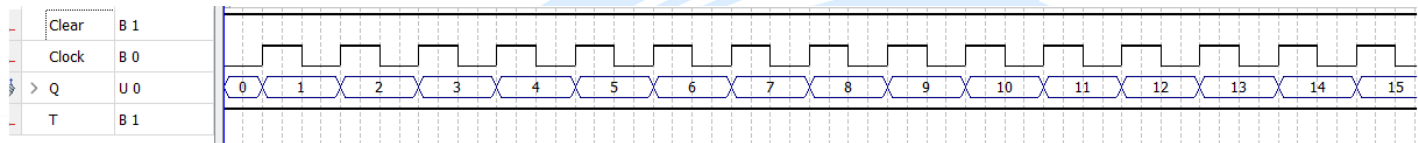
```

1  module Counter(Q, T, Clock, Clear);
2  output [3:0] Q;
3  input Clock, Clear, T;
4  wire T1, T2, T3;
5  T_FF(T, Q[0], Clock, Clear);
6  and (T1, Q[0], T);
7  T_FF(T1, Q[1], Clock, Clear);
8  and (T2, Q[1], T1);
9  T_FF(T2, Q[2], Clock, Clear);
10 and (T3, Q[2], T2);
11 T_FF(T3, Q[3], Clock, Clear);
12 endmodule
    
```

RTL Viewer:



Mô phỏng waveform:



Giải thích:

1. Mô tả hoạt động của mạch

- Inputs (đầu vào):
 - Clear (Reset): Khi Clear = 0, mạch được reset về giá trị ban đầu (Q = 0).
 - Clock (Xung nhịp): Tín hiệu xung nhịp điều khiển quá trình đếm.
 - T (Toggle): Khi T = 1, mạch sẽ thực hiện quá trình đếm theo xung clock.
- Output (đầu ra):
 - Q (4-bit Counter Output): Biểu diễn giá trị đếm (0 → 15 theo nhị phân).

2. Chức năng của mạch

- Đây là mạch đếm nhị phân 4-bit tăng dần (Binary Up Counter).
- Mạch có khả năng đếm từ 0 đến 15 ($2^4 - 1$) rồi quay về 0.
- Mỗi lần có một xung clock, giá trị Q tăng lên 1.
- Khi đến giá trị tối đa (15 - 1111₂), mạch sẽ quay lại 0 và bắt đầu đếm lại.

1.2. Thiết kế bộ đếm 4-bit như trong câu 1.1, nhưng mô tả ở mức Behavior

$$Q \leq Q + 1$$

Thực hiện gán chân như câu 1.1

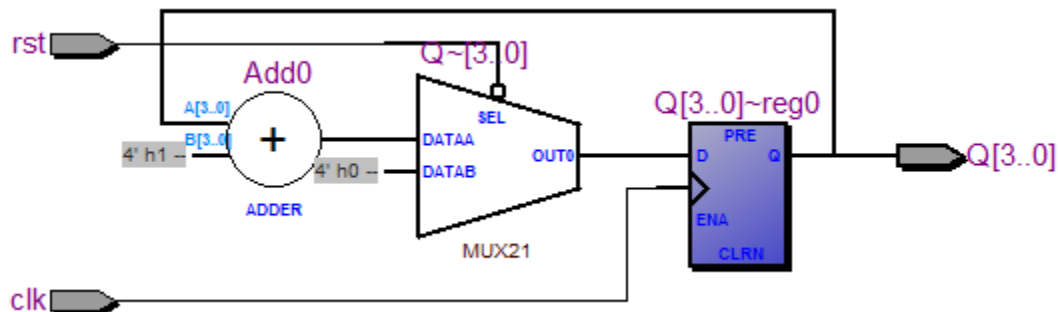
Sử dụng RTL Viewer để xem mạch sau khi Synthesis so với mạch trong câu 1.1

So sánh tần số Fmax của mạch trong câu 1.1 và câu 1.2

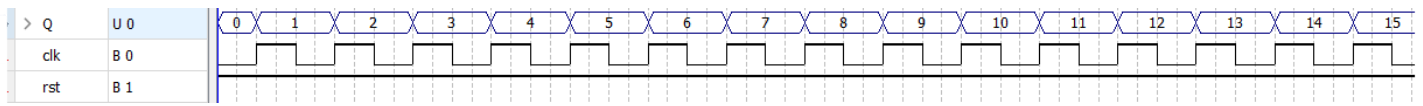
Code thực thi:

```
1 module Counter_Behavior (clk, rst, Q);
2   input wire clk;
3   input wire rst;
4   output reg [3:0] Q;
5   always @ (posedge clk) begin
6     if (!rst)
7       Q <= 4'b0000;
8     else
9       Q <= Q + 1;
10  end
11
12 endmodule
```

RTL Viewer:



Mô phỏng Waveform:



Giải thích

1. Mô tả hoạt động của mạch

- **clk (Clock):**
 - Là tín hiệu xung nhịp, điều khiển quá trình đếm.
 - Khi có xung cạnh lên (posedge), giá trị của bộ đếm tăng lên 1.
- **rst (Reset - Clear):**
 - Là tín hiệu đặt lại (Reset đồng bộ).
 - Khi **rst = 0**, bộ đếm sẽ được đặt về 0 ngay tại cạnh lên của xung clock.
 - Khi **rst = 1**, bộ đếm sẽ tiếp tục tăng theo clock.
- **Q (4-bit Output):**
 - Là giá trị của bộ đếm.
 - Bắt đầu từ **0000 (0 decimal)**.
 - Tăng dần mỗi chu kỳ clock ($Q \leq Q + 1$).
 - Khi đến giá trị **1111 (15 decimal)**, xung clock tiếp theo sẽ làm nó quay lại **0000**.

2. Cách hoạt động của bộ đếm

- Ban đầu, nếu **rst = 1**, Q sẽ được đặt lại về **0000**.
- Khi **clk có cạnh lên** và **rst = 1**, giá trị Q sẽ tăng dần từ **0 → 15 (0000 → 1111)**.
- Khi Q đạt **1111 (15 decimal)**, xung clock tiếp theo sẽ làm Q quay lại **0000 (0 decimal)** và tiếp tục đếm.



So sánh tần số Fmax:

Tần số Fmax của câu 1.1:

Slow Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	585.48 MHz	380.08 MHz	clk	limit due to minimum period restriction (max I/O toggle rate)

Tần số Fmax của câu 1.2:

Slow Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	585.48 MHz	380.08 MHz	clk	limit due to minimum period restriction (max I/O toggle rate)

- Tần số Fmax của mạch 1.1 và 1.2 là bằng nhau.

Câu 2.

2.1. Sửa lại thiết kế bộ đếm 4-bit trong câu 1.2. bằng cách dùng Parameter và Clear đồng bộ.

Verilog code cho mạch có Clear đồng bộ, tích cực mức 0:

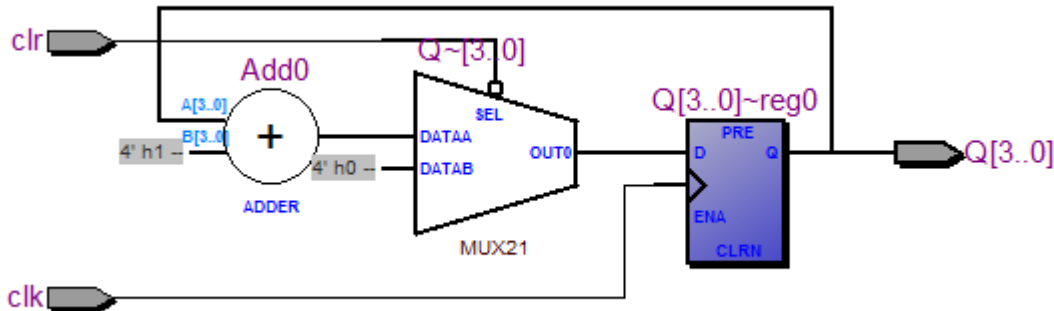
```
always @(posedge Clock)
    if (Clear == 1'b0)
```

Code Thực Thi:

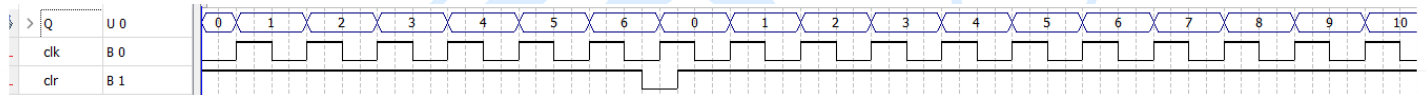
```
module CounterParameterNew #(parameter WIDTH = 4) (
    input wire clk,
    input wire clr,
    output reg [WIDTH-1:0] Q
);

    always @(posedge clk) begin
        if (clr == 1'b0)
            Q <= {WIDTH{1'b0}};
        else
            Q <= Q + 1;
        end
    endmodule
```


RTL Viewer:



Mô phỏng Waveform:



Giải thích:

1. Chức năng

- Bộ đếm nhận tín hiệu **xung nhịp (clk)** để điều khiển quá trình đếm.
- Khi tín hiệu **xóa (clr) = 0**, bộ đếm sẽ được reset về 0000.
- Khi **clr = 1**, bộ đếm sẽ tăng giá trị lên mỗi khi có cạnh lên (posedge) của tín hiệu xung nhịp.
- Độ rộng bộ đếm được xác định bởi tham số WIDTH (mặc định là 4-bit).

2. Cách Hoạt Động

Trạng thái đầu (Reset)

- Khi **clr = 0** (Clear mức thấp), giá trị Q sẽ được đặt lại về 0000.

Hoạt động đếm

- Khi **clr = 1**, bộ đếm bắt đầu tăng từ 0000, 0001, 0010, ... đến 1111 rồi quay lại 0000 (vì đây là bộ đếm 4-bit).
- Quá trình này lặp lại mỗi khi có cạnh lên (posedge) của tín hiệu clk.

Cách 2 (Special):

Đầu tiên, ta thiết kế lại bộ FlipFlop với chân set và clear (reset)

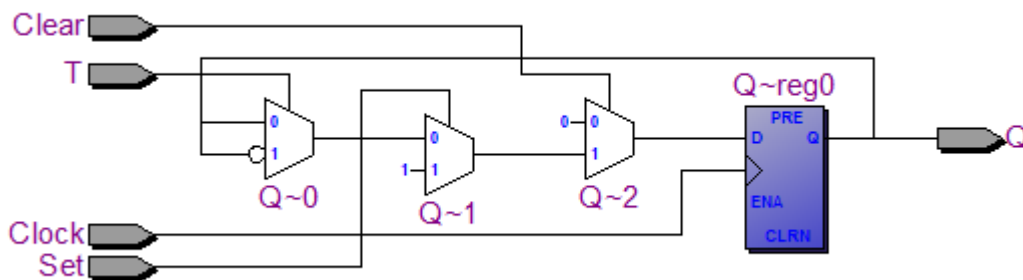
Code Thực Thi:

```

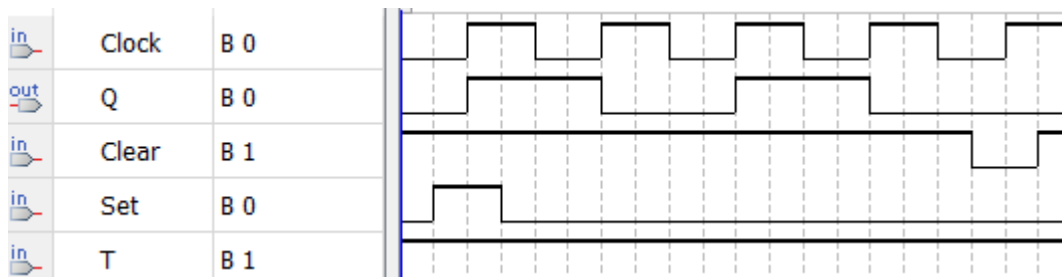
1  module T_FF_With_Sync_Set_And_Reset(T, Q, Clock, Set, Clear).
2      input T, Clock, Set, Clear;
3      output reg Q;
4      always @(posedge Clock) begin
5          if (!Clear) begin
6              Q <= 0;
7          end else if (Set) begin
8              Q <= 1;
9          end else if (T) begin
10             Q <= ~Q;
11         end
12     end
13 endmodule

```

RTL Viewer:



Mô phỏng Waveform:



Giải thích:

1. Mô tả chức năng của T Flip-Flop có chân Set và Clear (Reset)

- Clock (clk): Điều khiển sự thay đổi trạng thái.
- Q: Đầu ra (Output).
- Clear (Reset): Đặt lại Q về 0.
- Set: Đặt Q về 1.
- T (Toggle Input): Khi T = 1, Q sẽ thay đổi trạng thái ở cạnh lên của Clock.



