ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA KỸ THUẬT MÁY TÍNH

BÁO CÁO BÀI THỰC HÀNH LAB2

HỌ VÀ TÊN: **NGUYỄN QUỐC TRƯỜNG AN – 21521810**

LỚP: **CE213.O11.2**

GIẢNG VIÊN HƯỚNG DẪN:

HỒ NGỌC DIỄM

TP. HỒ CHÍ MINH – Tháng 10 năm 2023

BÀI THỰC HÀNH SỐ 2

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:

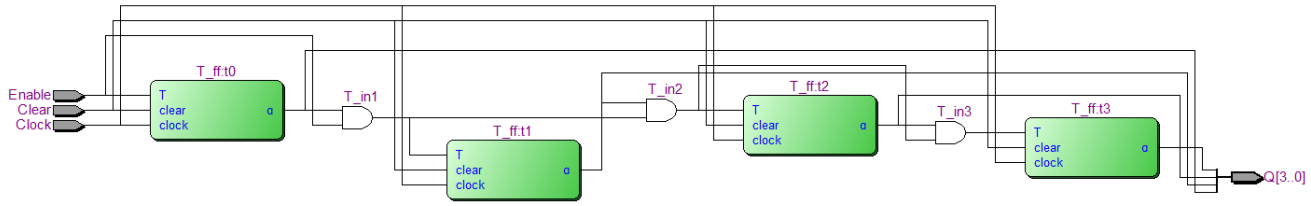
1.1: Thiết kế bộ đếm 4-bit

a) Code

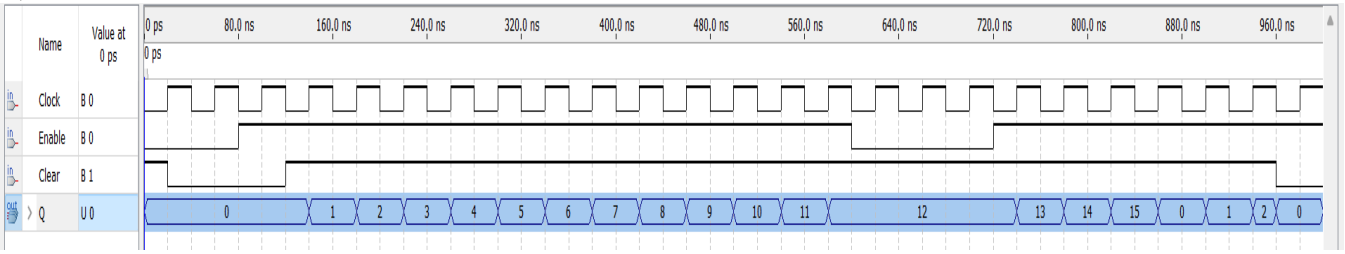
```
1  module T_ff (q,
2               clock,
3               T,
4               clear
5               );
6
7     output q;
8
9     input clock;
10    input T;
11    input clear;
12
13    reg q;
14
15    always @(posedge clock or negedge clear) begin
16
17       if (!clear)
18          q = 0;
19
20       else if (T)
21          q = ~q;
22
23       else
24          q = q;
25
26    end
27
28 endmodule
```

```
29
30  module counter_4bit (Q,
31                       Clock,
32                       Enable,
33                       Clear
34                       );
35
36      output [3:0] Q;
37
38      input Clock;
39      input Enable;
40      input Clear;
41
42      reg T_in1;
43      reg T_in2;
44      reg T_in3;
45
46      T_ff t0 (.q(Q[0]), .clock(Clock), .T(Enable), .clear(Clear));
47      T_ff t1 (.q(Q[1]), .clock(Clock), .T(T_in1), .clear(Clear));
48      T_ff t2 (.q(Q[2]), .clock(Clock), .T(T_in2), .clear(Clear));
49      T_ff t3 (.q(Q[3]), .clock(Clock), .T(T_in3), .clear(Clear));
50
51      always @(*) begin
52
53          T_in1 = Enable & Q[0];
54          T_in2 = T_in1 & Q[1];
55          T_in3 = T_in2 & Q[2];
56
57      end
58
59  endmodule
60
```

