2.LAB2
2.2.Phần 2

Các bạn đang thiết kế một mạch chuyển đổi một số nhị phân bốn bit V = v3v2v1v0 thành một số thập phân hai chữ số tương đương D = d1d0. Bảng 1 cho thấy các giá trị đầu ra cần thiết. Thiết kế một phần của mạch này được đưa ra trong hình 1. Nó bao gồm một bộ so sánh để kiểm tra xem giá trị của V có lớn hơn 9 không, và sử dụng đầu ra của bộ so sánh này để điều khiển LED 7 thanh. Các bạn sắp hoàn thành thiết kế của mạch này bằng cách tạo ra một mô–đun Verilog bao gồm bộ so sánh, bộ ghép kênh, và mạch A (không bao gồm mạch B hoặc bộ giải mã LED 7 thanh tại thời điểm này). Module Verilog của các bạn sẽ có đầu vào 4–bit V, đầu ra 4–bit M và đầu ra z. Mục đích của bài tập này là sử dụng các biểu thức gán Verilog đơn giản xác định các chức năng logic yêu cầu của mạch bằng cách sử dụng các biểu thức Boolean. Mã Verilog của các bạn sẽ không có bất kỳ biểu thức if – else, case, hoặc các biểu thức tương tự nào cả.

Binary value	Decimal digits	
0000	0	0
0001	0	1
0010	0	2
...	...	...
1001	0	9
1010	1	0
1011	1	1
1100	1	2
1101	1	3
1110	1	4
1111	1	5

Thực hiện các bước sau đây:

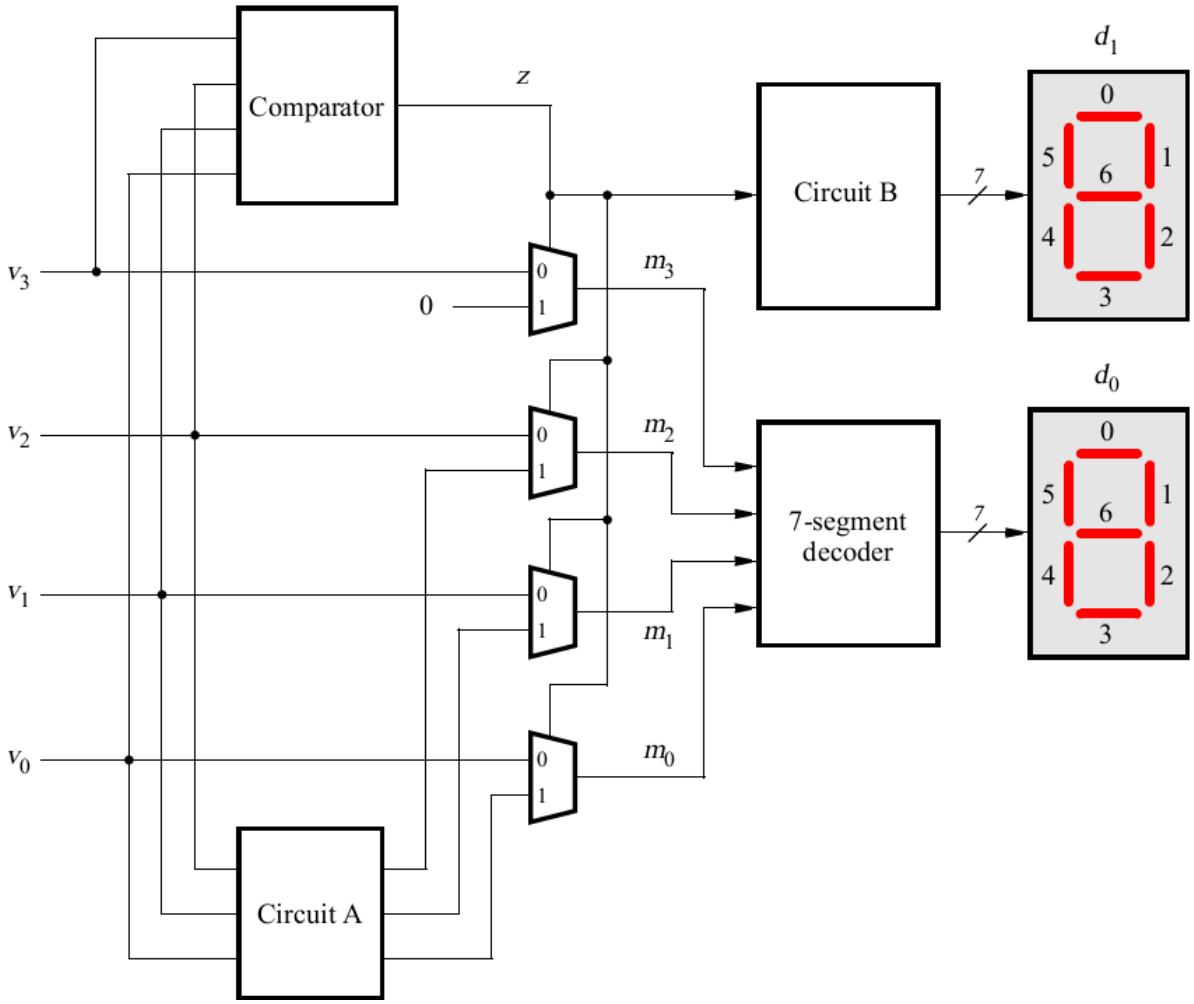
1. Tạo một dự án Quartus II mới cho module Verilog của các bạn.