2. Chức năng của từng tín hiệu

1. Clear (Reset):

- Khi Clear = 0, Q ngay lập tức về 0, bất kể Clock hay T như thế nào.
- Đây là một tín hiệu ưu tiên (Asynchronous Reset).

2. Set:

- Khi Set = 1, Q ngay lập tức được đặt về 1, bất kể Clock hay T như thế nào.
- Thường là tín hiệu ưu tiên thứ hai sau Reset.

3. T (Toggle Input):

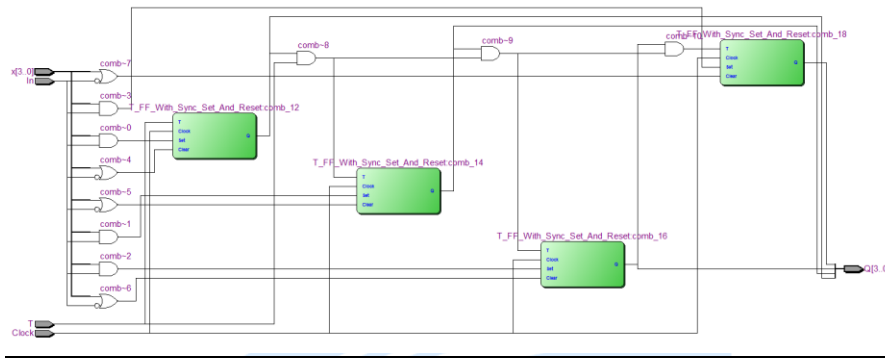
- Khi T = 0, giá trị của Q không thay đổi.
- Khi T = 1, Q sẽ đảo trạng thái (Toggle) vào mỗi cạnh lên của Clock.
- Nếu Q đang là 0, nó sẽ thành 1, và ngược lại.

Cuối cùng ta thiết kế lại bộ đếm 4 bit với Parameter

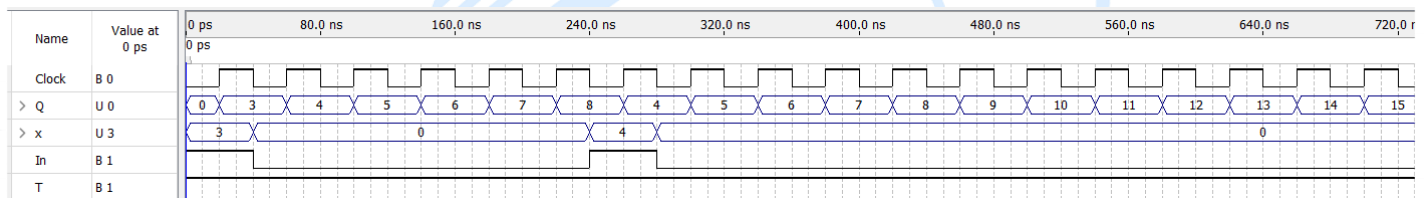
Code Thực Thi:

```
1 module CounterParameter(x, T, Clock, Q, In);
2   input [3:0] x;
3   output [3:0] Q;
4   input T, In, Clock;
5   wire A1, A2, A3, A4, O1, O2, O3, O4;
6   and (A1, x[0], In);
7   and (A2, x[1], In);
8   and (A3, x[2], In);
9   and (A4, x[3], In);
10  wire nIn;
11  not (nIn, In);
12  or (O1, x[0], nIn);
13  or (O2, x[1], nIn);
14  or (O3, x[2], nIn);
15  or (O4, x[3], nIn);
16  wire T1, T2, T3;
17  T_FF_With_Sync_Set_And_Reset(T, Q[0], Clock, A1, O1);
18  and (T1, Q[0], T);
19  T_FF_With_Sync_Set_And_Reset(T1, Q[1], Clock, A2, O2);
20  and (T2, Q[1], T1);
21  T_FF_With_Sync_Set_And_Reset(T2, Q[2], Clock, A3, O3);
22  and (T3, Q[2], T2);
23  T_FF_With_Sync_Set_And_Reset(T3, Q[3], Clock, A4, O4);
24  endmodule
```

RTL Viewer:



Mô phỏng Wareform:



Giải thích:

1. Mô tả chức năng:

- Clock: Xung nhịp điều khiển bộ đếm, giá trị của bộ đếm thay đổi theo cạnh lên của xung clock.
- Q: Giá trị hiện tại của bộ đếm.
- X: Giá trị đầu vào có thể được nạp vào bộ đếm.
- In: Tín hiệu điều khiển cho phép nạp giá trị từ X vào Q.
- T: Tín hiệu toggle, nếu $T = 1$, bộ đếm sẽ tăng dần.

2. Cách hoạt động

Chế độ đếm thông thường (Increment Mode)

- Khi $In = 0$, bộ đếm hoạt động như bộ đếm nhị phân bình thường.

- Ở mỗi cạnh lên của Clock, giá trị Q tăng dần: 0-1-2-3-...-15-0.

Chế độ nạp giá trị (Load Mode)

- Khi In = 1, bộ đếm sẽ nạp giá trị từ X vào Q tại cạnh lên của xung Clock.

Quan sát hình, ta thấy:

- Ban đầu bộ đếm hoạt động bình thường (Q tăng từ 0,1,2...).
- Khi In = 1 vào khoảng 80 ns, Q lập tức nhận giá trị từ X = 3.
- Sau đó, bộ đếm tiếp tục tăng dần từ 3 (4,5,6...).
- Khi In = 1 vào khoảng 240 ns, Q nhận giá trị mới từ X = 4 và tiếp tục tăng từ đó.

2.2. Thiết kế bộ đếm 8-bit bằng cách gọi bộ đếm 4-bit trong câu 2.1. và hiệu chỉnh giá trị Parameter.

Giá trị ngõ ra bộ đếm Q[7:0] được gán cho LEDG[7:0]

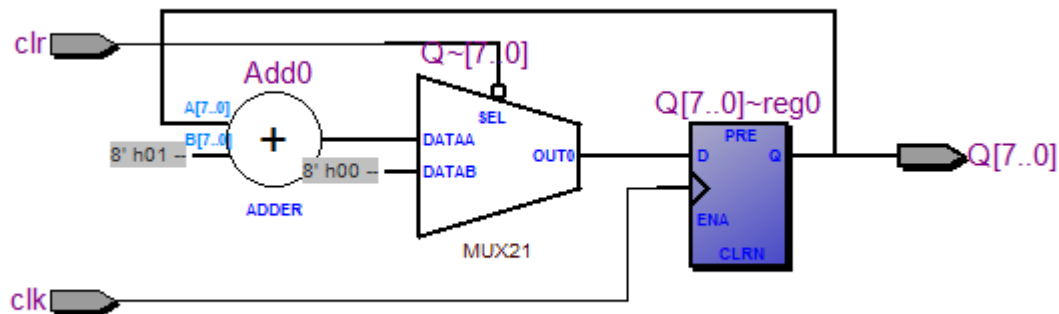
Gán chân cho Clock, Enable, Clear giống câu 1.1

Cách 1:

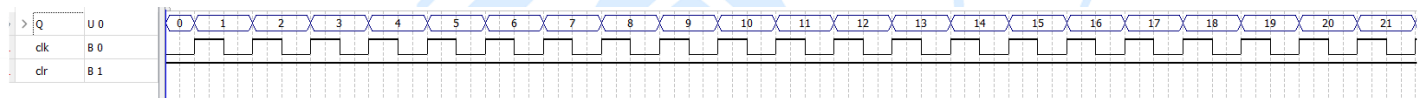
Code Thực Thi:

```
1 module CounterParameter8New #(parameter WIDTH = 8) (  
2     input wire clk,  
3     input wire clr,  
4     output reg [WIDTH-1:0] Q  
5 );  
6  
7 always @(posedge clk) begin  
8     if (clr == 1'b0)  
9         Q <= {WIDTH{1'b0}};  
10    else  
11        Q <= Q + 1;  
12 end  
13
```

RTL Viewer:



Mô phỏng Waveform:



Giải thích:

1. Chức năng

- Đếm tăng (Q tăng lên mỗi cạnh lên của clk).
- Sử dụng tham số WIDTH để dễ dàng thay đổi độ rộng bit của bộ đếm (mặc định là 8-bit).
- Xóa đồng bộ (clr) mức thấp:
 - Khi clr = 0, bộ đếm sẽ được đặt về 00000000.
 - Khi clr = 1, bộ đếm sẽ tiếp tục đếm từ giá trị hiện tại.

2. Cách Hoạt Động

Trạng thái ban đầu (Reset)

- Khi clr = 0, bộ đếm sẽ reset về 00000000 (tất cả bit bằng 0).

- Điều này xảy ra khi có cạnh lên (posedge clk).

Quá trình đếm

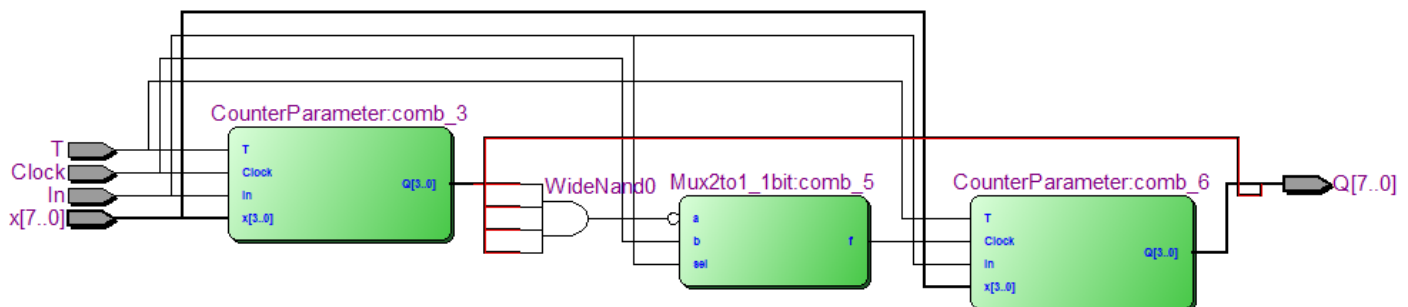
- Khi clr = 1, bộ đếm bắt đầu đếm từ 00000000, 00000001, 00000010, ... cho đến 11111111 (giá trị tối đa với 8-bit).
- Khi đạt đến giá trị lớn nhất (11111111), bộ đếm sẽ quay lại 00000000 ở lần xung nhịp tiếp theo.