b) Netlist



b) Simulation

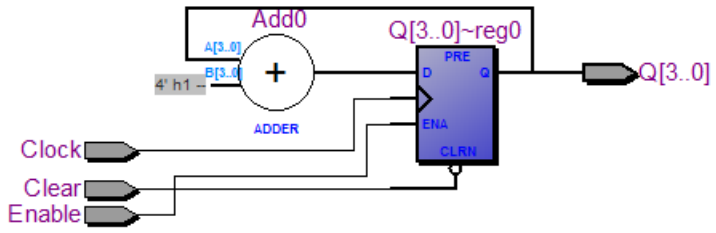


1.2: Thiết kế bộ đếm 4-bit như trong câu 1.1, nhưng mô tả ở mức Behavior

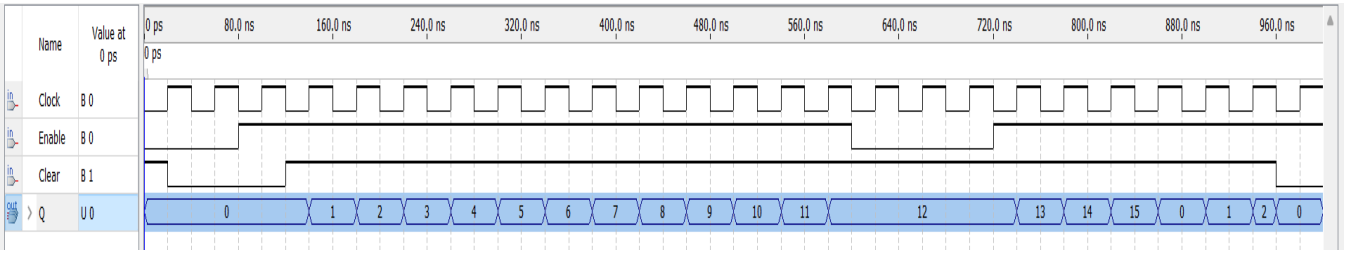
a) Code

```
1  module counter_behavior (Q,
2                           Clock,
3                           Enable,
4                           Clear
5                           );
6
7     output [3:0] Q;
8
9     input Clock;
10    input Enable;
11    input Clear;
12
13    reg [3:0] Q;
14
15    always @(posedge Clock or negedge Clear) begin
16
17       if(!Clear)
18          Q <= 0;
19
20       else if(Enable)
21          Q <= Q + 1;
22
23    end
24
25 endmodule
```