2. Biên dịch mạch và sử dụng chức năng mô phỏng nhằm xác minh các hoạt động chính xác của bộ so sánh, bộ ghép kênh, và mạch A của các bạn.

3. Mở rộng mã Verilog của các bạn để thêm vào mạch B trong hình 1 cũng như bộ giải mã LED 7 thanh. Thay đổi các đầu vào và đầu ra trong mã Verilog của các bạn để sử dụng các công tắc SW3–0 trên Kit DE2 để đại diện cho số nhị phân V, và các LED 7 thanh HEX1 và HEX0 để hiển thị 2 chữ số thập phân d1 và d0. Hãy chắc chắn rằng trong dự án của các bạn đã có các phép gán chân cần thiết cho Kit DE2.

4. Biên dịch lại dự án, và sau đó nạp lại mạch đã biên dịch vào chip FPGA.

5. Kiểm tra mạch của các bạn bằng cách thử tất cả các giá trị có thể của V và quan sát LED 7 thanh.


Comparator
z
Circuit B
7
d_1
0
5
1
6
4
2
3
v_3
0
m_3
v_2
m_2
v_1
m_1
v_0
m_0
7-segment
decoder
7
d_0
Circuit A

Bài làm

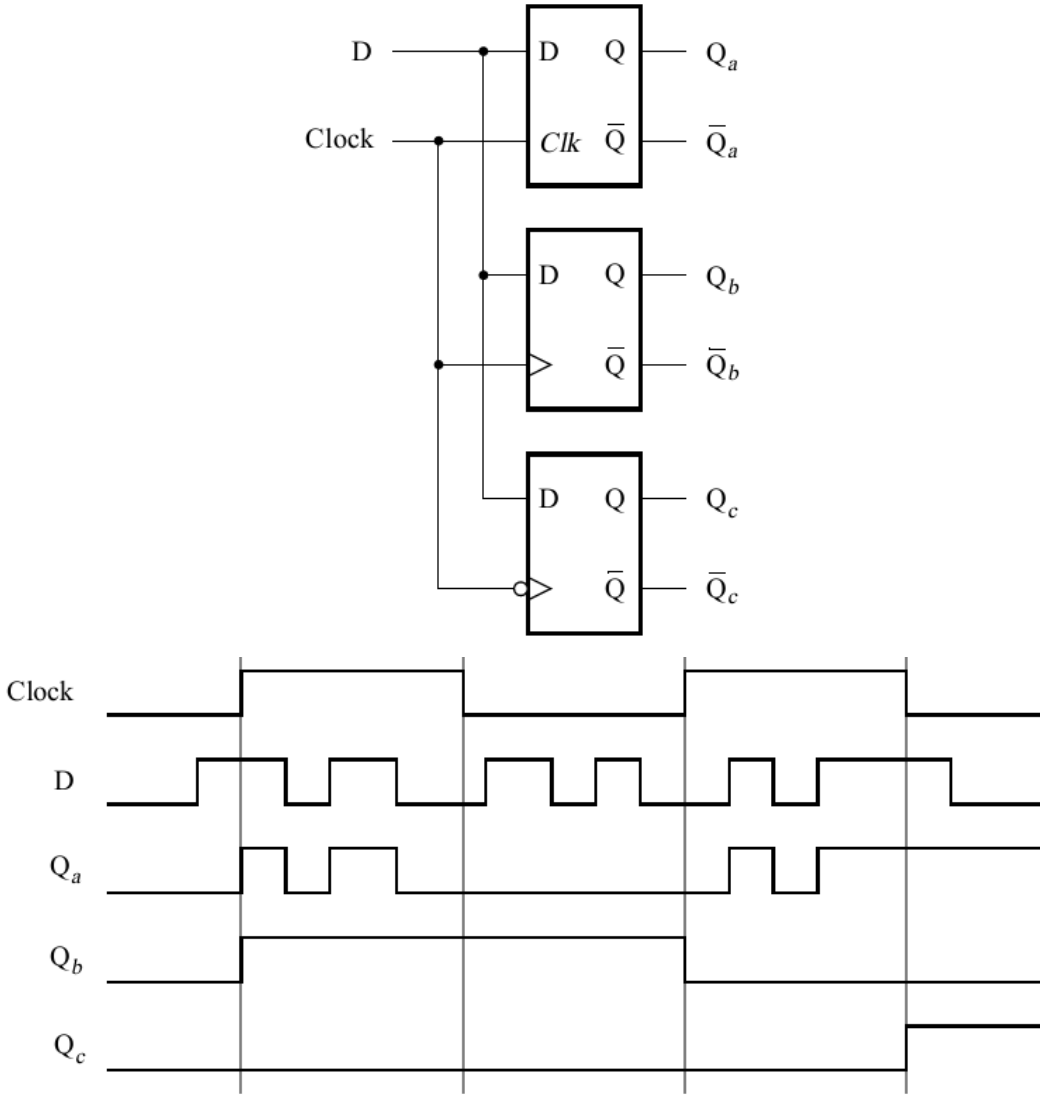
```
module comparator(V, z);
	input [3:0] V;
	output z;

	assign z = V[3] & (V[2] | V[1]);
endmodule

module circuitB(z, hex);
	input z;
	output [0:6] hex;

	assign hex = (z == 1)? 7'b1001111 : 7'b0000001;
endmodule

module circuitA(V, A);
	input [2:0] V;
	output [2:0] A;

	assign A[0] = V[0];
	assign A[1] = ~V[1];
	assign A[2] = V[2] & V[1];
endmodule