Cách 2:

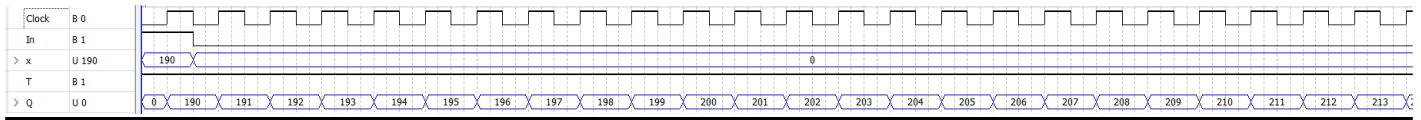
Code Thực Thi:

```
1  module Counter8bit(x, T, In, Clock, Q);
2  input T, Clock, In;
3  input [7:0]x;
4  output [7:0]Q;
5  CounterParameter(x[3:0], T, Clock, Q[3:0], In);
6  wire E2, E1;
7  nand(E1, Q[0], Q[1], Q[2], Q[3]);
8  Mux2to1_1bit(E2, E1, Clock, In);
9  CounterParameter(x[7:4], T, E2, Q[7:4], In);
10 endmodule
11
```

RTL Viewer:



Mô phỏng Waveform:



Giải thích:

1. Mô tả chức năng của mạch:

- **Clock:** Tín hiệu xung nhịp điều khiển bộ đếm. Mỗi cạnh lên của xung clock sẽ làm tăng giá trị của bộ đếm.
- **In:** Khi tín hiệu này được kích hoạt, mạch sẽ nạp giá trị từ đầu vào x vào thanh ghi đếm.
- **X[7:0]:** Đây là giá trị được nạp vào bộ đếm khi tín hiệu In được kích hoạt.
- **T:** Điều khiển hoạt động đếm (có thể là tín hiệu cho phép đếm).
- **Q [7:0]:** Đây là giá trị đầu ra của bộ đếm, biểu thị giá trị hiện tại của bộ đếm.

2. Cách hoạt động:

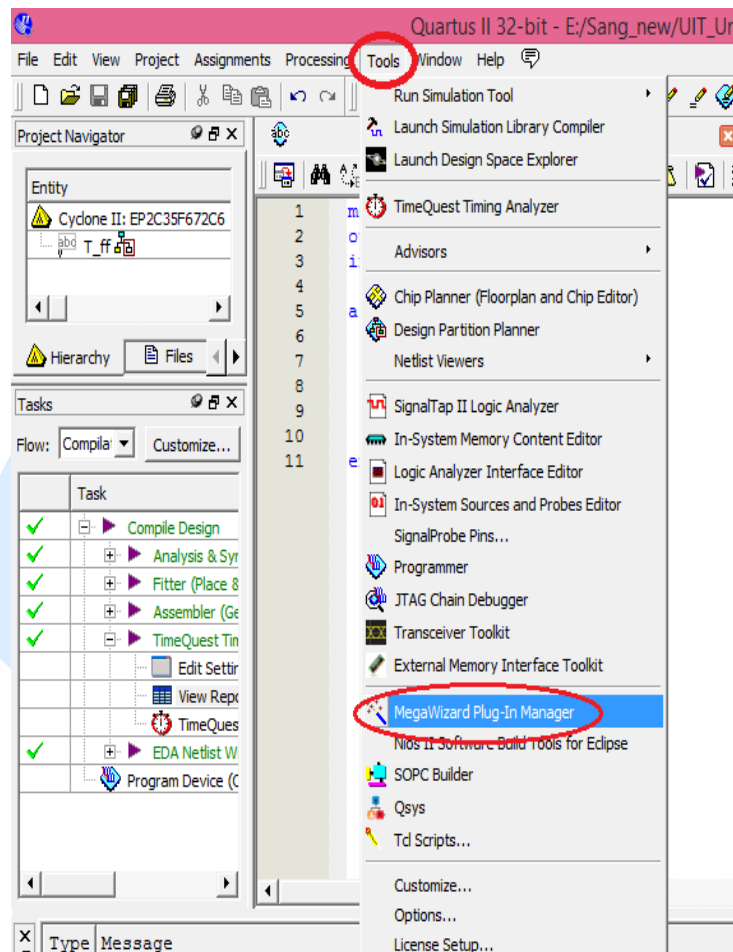
- Ban đầu, Q có giá trị 0.
- Khi In được kích hoạt, giá trị 190 được nạp vào Q.
- Sau đó, ở mỗi xung clock tiếp theo, Q tăng dần từ 190, 191, 192, ..., tiếp tục đếm lên.
- Khi không có tín hiệu In, bộ đếm tiếp tục tăng theo xung

clock mà không bị nạp lại giá trị mới

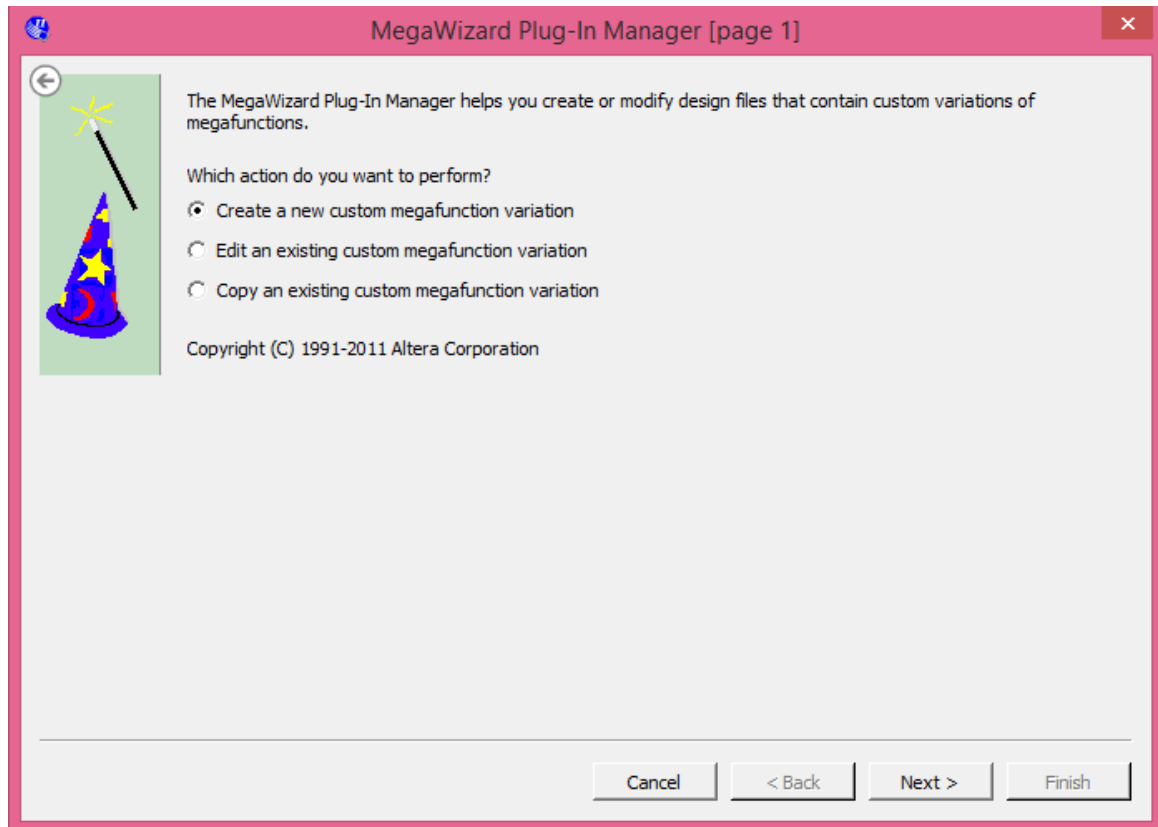
2.3. Thiết kế bộ đếm 8-bit bằng các sử dụng **LPM** (Library of Parameterized Modules) của Altera.

Để sử dụng LPM của Altera:

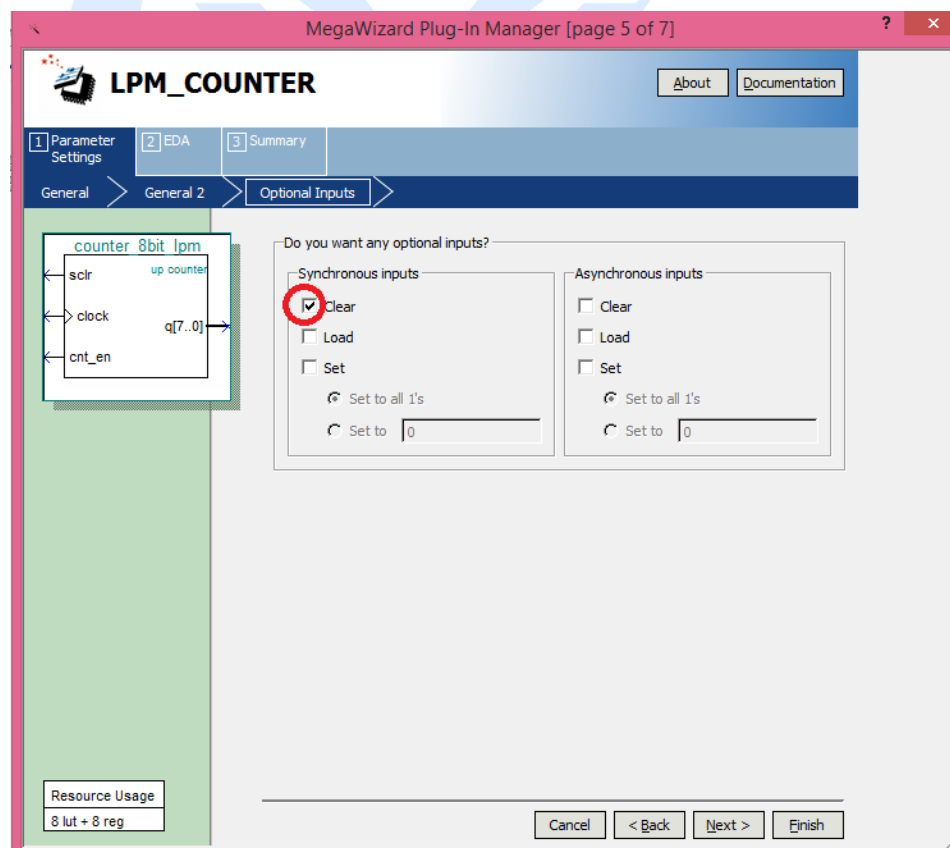
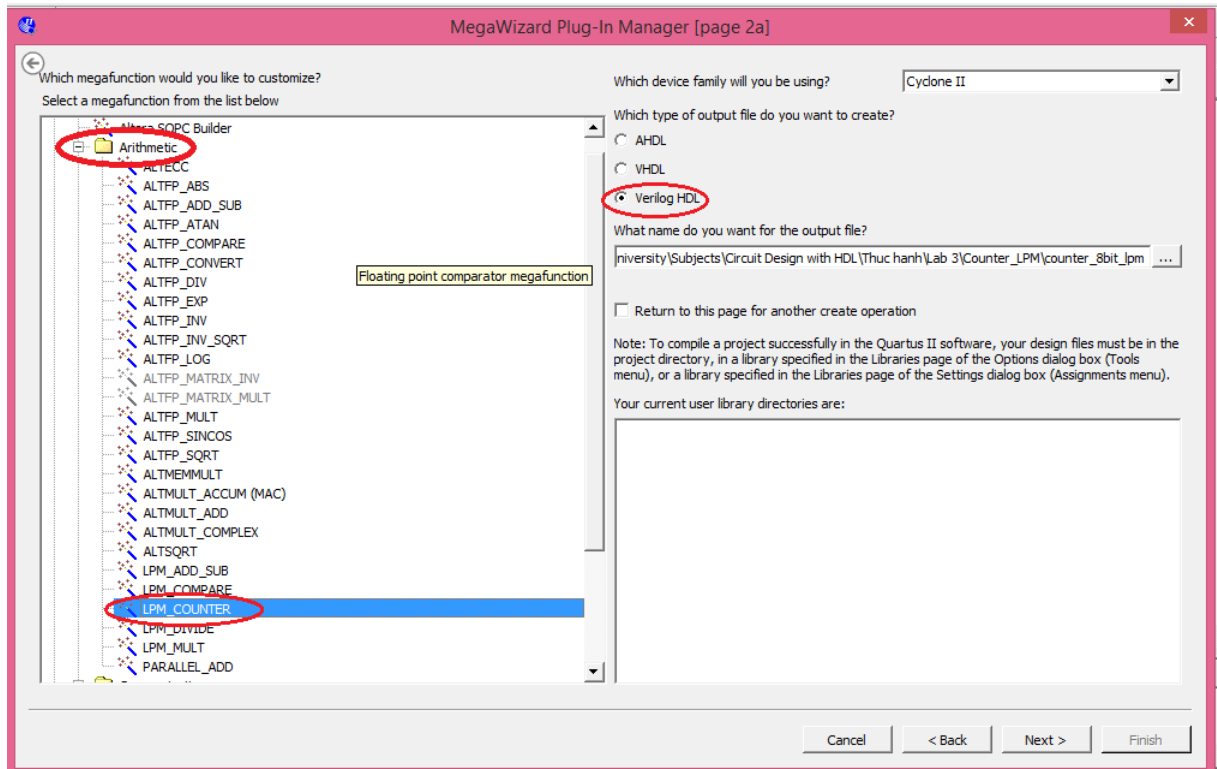
Mở phần mềm Quartus II → Chọn **Tool** → **MegaWizard Plug-in Manager**