b) Netlist



b) Simulation



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

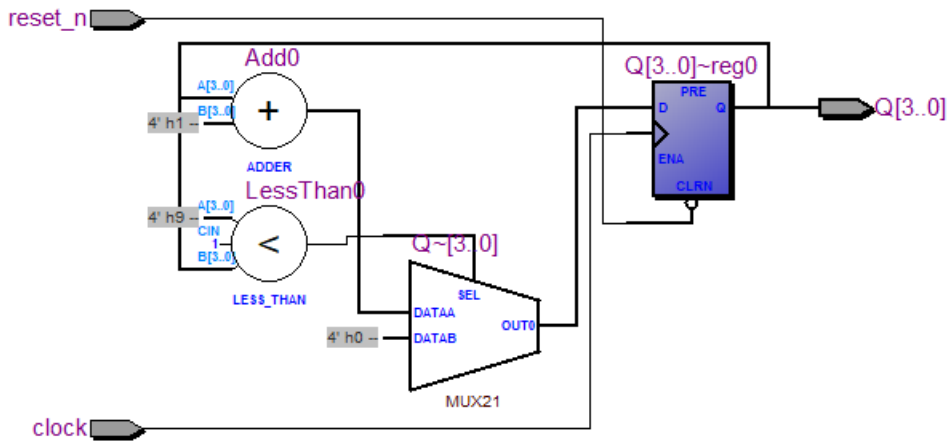
a) Code

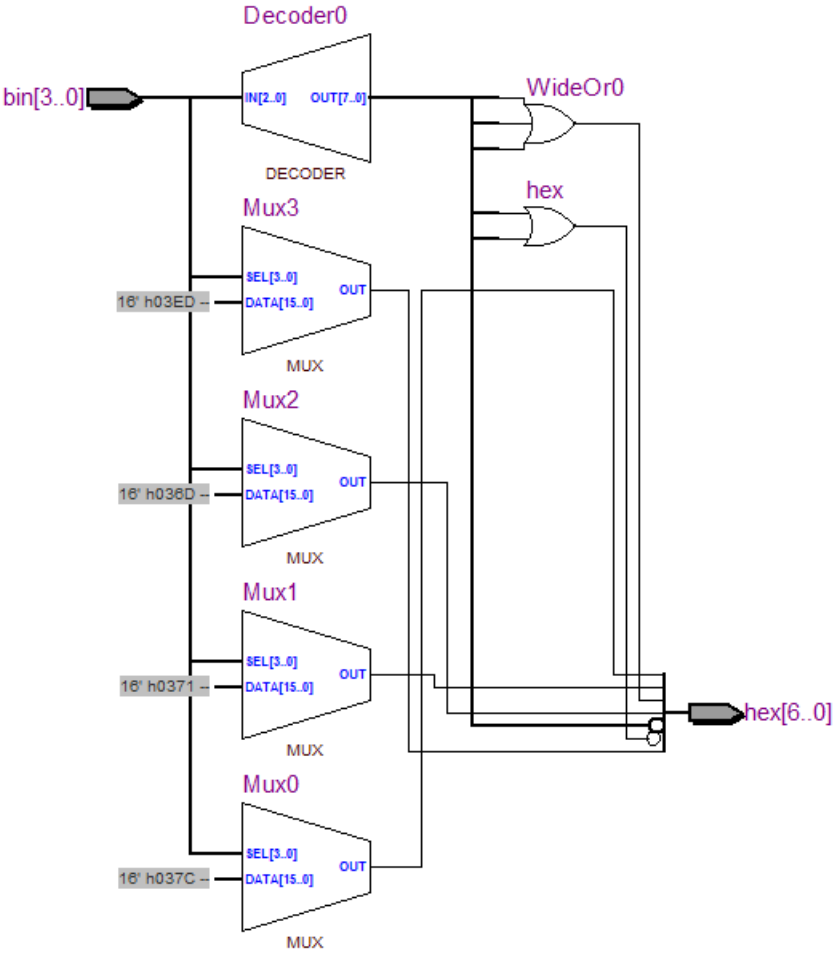
```
module Seven_Segment_Decoder (hex,
                              bin
                              );

    output [6:0] hex;

    input [3:0] bin;

    reg [6:0] hex;

    always @(*) begin

        case (bin)

            4'b0000: hex = 7'b0111111;
            4'b0001: hex = 7'b0000110;
            4'b0010: hex = 7'b1011011;
            4'b0011: hex = 7'b1001111;
            4'b0100: hex = 7'b1100110;
            4'b0101: hex = 7'b1101101;
            4'b0110: hex = 7'b1111101;
            4'b0111: hex = 7'b0000111;
            4'b1000: hex = 7'b1111111;
            4'b1001: hex = 7'b1101111;

            default: hex = 7'bxxxxxxx;

        endcase

    end

endmodule
```

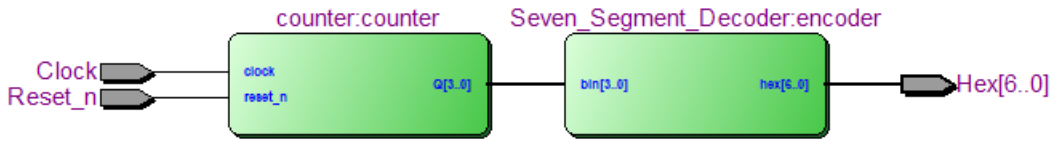
```
35  module counter (Q,
36                  clock,
37                  reset_n
38                  );
39
40      output [3:0] Q;
41
42      input clock;
43      input reset_n;
44
45      reg [3:0] Q;
46
47      always @(posedge clock or negedge reset_n) begin
48
49          if(!reset_n)
50              Q <= 0;
51
52          else if (Q >= 9)
53              Q <= 0;
54
55          else
56              Q <= Q + 1;
57
58      end
59
60  endmodule
```

```
62  module BCD_counter (Hex,
63                      Clock,
64                      Reset_n
65                      );
66
67      output [6:0] Hex;
68
69      input Clock;
70      input Reset_n;
71
72      wire [3:0] bcd;
73
74      counter counter(.Q(bcd), .clock(Clock), .reset_n(Reset_n));
75
76      Seven_Segment_Decoder encoder(.hex(Hex), .bin(bcd));
77
78  endmodule
79
```