module b2d_7seg(A, hex);
	input [3:0] A;
	output [0:6] hex;

	always(A) begin
		case(A)
			0: hex = 7'b0000001;
			1: hex = 7'b1001111;
```

```
                  2: hex = 7'b0010010;
                  3: hex = 7'b0000110;
                  4: hex = 7'b1001100;
                  5: hex = 7'b0100100;
                  6: hex = 7'b0100000;
                  7: hex = 7'b0001111;
                  8: hex = 7'b0000000;
                  9: hex = 7'b0001100;
            endcase
      end
endmodule

module mux_4bit_2to1 (z, V, A, M);
      input [3:0] V, A;
      input z;
      output [3:0] M;

      assign M[3] = (~z & V[3]) | 0;
      assign M[2] = (~z & V[2]) | (z & A[2]);
      assign M[1] = (~z & V[1]) | (z & A[1]);
      assign M[0] = (~z & V[0]) | (z & A[0]);
endmodule



module L2P2(SW, HEX1, HEX0);
      input [3:0] SW;
      output [0:6] HEX1, HEX0;
      wire [3:0] A, M;
      wire z;

      assign A[3] = 0;
```

```
      comparator C1(SW[3:0], z);
      circuitA(SW[3:0], A[2:0]);
      circuitB(z, HEX1);
      mul(z, SW, A, M);
      b2d_7seg(M, HEX0);
endmodule
```

3.LAB3
3.4.Phần 4

Hình 6 biểu diễn một mạch với ba phần tử nhớ khác nhau: một chốt D, một flip–flop D sườn dương, và flip–flop D sườn âm.