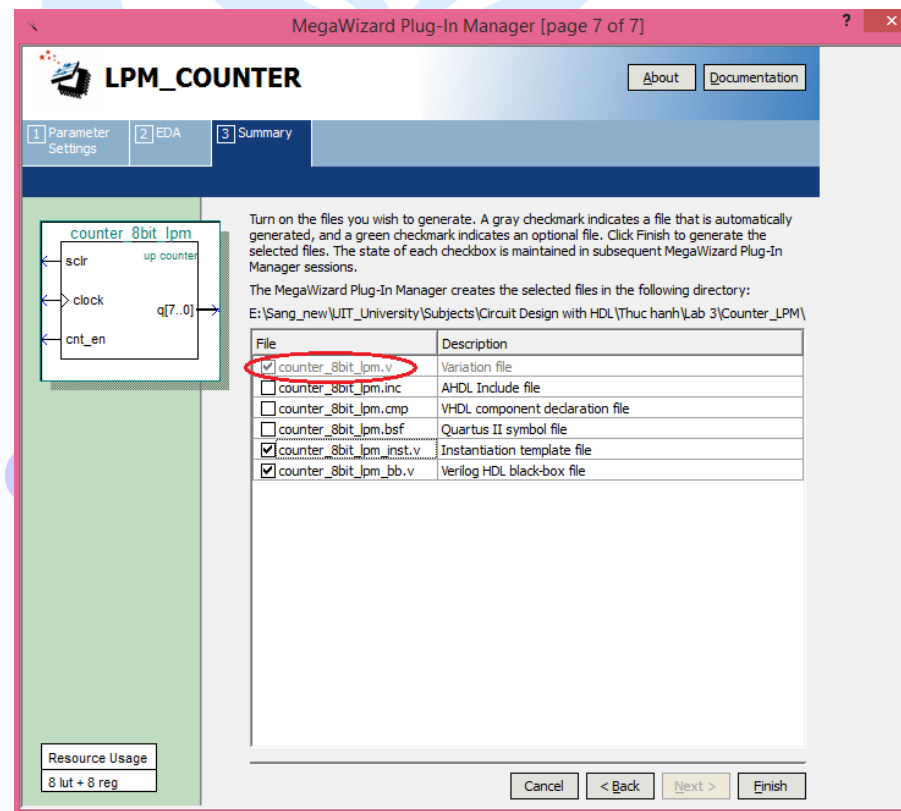
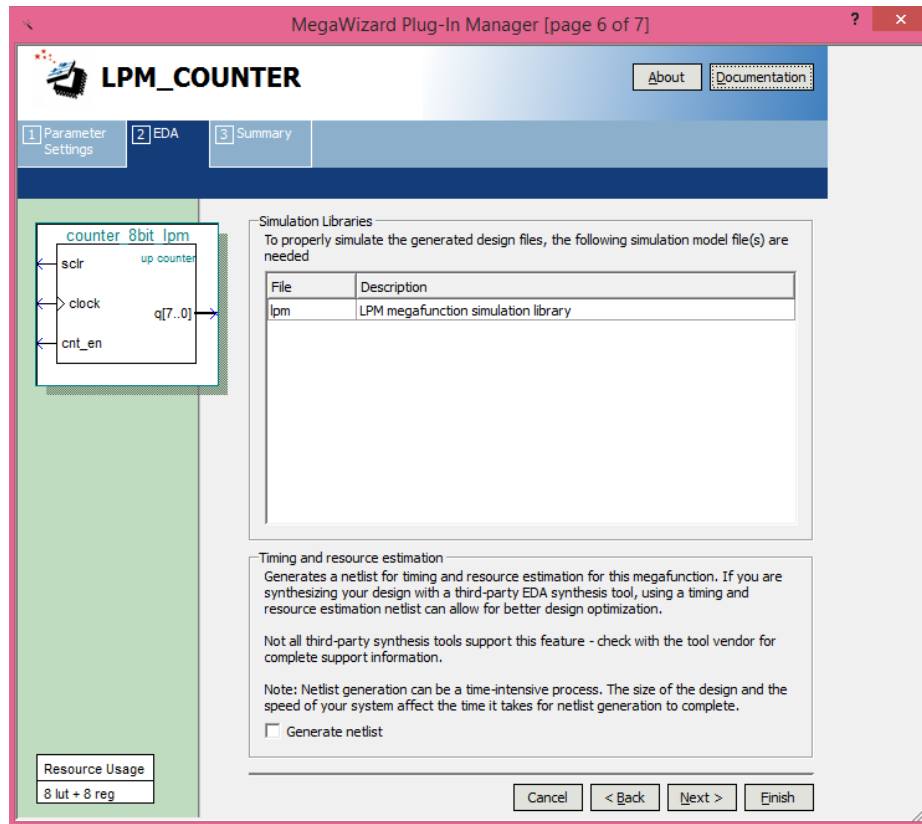


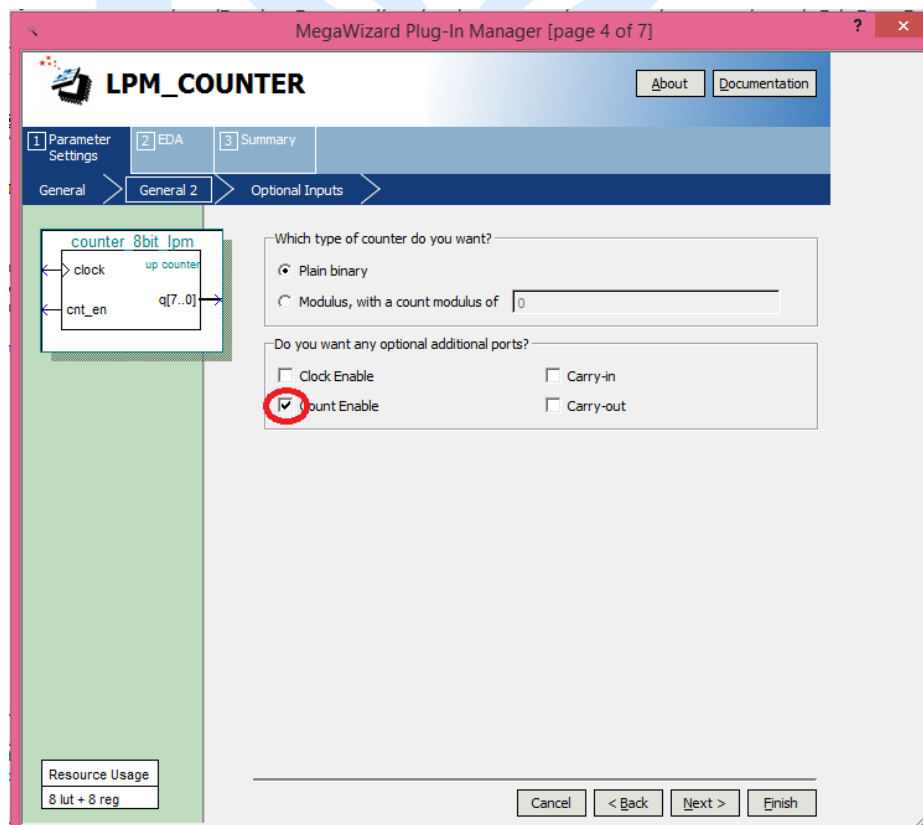
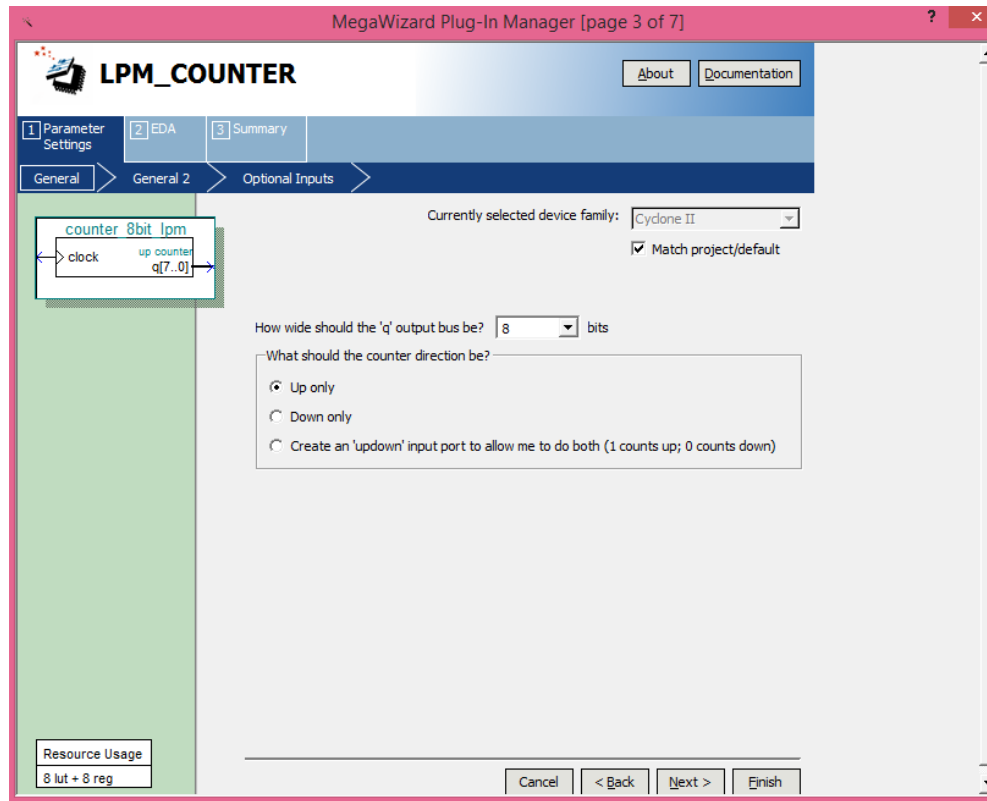
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN



UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN







- Thực hiện gán chân giống câu 2.2.
- So sánh tần số Fmax so với bộ đếm trong câu 2.2

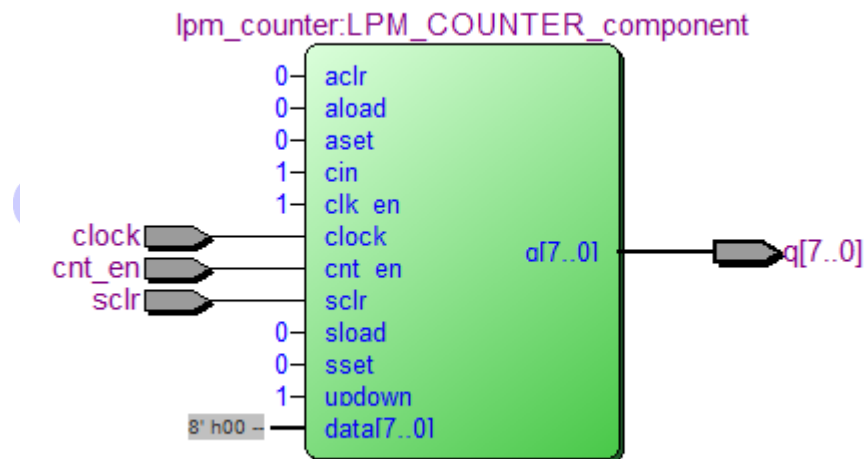
Lưu ý: Sinh viên tham khảo thêm cách sử dụng các LPM trong file “tut_lpms_verilog.pdf”