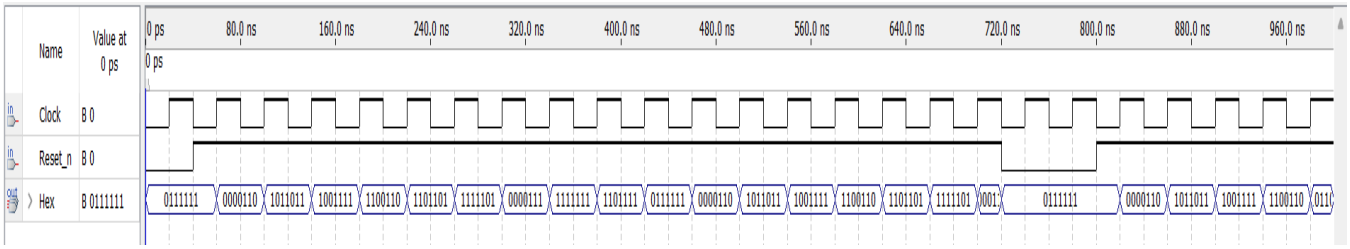
b) Netlist


reset_n
Add0
A[3..0]
B[3..0]
4' h1 --
+
ADDER
LessThan0
4' h9 --
A[3..0]
CIN
1
B[3..0]
<
LESS_THAN
Q~[3..0]
$EL
DATAA
DATAB
4' h0 --
OUT0
MUX21
Q[3..0]~reg0
PRE
D
Q
ENA
CLRN
Q[3..0]
clock


Decoder0
bin[3..0]
IN[2..0]
OUT[7..0]
DECODER
WideOr0
hex
Mux3
$EL[3..0]
DATA[15..0]
16' h03ED --
OUT
MUX
Mux2
16' h036D --
Mux1
16' h0371 --
Mux0
16' h037C --
hex[6..0]



c) Simulation



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

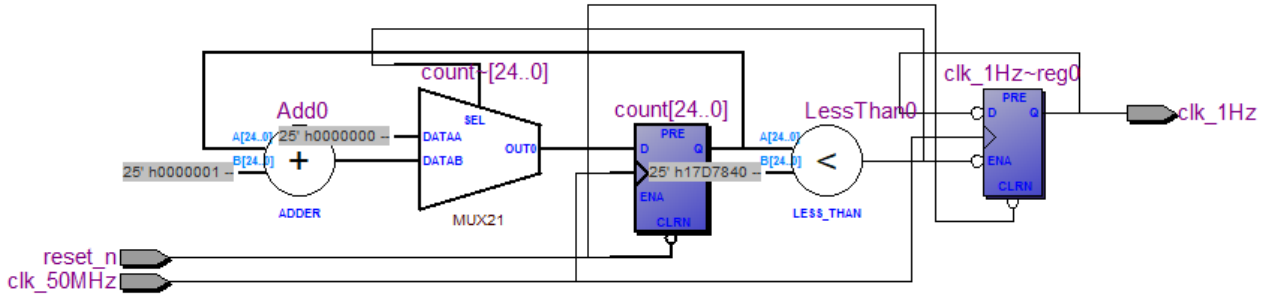
Ý tưởng thực hiện bộ chuyển đổi 50MHz sang 1Hz: Thực hiện đếm đến 25000 lần và đảo cạnh xung clock

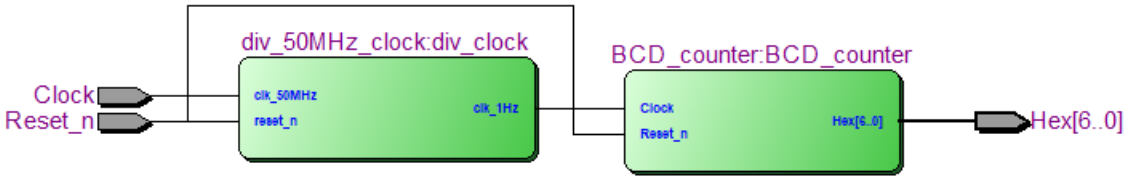
a) Code

```
1   module div_50MHz_clock (clk_1Hz,
2                           clk_50MHz,
3                           reset_n
4                           );
5
6       output clk_1Hz;
7
8       input clk_50MHz;
9       input reset_n;
10
11      reg clk_1Hz;
12      reg [24:0] count;
13
14      always @(posedge clk_50MHz or negedge reset_n) begin
15
16          if(!reset_n) begin
17              count <= 0;
18              clk_1Hz <= 0;
19          end
20
21          else if(count < 25000000) begin
22              count <= count + 1;
23              end
24
25          else begin
26              count <= 0;
27              clk_1Hz <= ~clk_1Hz;
28          end
29
30      end
31  endmodule
32
```

```
34  module BCD_counter_1s (Hex,
35                         Clock,
36                         Reset_n
37                         );
38
39      output [6:0] Hex;
40
41      input Clock;
42      input Reset_n;
43
44      wire clk_1Hz;
45
46      div_50MHz_clock div_clock(.clk_1Hz(clk_1Hz), .clk_50MHz(Clock), .reset_n(Reset_n));
47
48      BCD_counter BCD_counter(.Hex(Hex), .Clock(clk_1Hz), .Reset_n(Reset_n));
49
50  endmodule
```