Thực hiện và mô phỏng mạch này bằng cách sử dụng phần mềm Quartus II như sau:

1. Tạo một dự án mới Quartus II.

2. Viết một tập tin Verilog bao gồm ba phần tử nhớ. Đối với phần này, bạn không còn phải sử dụng chỉ thị /* synthesis keep */ từ Phần I và III. Hình 7 đưa ra mô hình hành vi của mã Verilog để định nghĩa các chốt D trong hình 4. Chốt này có thể được thực hiện theo một bảng tra cứu 4–đầu vào. Sử dụng mã theo kiểu tương tự để định nghĩa các flip–flop trong hình 6.

3. Biên dịch mã của bạn và sử dụng công cụ Technology Viewer để kiểm tra việc thực hiện mạch. Chắc chắn rằng chốt chỉ sử dụng một bảng tra cứu đầu vào và các flip–flop được thực hiện bằng cách sử dụng các flip–flop cung cấp sẵn trong chip FPGA mục tiêu.

4. Tạo một tập tin VectorWaveform (.vwf), trong đó quy định cụ thể các đầu vào và đầu ra của mạch. Vẽ các phần tử đầu vào D và Clock như được biểu diễn trong hình 6. Sử dụng mô phỏng chức năng để có được ba tín hiệu đầu ra. Quan sát các hành vi khác nhau của ba phần tử nhớ.

```
module D_latch (D, Clk, Q);

input D, Clk;

output reg Q;

always@ (D, Clk)

if (Clk)

Q = D;

endmodule
```

Hình 7. Mã Verilog kiểu hành vi định nghĩa một chốt D.

Bài làm

```
module Lab3_part4(D, Clk, Qa, Qb, Qc);
input D, Clk;
output Qa, Qb, Qc;
D_latch (D, Clk, Qa);
PD_ff(D, Clk, Qb);
ND_ff(D, ~Clk, Qc);
endmodule

// GATED D LATCH
module D_latch (D, Clk, Q);
input D, Clk;
output reg Q;
always @ (D, Clk)
if (Clk)
Q = D;
endmodule

// Positive-edge D Flip Flop
module PD_ff(D, Clk, Q);
input D, Clk;
output reg Q;
always @(posedge Clk)
if (Clk)
Q = D;
endmodule

// Negative-edge D Flip Flop
module ND_ff(D, Clk, Q);
input D, Clk;
output reg Q;
always @(negedge Clk)
```

```
if (~Clk)
Q = D;
endmodule
```

3.5.Phần 5

Chúng ta muốn hiển thị giá trị thập lục phân của một số 16–bit A trên 4 LED 7 thanh HEX7–4. Chúng ta cũng muốn hiển thị giá trị hex của một số B 16–bit trên 4 LED 7 thanh HEX3–0. Các giá trị của A và B là các đầu vào của mạch được cung cấp bởi các các công tắc SW15–0. Điều này được thực hiện bằng cách thiết lập các công tắc ứng với giá trị của A và sau đó thiết lập các công tắc còn lại ứng với giá trị của B, do đó, giá trị của A phải được nhớ trong mạch.

1. Tạo một dự án Quartus II mới sẽ được sử dụng để thực hiện các mạch mong muốn trên Kit DE2 của Altera.

2. Viết một tập tin Verilog cung cấp các chức năng cần thiết. Sử dụng KEY0 làm nút reset ở mức thấp, và sử dụng KEY1 làm đầu vào Clock.

3. Thêm tập tin Verilog vào trong dự án của bạn và biên dịch mạch.

4. Gán chân trên FPGA để kết nối với các công tắc và các LED 7 thanh, như được chỉ ra trong Hướng dẫn Sử dụng cho Kit DE2.