Làm theo hướng dẫn, ta được đoạn code thực thi như sau:

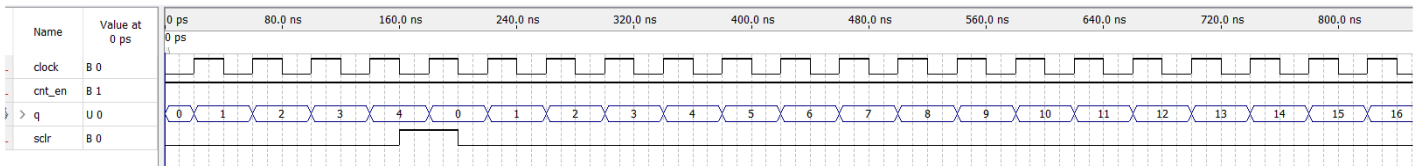
Code Thực Thi:

```
37 module counter_8bit_lpm (  
38     clock,  
39     cnt_en,  
40     sclr,  
41     q);  
42     input    clock;  
43     input    cnt_en;  
44     input    sclr;  
45     output   [7:0] q;  
46     wire [7:0] sub_wire0;  
47     wire [7:0] q = sub_wire0[7:0];  
48     lpm_counter LPM_COUNTER_component (  
49         .clock (clock),  
50         .cnt_en (cnt_en),  
51         .sclr (sclr),  
52         .q (sub_wire0),  
53         .aclr (1'b0),  
54         .aload (1'b0),  
55         .aset (1'b0),  
56         .cin (1'b1),  
57         .clk_en (1'b1),  
58         .cout (),  
59         .data ({8{1'b0}}),  
60         .eq (),  
61         .sload (1'b0),  
62         .sset (1'b0),  
63         .updown (1'b1));  
64     defparam  
65         LPM_COUNTER_component.lpm_direction = "UP",  
66         LPM_COUNTER_component.lpm_port_updown = "PORT_UNUSED",  
67         LPM_COUNTER_component.lpm_type = "LPM_COUNTER",  
68         LPM_COUNTER_component.lpm_width = 8;  
69 endmodule
```

RTL Viewer:



Mô phỏng Waveform:



So sánh tần số Fmax:

Tần số Fmax của câu 2.2:

Slow Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	434.59 MHz	380.08 MHz	clk	limit due to minimum period restriction (max I/O toggle rate)

Tần số Fmax của câu 2.3:

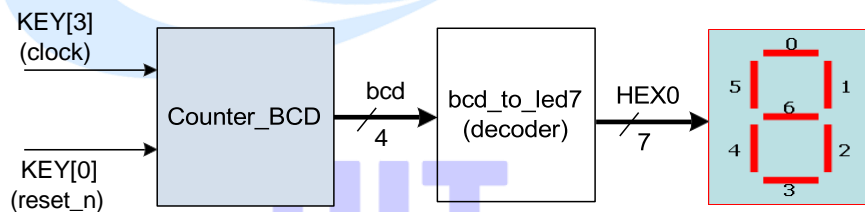
Slow Model Fmax Summary

	Fmax	Restricted Fmax	Clock Name	Note
1	434.59 MHz	380.08 MHz	clock	limit due to minimum period restriction (max I/O toggle rate)

→ Tần số của hai câu là bằng nhau

Câu 3.

3.1. Thiết kế mạch đếm BCD 0 đến 9 sao cho giá trị bộ đếm tăng lên 1 khi có cạnh lên của xung clock KEY[3].



Tín hiệu **reset_n** là tín hiệu reset bất đồng bộ và tích cực mức thấp

Yêu cầu: Sinh viên thực hiện bộ **decoder** bằng lệnh “**case ... endcase**”

Đầu tiên ta thiết kế 1 bộ Counter_BCD

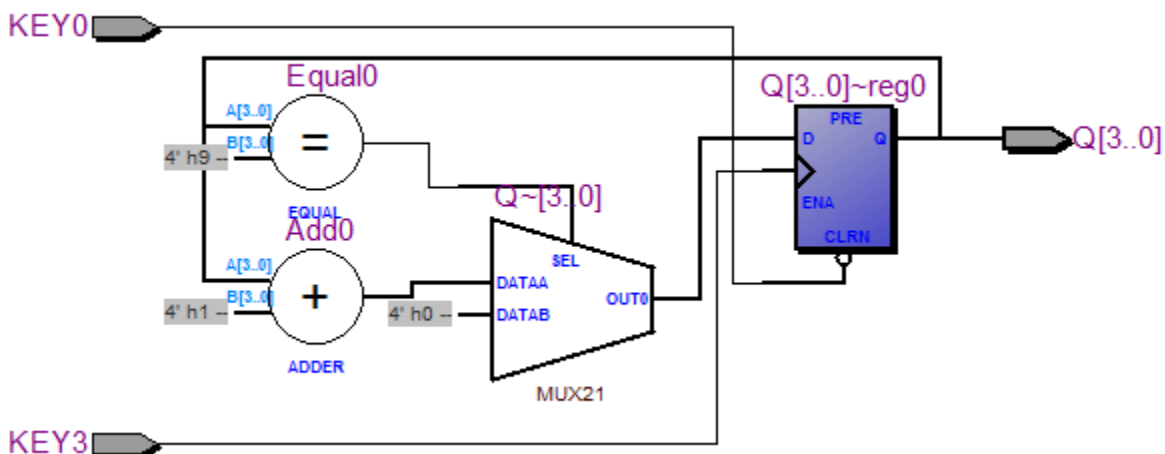
Code Thực Thi:

```

1  module CounterBCD(KEY3, KEY0, Q);
2  input wire KEY3, KEY0;
3  output reg [3:0] Q;
4  always @ (posedge KEY3 or negedge KEY0) begin
5      if (!KEY0)
6          Q <= 4'd0;
7      else if (Q == 4'd9)
8          Q <= 4'd0;
9      else
10         Q <= Q + 1;
11     end
12 endmodule

```

RTL Viewer:



Tiếp theo thiết kế 1 bộ bcd to 7leds:

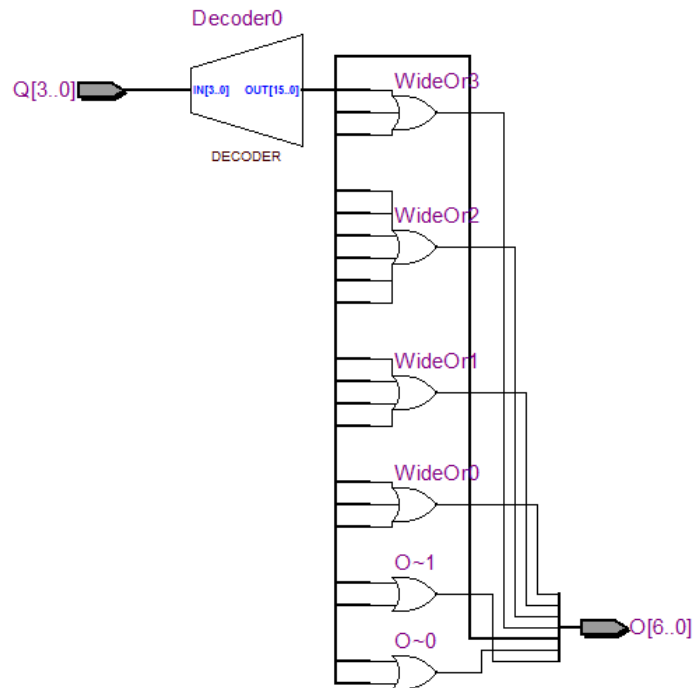
Code Thực Thi:



```
1 module BCD_Decoder(Q, O);
2   input wire [3:0] Q;
3   output reg [6:0] O;
4   always @ (*) begin
5     case (Q)
6       4'd0 : O = 7'b1000000;
7       4'd1 : O = 7'b1111001;
8       4'd2 : O = 7'b0100100;
9       4'd3 : O = 7'b0110000;
10      4'd4 : O = 7'b0011001;
11      4'd5 : O = 7'b0010010;
12      4'd6 : O = 7'b0000010;
13      4'd7 : O = 7'b1111000;
14      4'd8 : O = 7'b0000000;
15      4'd9 : O = 7'b0010000;
16      default: O = 7'b0000000;
17    endcase
18  end
19 endmodule
```

RTL Viewer:

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

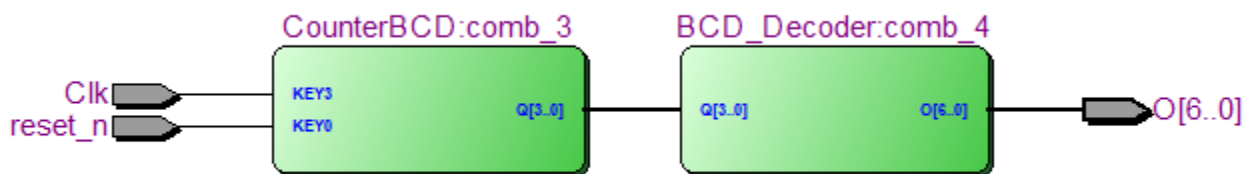


Cuối cùng kết hợp lại hai bộ ta được 1 bộ BCD Counter hoàn chỉnh

Code Thực Thi:

```
1 module FinalBCD(Clk, reset_n, O);
2   input wire Clk, reset_n;
3   output wire [6:0] O;
4   wire [3:0] n;
5   CounterBCD(Clk, reset_n, n);
6   BCD_Decoder(n, O);
7 endmodule
```

RTL Viewer:



Mô phỏng Waveform:



Clk	B 0	
> O	B 1000000	1000000 1111001 0100100 0110000 0011001 0010010 0000010 1111000 0000000 0010000
reset_n	B 1	

Giải thích:

Mạch trên kết hợp hai khối chức năng chính:

1. Bộ đếm BCD lên đến 9 (Counter_BCD)

- Đây là một bộ đếm nhị phân mã hóa thập phân (BCD), nghĩa là nó đếm từ 0000 (0) đến 1001 (9) rồi quay lại 0000.
- Ở hình trên, giá trị đầu ra O thay đổi tuần tự theo từng xung nhịp Clk, biểu diễn quá trình đếm.
- Giá trị reset_n ở mức cao (1), nghĩa là bộ đếm không bị reset và tiếp tục đếm bình thường.

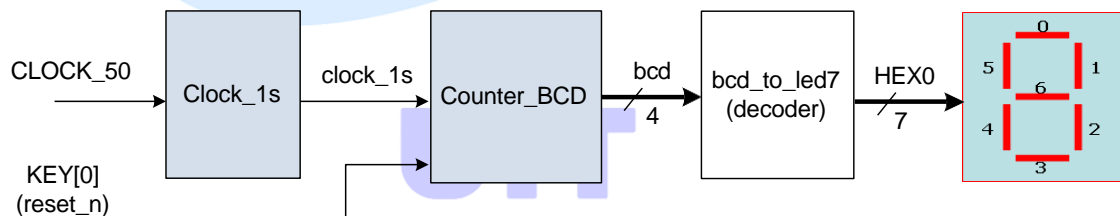
2. Bộ giải mã BCD sang LED 7 thanh (bcd_to_led7)

- Chuyển đổi giá trị BCD từ bộ đếm sang mã hiển thị phù hợp cho LED 7 đoạn.
- Các giá trị đầu ra O của bộ đếm được chuyển thành các mã LED tương ứng để hiển thị số từ 0-9

Loại động của mạch:

- Khi Clk tạo ra xung clock, bộ đếm Counter_BCD sẽ tăng giá trị từ 0 đến 9 theo hệ BCD.
- Mỗi giá trị đầu ra từ Counter_BCD được gửi đến bcd_to_led7, sau đó giải mã để điều khiển LED 7 đoạn, hiển thị số tương ứng (ví dụ 1 là 1111001).
- Khi bộ đếm đạt 1001 (9), chu trình lặp lại về 0000 (0).