b) Netlist





c) Simulation

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 (bất 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.

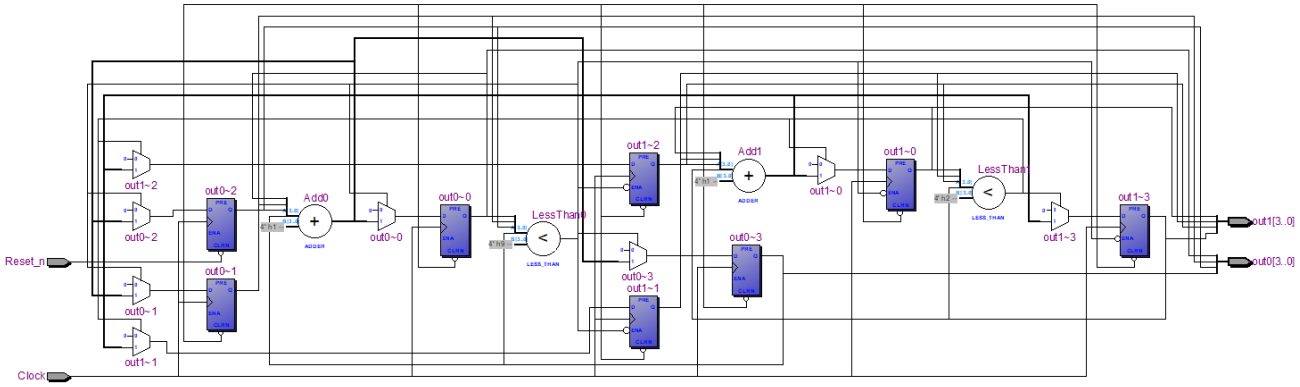
Ý tưởng của bộ counter là lưu đồng thời 2 số đơn vị và hàng chục, tăng đơn vị sau mỗi một clock, kiểm tra nếu đơn vị vượt quá 9 thì thực hiện tăng hàng chục và đưa đơn vị về lại 0.