5. Biên dịch lại mạch và nạp vào chip FPGA.

6. Kiểm tra các chức năng của thiết kế của bạn bằng cách gạt công tắc và quan sát hiển thị ở đầu ra.

Bài làm

```
//LAB 3 Part 5
module part5 (SW, KEY, HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, HEX6, HEX7)
      input [15:0] SW;  //nhap vao so 16 bit
      input [1:0] KEY;  //KEY[1] clock, KEY[0] negedge reset
      output [6:0] HEX0, HEX1, HEX2, HEX3, HEX4, HEX5, HEX6, HEX7;
      wire [15:0]temp;

      PD_FF F1(SW, KEY[1], KEY[0], temp);

      bin2d H0(SW[3:0],HEX0);
      bin2d H1(SW[7:4],HEX1);
      bin2d H2(SW[11:8],HEX2);
      bin2d H3(SW[15:12],HEX3);

      bin2d H4(temp[3:0], HEX4);
      bin2d H5(temp[7:4], HEX5);
      bin2d H6(temp[11:8], HEX6);
      bin2d H7(temp[15:12], HEX7);



endmodule

module PD_FF(D, Clk, reset, Q)
      input [15:0] D;
      input Clk, reset;
      output reg [15:0] Q;
      always@(posedge Clk or negedge reset)
            begin
                  if(~reset)
                        Q = 0;
```

```
                  else if(Clk)
                        Q = D;
            end
endmodule

module bin2d(bin, hex)
      input [15:0] bin;
      output reg [0:6] hex;
      always@(bin)
            begin
                  case(bin)
                        0: hex = 7'b0000001;
                        2: hex = 7'b0100100;
                        1: hex = 7'b1111001;
                        3: hex = 7'b0110000;
                        4: hex = 7'b1110000;
                        5: hex = 7'b0010010;
                        6: hex = 7'b0000010;
                        7: hex = 7'b1111000;
                        8: hex = 7'b0000000;
                        9: hex = 7'b0010000;
                        10: hex = 7'b0001000; //A
                        11: hex = 7'b0000011; //b
                        12: hex = 7'b1000110; //C
                        13: hex = 7'b0100001; //d
                        14: hex = 7'b0000110; //E
                        15: hex = 7'b0001110; //F
                  endcase
            end
endmodule
```

4.LAB4
4.5.Phần 5

Thiết kế và thực hiện một mạch hiển thị từ HELLO, theo kiểu ticker tape, trên tám LED 7 thanh HEX7–0. Hãy làm cho các chữ di chuyển từ phải sang trái trong khoảng thời gian khoảng một giây. Các mẫu sẽ được hiển thị trong các khoảng thời gian đồng hồ liên tiếp được đưa ra trong Bảng 1.

Clock cycle	Displayed pattern							
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							

Bài làm

5.LAB5
5.1.Phần I

Thực hiện một bộ đếm BCD 3 chữ số. Hiển thị nội dung của các truy bộ đếm trên các LED 7 thanh, HEX2–0. Lấy ra các tín hiệu điều khiển, từ tín hiệu đồng hồ 50 MHz được cung cấp trên Kit DE2 của Altera, để tăng giá trị của bộ đếm trong khoảng thời gian một giây. Sử dụng nút bấm KEY0 để thiết lập lại bộ đếm về 0.

1. Tạo một dự án Quartus II mới sẽ được sử dụng để thực hiện mạch mong muốn trên Kit DE2.

2. Viết một tập tin Verilog mô tả cụ thể mạch mong muốn.

3. Thêm tập tin Verilog trên trong dự án của bạn và biên dịch mạch.

4. Mô phỏng mạch được thiết kế để xác minh chức năng của nó.

5. Gán chân trên FPGA để kết nối với LED 7 thanh và các nút bấm, như được chỉ ra đến trong Hướng dẫn Sử dụng Kit DE2.

6. Biên dịch lại mạch và nạp nó vào chip FPGA.

7. Xác minh rằng mạch của bạn làm việc một cách chính xác bằng cách quan sát các LED 7 thanh.

Bài làm

```
module part1(KEY,LEDR)
      input [1:0] KEY;
      output [17:0] LEDR;

      counter C1(KEY[1], KEY[0], LEDR[3:0])
      defparam C1.n = 4;
      defparam C1.k = 9;
endmodule
```

5.2.Phần II

Thiết kế và thực hiện một mạch trên Kit DE2 hoạt động như một đồng hồ thời gian trong ngày. Nó sẽ hiển thị giờ (từ 0 đến 23) trên LED 7 thanh HEX7–6, phút (từ 0 đến 60) trên HEX5–4 và giây (từ 0 đến 60) trên HEX3–2. Sử dụng các công tắc SW15–0 để thiết lập các phần giờ và phút của đồng hồ được hiển thị.