3.2. Thiết kế mạch đếm BCD 0 đến 9 sao cho giá trị bộ đếm tăng lên 1 sau mỗi 1s

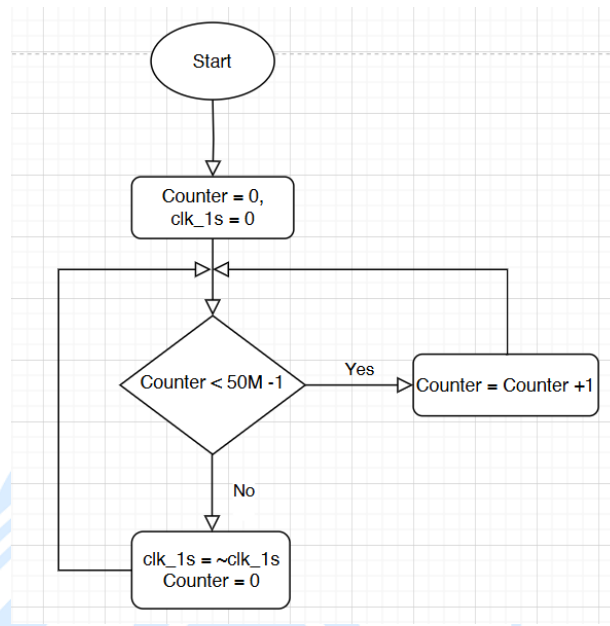


Sử dụng xung CLOCK_50 (50 MHz) để tạo xung clock_1s (1Hz)

Yêu cầu: SV vẽ lưu đồ giải thuật cho khối tạo xung "Clock_1s"

Tương tự với câu 3.1 nhưng ở đây có thêm Clock_1s, ta bắt đầu thiết kế:

Lưu đồ giải thuật:



Code Thực Thi:

```

1  module Clock_Divider (clk_50MHz, clk_1s);
2      input wire clk_50MHz;
3      output reg clk_1s;
4      reg [25:0] counter;
5      always @(posedge clk_50MHz) begin
6          if (counter == 50_000_000 - 1) begin
7              counter <= 0;
8              clk_1s <= ~clk_1s;
9          end else begin
10             counter <= counter + 1;
11         end
12     end
13 endmodule
    
```

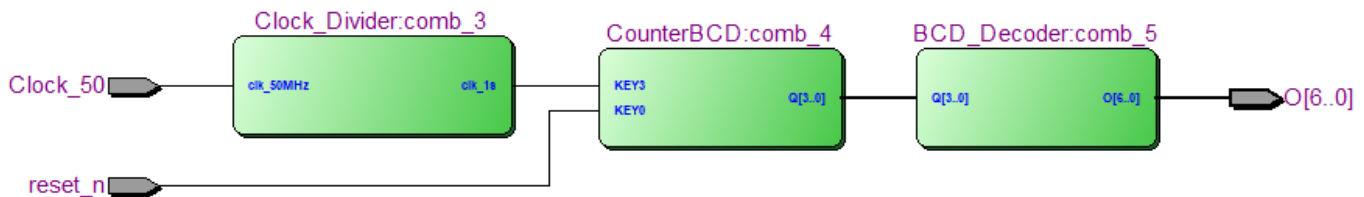
Kết hợp với 2 bộ BCD Counter và bcd to 7led đã làm ở trên ta được:

Code thực thi:



```
1 module BCDFinal_WithCD(Clock_50, reset_n, 0);
2 input Clock_50, reset_n;
3 output [6:0]O;
4 wire clk_1s;
5 wire [3:0] n;
6 Clock_Divider(Clock_50, clk_1s);
7 CounterBCD(clk_1s, reset_n, n);
8 BCD_Decoder(n, O);
9 endmodule
```

RTL Viewer:



Giải thích:

Ô tả hoạt động của hệ thống sau khi kết hợp 3 mô-đun:

1. **Clock_Divider** – Chia tần số từ 50 MHz xuống 1 Hz để làm clock cho bộ đếm.
2. **Counter_BCD** – Bộ đếm BCD tăng giá trị mỗi giây.
3. **bcd_to_led7** – Giải mã giá trị BCD từ bộ đếm và xuất ra dạng điều khiển LED 7 đoạn.

Hoạt động của hệ thống

1. **Tạo clock 1 Hz từ clock 50 MHz**
 - Clock_Divider giảm tần số từ 50 MHz xuống 1 Hz, tức là mỗi giây nó tạo một xung nhịp (clk_1s).
 - Clock này sẽ điều khiển bộ đếm.
2. **Bộ đếm BCD hoạt động**
 - Bộ đếm nhận clk_1s làm clock.
 - Mỗi xung clock 1 Hz đến, bộ đếm tăng giá trị từ 0000 (0) đến 1001 (9), sau đó quay lại 0000.
 - Nếu reset_n được kích hoạt (0), bộ đếm được đặt lại về 0000.
3. **Giải mã giá trị BCD ra LED 7 đoạn**
 - Giá trị BCD từ bộ đếm (bcd_out) được gửi đến bcd_to_led7.
 - bcd_to_led7 chuyển đổi mã BCD thành tín hiệu điều khiển LED 7 đoạn.
 - LED sẽ hiển thị số tương ứng với giá trị của bộ đếm, thay đổi mỗi giây.

Câu 4.

4.1. Thực hiện bảng chạy chữ như mô tả ở hình dưới. Sử dụng HEX7-0 để hiển thị kí tự.

Sử dụng:

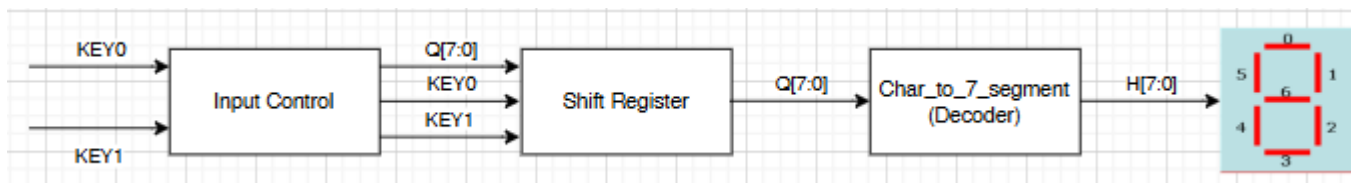
- KEY[0] làm xung clock để điều khiển sự dịch chuyển.
- KEY[1] làm tín hiệu reset của mạch (khi reset tích cực, thì các HEX hiển thị dòng chữ tại

Clock cycle 0)

Yêu cầu: SV thiết kế SPEC (sơ đồ khối) để thực hiện mạch trên

Clock cycle	Character pattern							
	H7	H6	H5	H4	H3	H2	H1	H0
0				H	E	L	L	O
1			H	E	L	L	O	
2		H	E	L	L	O		
3	H	E	L	L	O			
4	E	L	L	O				H
5	L	L	O				H	E
6	L	O				H	E	L
7	O				H	E	L	L
8				H	E	L	L	O
...	and so on							

Sơ đồ khối:



TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN



Code Thực Thi:

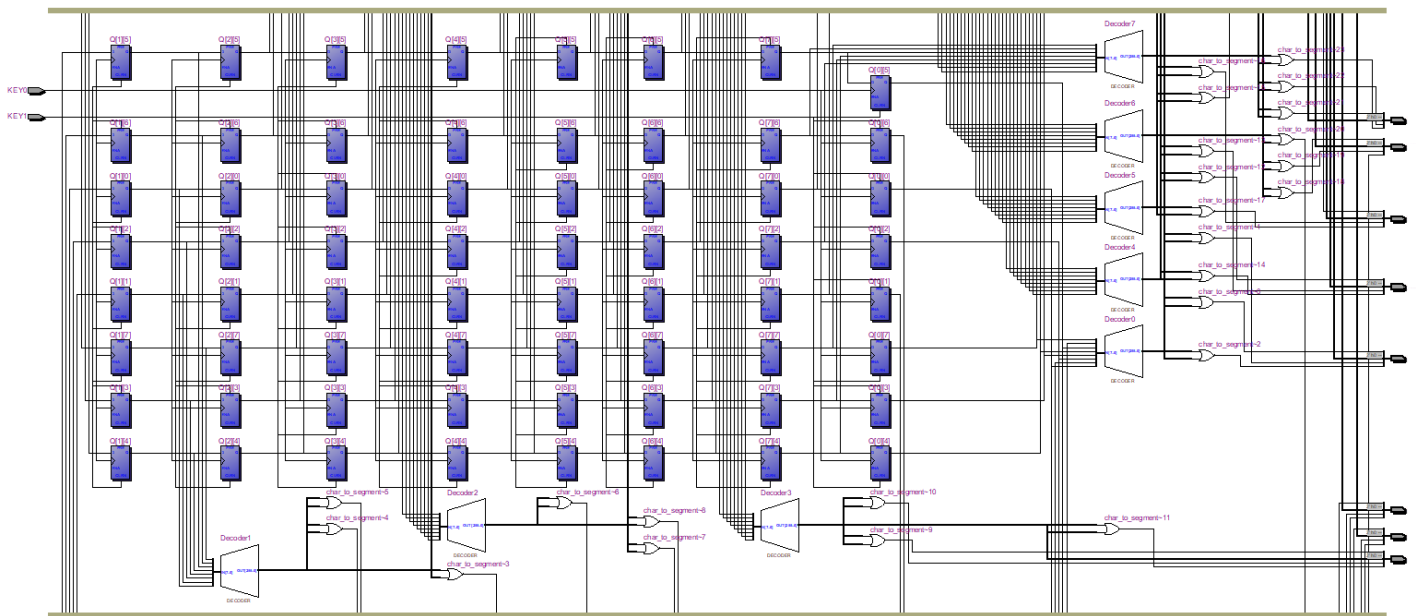
```

1  module BCD_Board(KEY1, KEY0, H0, H1, H2, H3, H4, H5, H6, H7);
2  input KEY1, KEY0;
3  output [6:0] H0;
4  output [6:0] H1;
5  output [6:0] H2;
6  output [6:0] H3;
7  output [6:0] H4;
8  output [6:0] H5;
9  output [6:0] H6;
10 output [6:0] H7;
11 function [6:0] char_to_segment;
12     input [7:0] char;
13     case (char)
14         "H" : char_to_segment = 7'b0001001;
15         "E" : char_to_segment = 7'b0000110;
16         "L" : char_to_segment = 7'b1000111;
17         "O" : char_to_segment = 7'b1000000;
18         default: char_to_segment = 7'b0000000;
19     endcase
20 endfunction
21 reg [7:0] Q [7:0];
22 initial begin
23     Q[7] = " ";
24     Q[6] = " ";
25     Q[5] = " ";
26     Q[4] = "H";
27     Q[3] = "E";
28     Q[2] = "L";
29     Q[1] = "L";
30     Q[0] = "O";
31 end
32 always @ (posedge KEY0 or posedge KEY1) begin
33     if(KEY1) begin
34         Q[7] <= " ";
35         Q[6] <= " ";
36         Q[5] <= " ";
37         Q[4] <= "H";
38         Q[3] <= "E";
39         Q[2] <= "L";
40         Q[1] <= "L";
41         Q[0] <= "O";
42     end else begin
43         Q[0] <= Q[7];
44         Q[1] <= Q[0];
45         Q[2] <= Q[1];
46         Q[3] <= Q[2];
47         Q[4] <= Q[3];
48         Q[5] <= Q[4];
49         Q[6] <= Q[5];
50         Q[7] <= Q[6];
51     end
52 end
53 assign H0 = char_to_segment(Q[0]);
54 assign H1 = char_to_segment(Q[1]);
55 assign H2 = char_to_segment(Q[2]);
56 assign H3 = char_to_segment(Q[3]);
57 assign H4 = char_to_segment(Q[4]);
58 assign H5 = char_to_segment(Q[5]);
59 assign H6 = char_to_segment(Q[6]);
60 assign H7 = char_to_segment(Q[7]);
61 endmodule

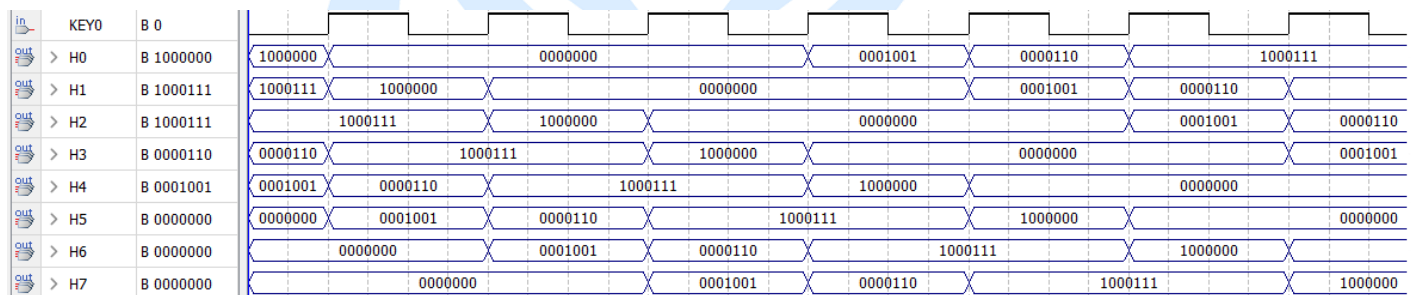
```