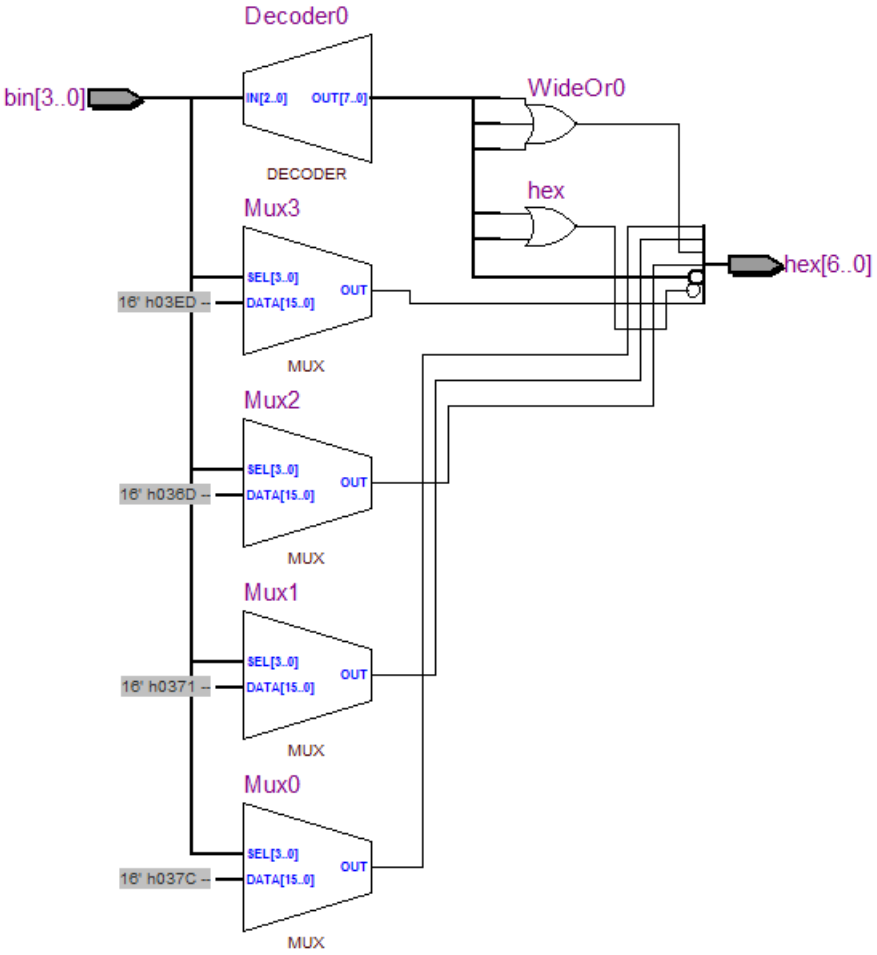
a) Code

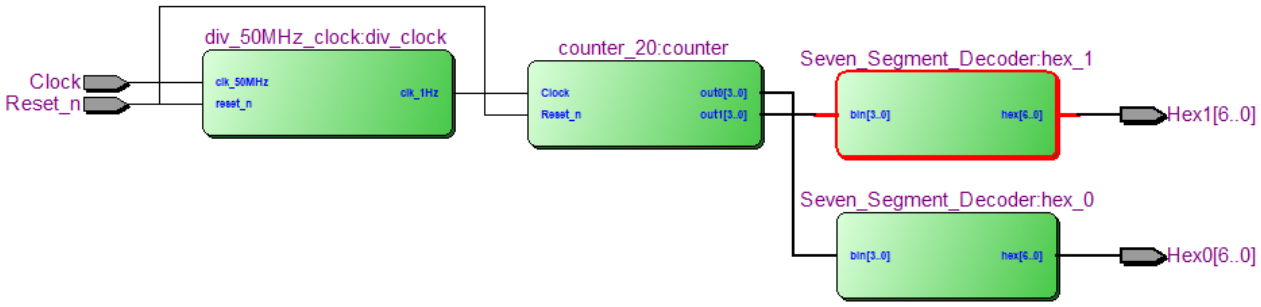
```
module counter_20 (out0,
                   out1,
                   Clock,
                   Reset_n
                   );

    output [3:0] out0;
    output [3:0] out1;

    input Clock;
    input Reset_n;

    reg [3:0] out0;
    reg [3:0] out1;

    always @(posedge Clock or negedge Reset_n) begin

       out0 <= (!Reset_n) ? 0 : (out0 < 9) ? (out0 + 1) : 0;
       out1 <= (!Reset_n) ? 0 : (out0 < 9) ? (out1) : (out1 < 2) ? (out1 + 1) : 0;

    end

endmodule
```

```
module BCD_2_digit (Hex0,
                    Hex1,
                    Clock,
                    Reset_n
                    );

   output [6:0] Hex0;
   output [6:0] Hex1;

   input Clock;
   input Reset_n;

   wire clk_1Hz;
   wire [3:0] digit0;
   wire [3:0] digit1;

   div_50MHz_clock div_clock(.clk_1Hz(clk_1Hz), .clk_50MHz(Clock), .reset_n(Reset_n));

   counter_20 counter(.out0(digit0), .out1(digit1), .Clock(clk_1Hz), .Reset_n(Reset_n));

   Seven_Segment_Decoder hex_0(.hex(Hex0), .bin(digit0));
   Seven_Segment_Decoder hex_1(.hex(Hex1), .bin(digit1));

endmodule
```

b) Netlist


Reset_n
Clock
out1~2
out0~2
out0~1
out1~1
out0~2
Add0
out0~0
out0~0
LessThan0
out1~2
out0~3
out1~1
Add1
out1~0
out0~3
out1~0
LessThan1
out1~3
out1~3
out1[3..0]
out0[3..0]


Decoder0
bin[3..0]
IN[2..0]
OUT[7..0]
DECODER
WideOr0
hex
hex[6..0]
Mux3
$EL[3..0]
DATA[15..0]
OUT
16' h03ED --
MUX
Mux2
16' h036D --
MUX
Mux1
16' h0371 --
MUX
Mux0
16' h037C --
MUX



c) Simulation

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.