Bài làm

```
module part2(CLOCK_50, KEY, LEDR, HEX2, HEX1, HEX0)
      input CLOCK_50, KEY;
      input [17:0] LEDR;
      output [6:0] HEX2, HEX1, HEx0;
      wire Clk1;
      reg one, ten, hundred, count1, count2, count3;



      freg_divide D1(CLOCK_50, KEY[0], Clk1);



      always@(posedge Clk1 or negedge KEY[0])
            begin
                  if(Clk1)
                        begin
                              count1 <= count1 + 1'b1;  //voi moi xung clock
(1s) thi one = 1 => counter C1 chay tang hang don vi len 1, dong thoi bien
dem count1 tang len 1 don vi
                              one = 1;
                              if(count1 == 9)   // khi bien count tang len 9,
trang thai cua one sau do la 0 (mac du van co xung clk), bien dem hang chuc
(count2) tang len 1 don vi
                                    begin
                                          one <= 0;
                                          ten <= 1;
                                          count1 <= 1'b0;
                                          count2 <= count2 + 1'b1; //neu
count1 tnag den 9 thi clk sau do no tro ve 0, con count2 tang len 1 don vi,
                                    end
      //bien ten bat 1, counter C2 chay, tang hang chuc len 1


                              if( count2 == 9 )// neu count2 tang den 9 thi
hang tram bat len, counter C3 chay, tang len, ... tuong tu
                                    begin
```

```
                                        hundred <= 1;
                                        ten <= 0;
                                        count2 <= 1'b0;
                                        count3 <= count3 + 1'b1;
                                  end
                              if( count3 == 9) // khi dem den 999 thi quay
lai 000
                                  begin
                                        hundred <= 1'b0;
                                        ten <= 1'b0;
                                        one <= 1'b0;
                                        count1 <= 1'b0;
                                        count2 <= 1'b0;
                                        count3 <= 1'b0;
                                  end
                        end
                    else  // tat ca cac truong hop khong co xung clk thi one
ten hundred deu set bang 0
                        begin
                              one = 0;
                              ten = 0;
                              hundred = 0;
                        end
            end
      counter C1(one, KEY[0], LEDR[3:0]);
      defparam C1.n = 4;
      defparam C1.k = 10;

      counter C2(ten, KEY[0], LEDR[7:4]);
      defparam C2.n = 4;
      defparam C2.k = 10;
```

```
	counter C3(hundred, KEY[0], LEDR[11:8]);
	defparam C3.n = 4;
	defparam C3.k = 10;

	b2d_ssd H2 (LEDR[11:8], HEX2);
	b2d_ssd H1 (LEDR[7:4], HEX1);
	b2d_ssd H0 (LEDR[3:0], HEX0);
endmodule



//bo dem theo xung clock
module counter(clk, rst, Q)
	parameter n = 4;
	parameter k = 9;

	input clk, rst;
	output reg [n-1;0] Q;

	always@(posedge clk or negedge rst)
		begin
			if(~rst)
				Q <= 'b0;
			else
				begin
					Q <= Q + 1'b1;
					if (Q == k-1)
				end
						Q <= 'b0;
		end
endmodule
```

```
//bo dem 1s, dau ra la xung click 1hz
module freg_divide(Clk50, reset, Clk1)
      input Clk, reset;
      output Clk1;
      reg [26:0] count;

      always@(posedge Clk50 or negedge reset)
      begin
            if(~reset)
                  count <= 26'b0;
            else if (count == 24999999)
                  begin
                        Clk1 <= 1'b1;
                        count <= count + 1'b1;
                  end
            else if (count == 49999999)
                  begin
                        Clk1 <= 1'b0;
                        count <= 26'b0;
                  end
            else count <= count + 1'b1;

      end
endmodule

//hien thi bit to decimal
module b2d_ssd (X, SSD);
  input [4:0] X;
```

```
  output reg [0:6] SSD;


  always begin
    case(X)
      0:SSD=7'b0000001;
      1:SSD=7'b1001111;
      2:SSD=7'b0010010;
      3:SSD=7'b0000110;
      4:SSD=7'b1001100;
      5:SSD=7'b0100100;
      6:SSD=7'b0100000;
      7:SSD=7'b0001111;
      8:SSD=7'b0000000;
      9:SSD=7'b0001100;
    endcase
  end
endmodule
```