IC
G TIN

RTL Viewer:



Mô phỏng Waveform:



Giải thích:

Module BCD_Board thực hiện dịch chuyển chữ "HELLO" sang phải trên 8 màn hình LED 7 đoạn, điều khiển bởi hai nút:

- **KEY0:** Dịch chuyển chuỗi sang phải một bước mỗi khi nhấn.
- **KEY1:** Reset chuỗi về trạng thái ban đầu ("HELLO" căn trái)
- Hàm char_to_segment chuyển ký tự sang mã LED 7 đoạn.
- Thanh ghi Q[7:0] lưu trạng thái hiển thị.
- Mỗi lần nhấn KEY0, chuỗi dịch sang phải.
 - "H" chuyển lên LED kế bên, "O" tiến dần ra ngoài.
 - Khi "HELLO" trượt ra khỏi LED, nó sẽ biến mất.
 - Dịch chuyển tiếp tục cho đến khi tất cả LED trống.
- Nhấn KEY1, chuỗi trở về "HELLO".



4.2. Dùng CLOCK_50 để thay KEY[0] trong câu a và điều khiển mạch sao cho các HEX sẽ tự động dịch chuyển sau khoảng thời gian 1s

Cũng giống 4.1, nhưng ở đây ta thêm vào 1 bộ Clock 1s đã được thiết kế ở câu 3:

Code Thực Thi:

```

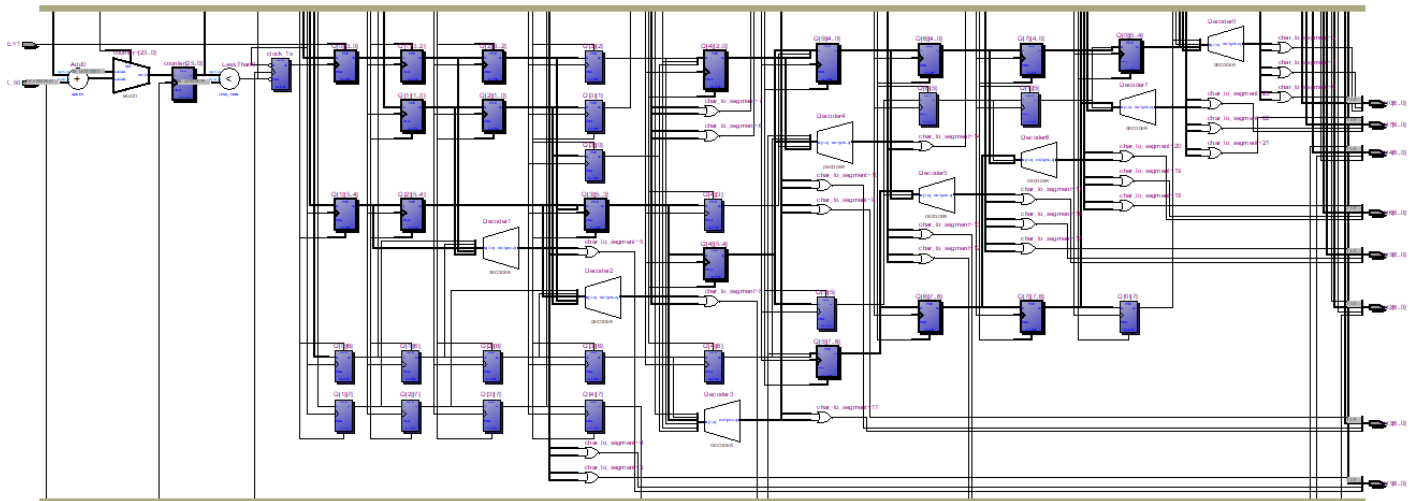
1 module BCD_Board_CLOCK(
2     input CLOCK_50,
3     input KEY1,
4     output [6:0] H0,
5     output [6:0] H1,
6     output [6:0] H2,
7     output [6:0] H3,
8     output [6:0] H4,
9     output [6:0] H5,
10    output [6:0] H6,
11    output [6:0] H7
12 );
13
14 reg [7:0] Q [7:0];
15 reg clock_1s;
16 reg [25:0] counter = 0;
17 always @(posedge CLOCK_50) begin
18     if (counter < 50_000_000 - 1)
19         counter <= counter + 1;
20     else begin
21         counter <= 0;
22         clock_1s <= ~clock_1s;
23     end
24 end
25 function automatic [6:0] char_to_segment(input [7:0] char);
26 case (char)
27     "H" : char_to_segment = 7'b0001001;
28     "E" : char_to_segment = 7'b0000110;
29     "L" : char_to_segment = 7'b1000111;
30     "O" : char_to_segment = 7'b1000000;
31     default: char_to_segment = 7'b0000000;
32 endcase
33 endfunction
34
35 initial begin
36     Q[7] = " ";
37     Q[6] = " ";
38     Q[5] = " ";
39     Q[4] = "H";
40     Q[3] = "E";
41     Q[2] = "L";
42     Q[1] = "L";
43     Q[0] = "O";
44 end
45 always @(posedge clock_1s or posedge KEY1) begin
46     if(KEY1) begin
47
48         Q[7] <= " ";
49         Q[6] <= " ";
50         Q[5] <= " ";
51         Q[4] <= "H";
52         Q[3] <= "E";
53         Q[2] <= "L";
54         Q[1] <= "L";
55         Q[0] <= "O";
56     end else begin
57         Q[0] <= Q[7];
58         Q[1] <= Q[0];
59         Q[2] <= Q[1];
60         Q[3] <= Q[2];
61         Q[4] <= Q[3];
62         Q[5] <= Q[4];
63         Q[6] <= Q[5];
64         Q[7] <= Q[6];

```

```

65     end
66     end
67     assign H0 = char_to_segment(Q[0]);
68     assign H1 = char_to_segment(Q[1]);
69     assign H2 = char_to_segment(Q[2]);
70     assign H3 = char_to_segment(Q[3]);
71     assign H4 = char_to_segment(Q[4]);
72     assign H5 = char_to_segment(Q[5]);
73     assign H6 = char_to_segment(Q[6]);
74     assign H7 = char_to_segment(Q[7]);
75 endmodule
    
```

RTL Viewer:



Giải thích:

Mạch này là một bộ hiển thị chữ chạy trên 8 LED 7 đoạn, hiển thị dòng chữ "HELLO" di chuyển từ phải sang trái mỗi giây. Khi nhấn KEY1, chữ sẽ được đặt lại về vị trí ban đầu.

Loại động tổng thể của mạch

1. **Ban đầu:**
 - LED hiển thị "HELLO" ở 5 vị trí cuối.
 - 3 LED đầu tiên trống.
2. **Sau mỗi giây (clock_1s):**
 - Ký tự dịch sang trái.
 - "H" chuyển lên LED kế bên, "O" tiến dần ra ngoài.
 - Khi "HELLO" trượt ra khỏi LED, nó sẽ biến mất.
 - Dịch chuyển tiếp tục cho đến khi tất cả LED trống.
 - **Sau 8 giây:** Tất cả LED tắt vì chữ "HELLO" đã di chuyển hết.
3. **Nhấn KEY1 bất cứ lúc nào:**
 - Chữ "HELLO" trở lại vị trí ban đầu, sẵn sàng dịch tiếp.



Câu 5.

Thiết kế bộ đếm 2-digit BCD counter đếm các giá trị từ 00 đến 20. Giá trị đếm được hiển thị lên hai Led 7 đoạn (HEX0 và HEX1). Bộ đếm có thể được Reset (bắt đồng bộ) về 0 khi KEY[0] được nhấn. Mỗi giá trị đếm được hiển thị trong 1s, sử dụng CLOCK_50 là giá trị xung clock tham khảo. Kiểm tra thiết kế trên board DE2.

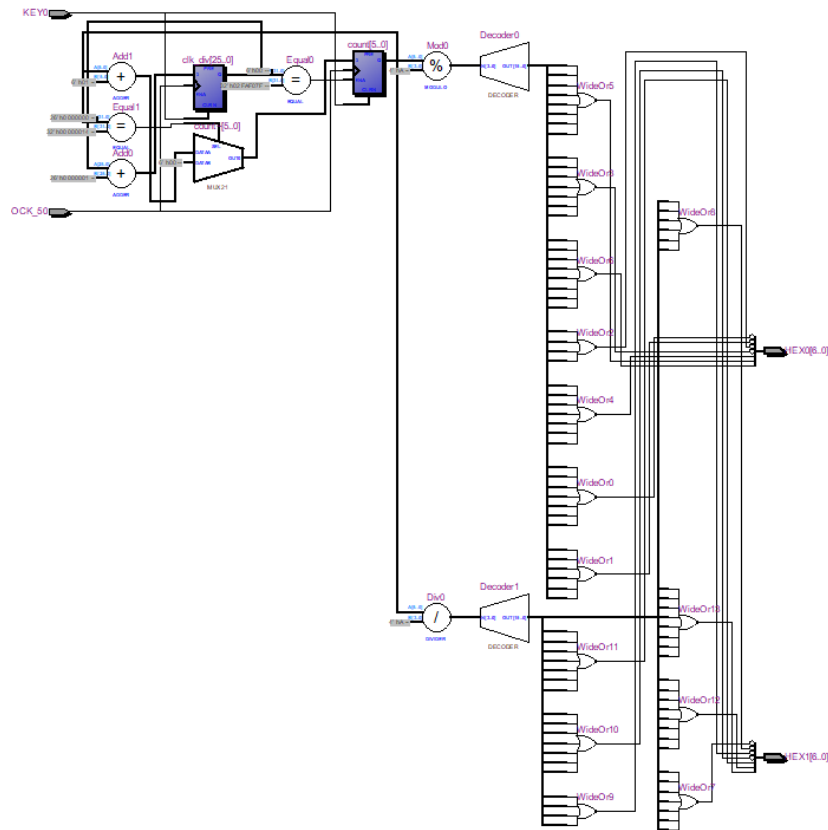
Code Thực Thi:

```

1  module TwoDigitBCDCounter(CLOCK_50, HEX0, HEX1, KEY0);
2  input CLOCK_50, KEY0;
3  output [6:0] HEX0;
4  output [6:0] HEX1;
5  wire clk_1Hz;
6  reg [5:0] count;
7  reg [25:0] clk_div;
8  always @ (posedge CLOCK_50 or posedge KEY0) begin
9      if (KEY0)
10         clk_div <= 0;
11     else if (CLOCK_50 == 50_000_000 - 1)
12         clk_div <= 0;
13     else
14         clk_div <= clk_div + 1;
15 end
16 assign clk_1Hz = (clk_div == 50_000_000 - 1) ? 1 : 0;
17
18 always @ (posedge CLOCK_50 or posedge KEY0) begin
19     if (KEY0)
20         count <= 0;
21     else if (clk_1Hz) begin
22         if (count == 20)
23             count <= 0;
24         else
25             count <= count + 1;
26     end
27 end
28 function [6:0] bcd_7_segment;
29 input [3:0] bcd;
30 case (bcd)
31     4'd0: bcd_7_segment = 7'b1000000;
32     4'd1: bcd_7_segment = 7'b1111001;
33     4'd2: bcd_7_segment = 7'b0100100;
34     4'd3: bcd_7_segment = 7'b0110000;
35     4'd4: bcd_7_segment = 7'b0011001;
36     4'd5: bcd_7_segment = 7'b0010010;
37     4'd6: bcd_7_segment = 7'b0000010;
38     4'd7: bcd_7_segment = 7'b1111000;
39     4'd8: bcd_7_segment = 7'b0000000;
40     4'd9: bcd_7_segment = 7'b0010000;
41     default: bcd_7_segment = 7'b1111111;
42 endcase
43 endfunction
44 assign HEX0 = bcd_7_segment(count % 10);
45 assign HEX1 = bcd_7_segment(count / 10);
46 endmodule