Ý tưởng thực hiện là lưu 3 biến second, minute, hour. Thực hiện đếm thời gian và đưa qua bộ tách số hàng chục và hàng đơn vị, sau đó đưa vào bộ hiển thị led 7-segments. Thực hiện chuyển đổi clock 50MHz sang 1Hz như bài trên.

a) Code

```
module split_digit(out0,
                   out1,
                   num
                   );

    output [3:0] out0;
    output [3:0] out1;

    input [5:0] num;

    reg [3:0] out0;
    reg [3:0] out1;

    always @(*) begin

        out0 = (num >= 50) ? (num - 50) :
               (num >= 40) ? (num - 40) :
               (num >= 30) ? (num - 30) :
               (num >= 20) ? (num - 20) :
               (num >= 10) ? (num - 10) : num;

        out1 = (num >= 50) ? 5 :
               (num >= 40) ? 4 :
               (num >= 30) ? 3 :
               (num >= 20) ? 2 :
               (num >= 10) ? 1 : 0;

    end

endmodule
```

```
32  module checker(error,
33                 hour,
34                 minute,
35                 second
36                 );
37
38      output error;
39
40      input [5:0] hour;
41      input [5:0] minute;
42      input [5:0] second;
43
44      reg error;
45
46      always @(*) begin
47
48          error = (hour > 23) ? 1 :
49                  (minute > 59) ? 1 :
50                  (second > 59) ? 1 : 0;
51
52      end
53
54  endmodule
```

```
56  module Time_Counter(hour,
57                      minute,
58                      second,
59                      clock,
60                      reset_n,
61                      config_en,
62                      hour_config,
63                      minute_config,
64                      second_config
65                      );
66
67      output [5:0] hour;
68      output [5:0] minute;
69      output [5:0] second;
70
71      input clock;
72      input reset_n;
73      input config_en;
74      input [5:0] hour_config;
75      input [5:0] minute_config;
76      input [5:0] second_config;
77
78      reg [5:0] hour;
79      reg [5:0] minute;
80      reg [5:0] second;
```