5.3.Phần III

Thiết kế và thực hiện trên Kit DE2 một mạch phản ứng hẹn giờ. Mạch sẽ hoạt động như sau:

1. Mạch được thiết lập lại bằng cách ấn phím bấm KEY0.

2. Sau một khoảng thời gian nhất định, đèn màu đỏ LEDR0 sẽ bật và bộ đếm BCD bốn chữ số sẽ bắt đầu đếm trong khoảng thời gian tính bằng mili giây. Khoảng thời gian trong vài giây từ khi mạch được tái lập cho đến khi LEDR0 bật được thiết lập bởi các công tắc SW7–0.

3. Một người có phản xạ đang được thử nghiệm phải nhấn KEY3 càng nhanh càng tốt để ngắt LED và đóng băng các bộ đếm trong trạng thái hiện tại của nó. Số đếm cho thấy thời gian phản ứng sẽ được hiển thị trên LED 7 thanh HEX2–0.

Bài làm

```
module L5P3(CLOCK_50, KEY, HEX7, HEX6, HEX5, HEX4, HEX3, HEX2, HEX1, HEX0,
SW);
	input CLOCK_50;
	input [1:0] KEY;
	input [15:0] SW;
	output [6:0] HEX7, HEX6, HEX5, HEX4, HEX3, HEX2, HEX1, HEX0;
	wire [27:0] PERSEC;
	wire [34:0] PERHOUR;
	wire [41:0] PERMIN;
	wire [5:0] SECOND, MINUTE;
	wire [4:0] HOUR;
	reg sec, min, hour;

	counter C0(CLOCK_50, KEY[0], PERSEC);
	defparam C0.n = 27;
	defparam C0.k = 50000000;

	counter C1(CLOCK_50, KEY[0], PERMIN);
	defparam C1.n = 34 ;
	defparam C2.k = 3000000000;

	counter C2(CLOCK_50, KEY[0], PERHOUR);
	defparam C2.n = 41 ;
	defparam C2.k = 180000000000;

	counter C3(sec, KEY[0], SECOND);
	defparam C3.n = 6;
	defparam C3.k = 60;

	counter C4(min, KEY[0], MINUTE);
	defparam C4.n = 6;
```

```
	defparam C4.k = 60;

	counter C5(hour, KEY[0], HOUR);
	defparam C5.n = 5;
	defparam C5.k = 24;

	twob2d(SECOND, HEX3, HEX2);
	twob2d(MINUTE, HEX5, HEX4);
	twob2d(HOUR, HEX7, HEX6);

	always@(posedge CLOCK_50)
		begin
			if(PERSEC == 49999999)
				sec = 1;
			else
				sec = 0;
			if(PERMIN == 2999999999)
				min = 1;
			else
				min = 0;
			if(PERHOUR == 179999999999)
				hour = 1;
			else
				hour = 0;
		end

endmodule

module counter(clk, rst, Q);
	parameter n = 4;
	parameter k = 9;
```

```
      input clk, rst;
      output reg [n-1:0] Q;
      always@(posedge clk or negedge rst)
            begin
                  if(~rst)
                        Q <= 'b0;
                  else
                        begin
                              Q <= Q + 1'b1;
                              if( Q == k - 1)
                                    Q <= 'b0;
                        end
            end
endmodule



module b2d_ssd (X, SSD);
  input [4:0] X;
  output reg [0:6] SSD;

  always begin
    case(X)
      0:SSD=7'b0000001;
      1:SSD=7'b1001111;
      2:SSD=7'b0010010;
      3:SSD=7'b0000110;
      4:SSD=7'b1001100;
      5:SSD=7'b0100100;
      6:SSD=7'b0100000;
```

```
      7:SSD=7'b0001111;
      8:SSD=7'b0000000;
      9:SSD=7'b0001100;
    endcase
  end
endmodule


module twob2d(D, hex1, hex0);
      input [6:0] D;
      output [6:0] hex0, hex1;
      reg [3:0] ONE, TEN;
      always begin
            ONE = D % 10;
            TEN = (D - 10) / 10;
      end
      b2d_ssd(ONE, hex0);
      b2d_ssd(TEN, hex1);
endmodule
```