```

RTL Viewer:



Giải thích:

Chức năng:

Mạch này là một bộ đếm BCD 2 chữ số (00 - 20) hiển thị trên 2 LED 7 đoạn (HEX1 cho hàng chục, HEX0 cho hàng đơn vị). Nó đếm với tần số 1Hz và có thể reset bằng nút KEY0.

Nguyên lý hoạt động:

1. Tạo xung nhịp 1Hz:

- Bộ chia tần clk_div giảm từ 50MHz xuống 1Hz.
- Khi clk_div đạt 50 triệu xung (50_000_000 - 1), tín hiệu clk_1Hz kích hoạt một lần mỗi giây.

2. Bộ đếm count (0 - 20):

- Tăng giá trị mỗi giây.
- Khi đạt 20, reset về 0.
- Nếu nhấn KEY0, bộ đếm reset về 0 ngay lập tức.



3. Hiển thị lên LED 7 đoạn (HEX0, HEX1):

- HEX0 hiển thị hàng đơn vị (count % 10).
- HEX1 hiển thị hàng chục (count / 10).
- Dùng hàm bcd_7_segment để chuyển đổi từ số BCD sang mã điều khiển LED 7 đoạn.

> Mạch này hoạt động như một bộ đếm giây từ 00 đến 20, có thể reset bằng nút nhấn.

Câu 6.

Hiện thực một đồng hồ hiển thị giờ, phút, giây trong ngày. Đồng hồ sẽ thể hiện giá trị “giờ” (từ 0 đến 23) lên các led 7-đoạn HEX7-6, giá trị “phút” (từ 0 đến 60) lên các led HEX5-4, và giá trị “giây” (từ 0 đến 60) lên các led HEX3-2. Sử dụng các SW15-0 để reset lại giá trị “giờ” và “phút” cho đồng hồ.

Mạch hiện thực phải có khả năng báo lỗi hoặc không cho thiết lập các giá trị giờ, phút, giây bất hợp lý. Kiểm tra thiết kế trên board DE2.

Code Thực Thi:

```

1  module digital_clock(HEX2, HEX3, HEX4, HEX5, HEX6, HEX7, clk, reset, SW);
2  input clk, reset;
3  input [15:0] SW;
4  output [6:0] HEX2, HEX3, HEX4, HEX5, HEX6, HEX7;
5  reg [5:0] second;
6  reg [5:0] minute;
7  reg [4:0] hour;
8  reg [25:0] counter;
9  function [6:0] bcd_7_segment;
10 |   input [3:0] bcd;
11 |   case(bcd)
12 |       4'd0: bcd_7_segment = 7'b1000000;
13 |       4'd1: bcd_7_segment = 7'b1111001;
14 |       4'd2: bcd_7_segment = 7'b0100100;
15 |       4'd3: bcd_7_segment = 7'b0110000;
16 |       4'd4: bcd_7_segment = 7'b0011001;
17 |       4'd5: bcd_7_segment = 7'b0010010;
18 |       4'd6: bcd_7_segment = 7'b0000010;
19 |       4'd7: bcd_7_segment = 7'b1111000;
20 |       4'd8: bcd_7_segment = 7'b0000000;
21 |       4'd9: bcd_7_segment = 7'b0010000;
22 |       default: bcd_7_segment = 7'b1111111;
23 |   endcase
24 | endfunction
25 always @ (posedge clk or posedge reset) begin
26 |   if(reset) begin
27 |       second <= 0;
28 |       minute <= 0;
29 |       hour <= 0;
30 |   end else begin
31 |       counter <= counter + 26'd1;
32 |       if(SW[15]) begin

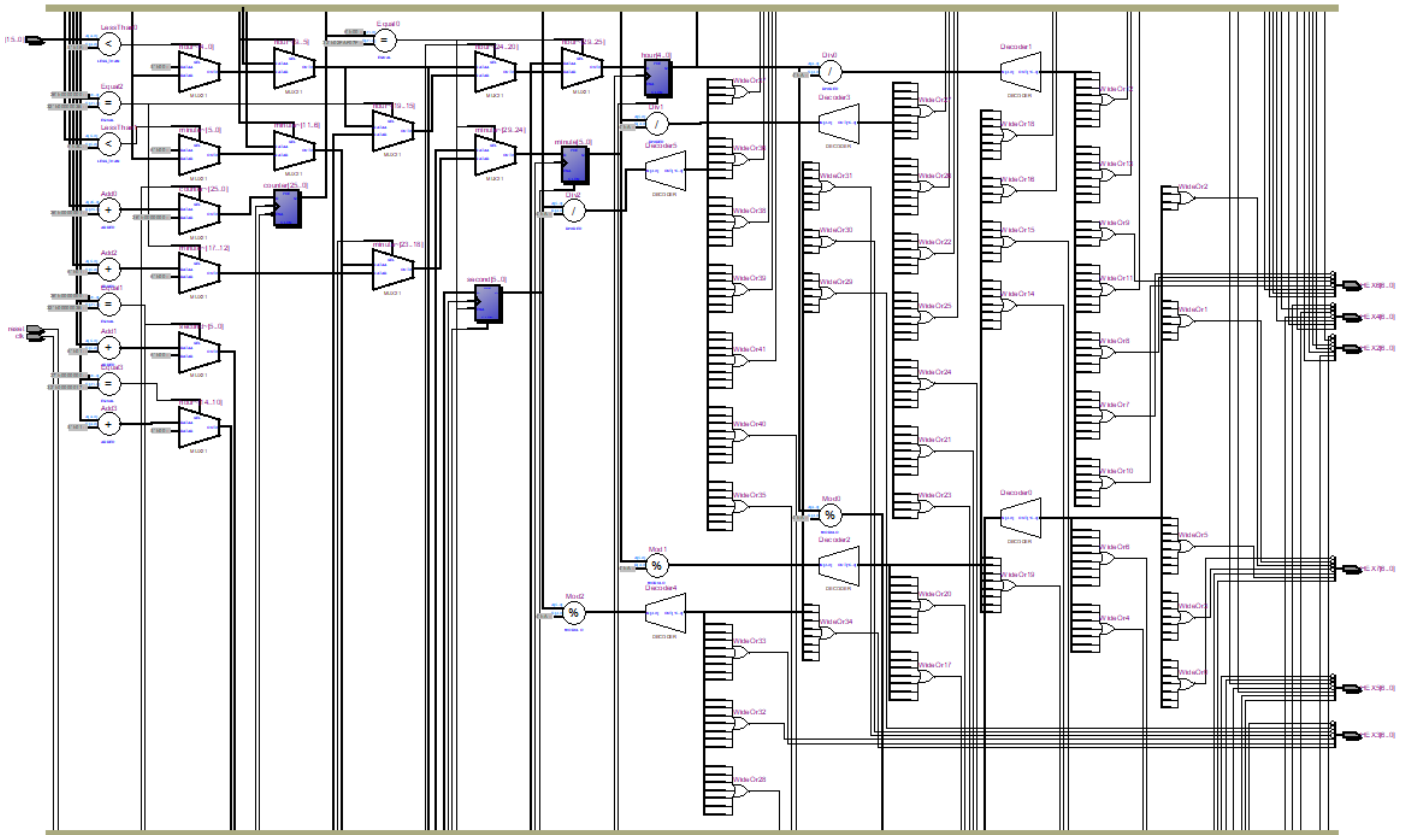
```



```
33         if(SW[10:6] < 24)
34             hour <= SW[10:6];
35         else
36             hour <= 0;
37     end if(SW[14]) begin
38         if(SW[5:0] < 60)
39             minute <= SW[5:0];
40         else
41             minute <= 0;
42     end if(counter == 50_000_000 - 1) begin
43         counter <= 0;
44         second <= second + 6'd1;
45         if(second == 59) begin
46             second <= 0;
47             minute <= minute + 6'd1;
48             if(minute == 59) begin
49                 minute <= 0;
50                 hour <= hour + 5'd1;
51                 if(hour == 23) begin
52                     hour <= 0;
53                 end
54             end
55         end
56     end
57 end
58
59 assign HEX7 = bcd_7_segment(hour % 10);
60 assign HEX6 = bcd_7_segment(hour / 10);
61 assign HEX5 = bcd_7_segment(minute % 10);
62 assign HEX4 = bcd_7_segment(minute / 10);
63 assign HEX3 = bcd_7_segment(second % 10);
64
65 assign HEX2 = bcd_7_segment(second / 10);
66 endmodule
```

UIT
TRƯỜNG ĐẠI HỌC
CÔNG NGHỆ THÔNG TIN

RTL Viewer:



Giải thích:

Chức năng:

Mạch này là một đồng hồ số hiển thị giờ, phút, giây trên 6 LED 7 đoạn (HEX7 - HEX2). Nó có khả năng đến thời gian thực và cho phép cài đặt giờ, phút bằng công tắc SW.

Mạch hoạt động:

- Tạo xung nhịp 1Hz từ clk:**
 - Bộ chia tần counter đếm đến 50 triệu (50_000_000 - 1) để tạo xung nhịp 1Hz (mỗi giây đếm lên 1 đơn vị).
- Bộ đếm thời gian (hour, minute, second):**
 - Mỗi giây: second tăng lên 1 đơn vị.
 - Khi second = 59: Reset về 0, minute tăng lên 1.
 - Khi minute = 59: Reset về 0, hour tăng lên 1.
 - Khi hour = 23: Reset về 0 (chu kỳ 24 giờ).
- Cài đặt giờ, phút bằng công tắc SW:**
 - Nếu SW[15] bật → Giờ (hour) được gán từ SW[10:6] (chỉ nhận giá trị hợp lệ <24).
 - Nếu SW[14] bật → Phút (minute) được gán từ SW[5:0] (chỉ nhận giá trị hợp lệ <60).
- Hiển thị thời gian lên LED 7 đoạn (HEX7 - HEX2):**
 - HEX7, HEX6: Giờ (hour).
 - HEX5, HEX4: Phút (minute).
 - HEX3, HEX2: Giây (second).
 - Sử dụng hàm bcd_7_segment để chuyển đổi số BCD sang mã điều khiển LED 7 đoạn.



> **Kết quả:** Đồng hồ số hoạt động chuẩn, có thể đếm thời gian thực và cài đặt giờ phút bằng SW.

TÀI LIỆU THAM KHẢO

Digital_Logics lab của Altera.