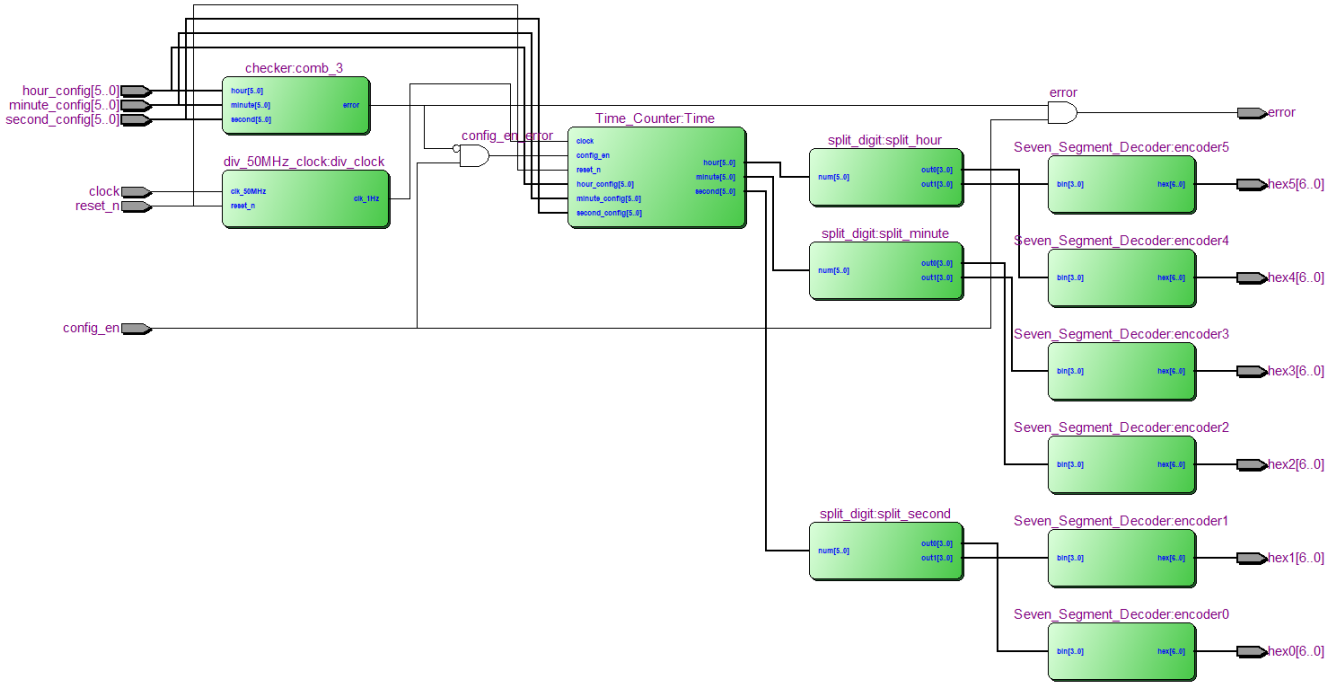
```
82    always @(posedge clock or negedge reset_n or posedge config_en) begin
83
84        second <= (!reset_n) ? 0 :              // neu khong can reset second thi sua lai ngay cho nay
85                (config_en) ? second_config :
86                (second < 59) ? (second + 1) : 0;
87
88        minute <= (!reset_n) ? 0 :
89                (config_en) ? minute_config :
90                (second < 59) ? minute :
91                (minute < 59) ? (minute + 1) : 0;
92
93        hour <= (!reset_n) ? 0 :
94                (config_en) ? hour_config :
95                (minute < 59) ? hour :
96                (second < 59) ? hour :
97                (hour < 23) ? (hour + 1) : 0;
98
99    end
100
101 endmodule
102
```

```
104 module Time_CLOCK (hex0,
105                     hex1,
106                     hex2,
107                     hex3,
108                     hex4,
109                     hex5,
110                     error,
111                     clock,
112                     reset_n,
113                     config_en,
114                     hour_config,
115                     minute_config,
116                     second_config);
117
118     output [6:0] hex5;
119     output [6:0] hex4;
120     output [6:0] hex3;
121     output [6:0] hex2;
122     output [6:0] hex1;
123     output [6:0] hex0;
124     output error;
125
126     input clock;
127     input reset_n;
128     input config_en;
129     input [5:0] hour_config;
130     input [5:0] minute_config;
131     input [5:0] second_config;
132
133     reg error;
```

```
135     wire clk_1Hz;
136     wire [5:0] hour;
137     wire [5:0] minute;
138     wire [5:0] second;
139     wire error_checker;
140     wire [3:0] hex_digit5;
141     wire [3:0] hex_digit4;
142     wire [3:0] hex_digit3;
143     wire [3:0] hex_digit2;
144     wire [3:0] hex_digit1;
145     wire [3:0] hex_digit0;
146
147     reg config_en_error;
148
149
150     div_50MHz_clock div_clock(.clk_1Hz(clk_1Hz), .clk_50MHz(clock), .reset_n(reset_n));
151
152     checker(.error(error_checker), .hour(hour_config), .minute(minute_config), .second(second_config));
153
154     always @(*) begin
155
156        config_en_error = (~error_checker) & config_en;
157        error = error_checker & config_en;
158
159     end
```

```
161     Time_Counter Time(.hour(hour),
162                       .minute(minute),
163                       .second(second),
164                       .clock(clk_1Hz),
165                       .reset_n(reset_n),
166                       .config_en(config_en_error),
167                       .hour_config(hour_config),
168                       .minute_config(minute_config),
169                       .second_config(second_config));
170
171     split_digit split_hour(.out0(hex_digit4), .out1(hex_digit5), .num(hour));
172     split_digit split_minute(.out0(hex_digit2), .out1(hex_digit3), .num(minute));
173     split_digit split_second(.out0(hex_digit0), .out1(hex_digit1), .num(second));
174
175     Seven_Segment_Decoder encoder0(.hex(hex0), .bin(hex_digit0));
176     Seven_Segment_Decoder encoder1(.hex(hex1), .bin(hex_digit1));
177     Seven_Segment_Decoder encoder2(.hex(hex2), .bin(hex_digit2));
178     Seven_Segment_Decoder encoder3(.hex(hex3), .bin(hex_digit3));
179     Seven_Segment_Decoder encoder4(.hex(hex4), .bin(hex_digit4));
180     Seven_Segment_Decoder encoder5(.hex(hex5), .bin(hex_digit5));
181
182 endmodule
183
```

b) Netlist



c) Simulation