**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KỸ THUẬT MÁY TÍNH**



**BÁO CÁO LAB03 CE213**

|  |  |
| --- | --- |
| **HỌ VÀ TÊN:** | **VÕ ĐÔNG THÀNH** |
| **MSSV:** | **21520457** |
| **LỚP:** | **CE213.O11.1** |

**GIẢNG VIÊN HƯỚNG DẪN:**

**HỒ NGỌC DIỄM**

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

**A diagram of a circuit

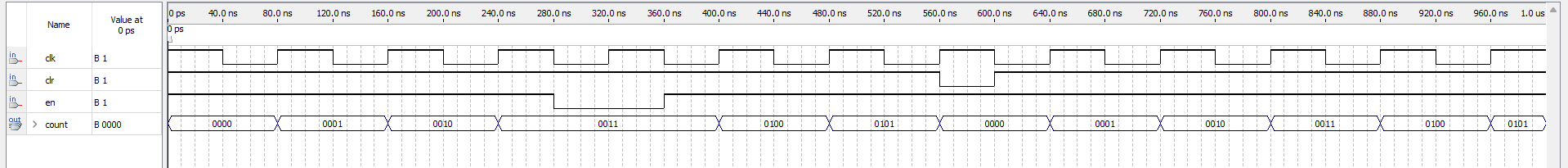
Description automatically generated**

**A white background with black text

Description automatically generated**

|  |  |
| --- | --- |
| **Code** | **RTL Viewer** |
| module four\_bit\_counter(  input clk, clr, en,  output [3:0] count  );  wire t1, t2, t3;  t\_flipflop T3(.Q(count[3]),.E(en),.C(clk),.CLR(clr));  and AND3(t3,en,count[3]);  t\_flipflop T2(.Q(count[2]),.E(t3),.C(clk),.CLR(clr));  and AND2(t2,t3,count[2]);  t\_flipflop T1(.Q(count[1]),.E(t2),.C(clk),.CLR(clr));  and AND1(t1,t2,count[1]);  t\_flipflop T0(.Q(count[0]),.E(t1),.C(clk),.CLR(clr));  endmodule |  |

**Kết quả mô phỏng:**

****

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

**Q <= Q + 1**

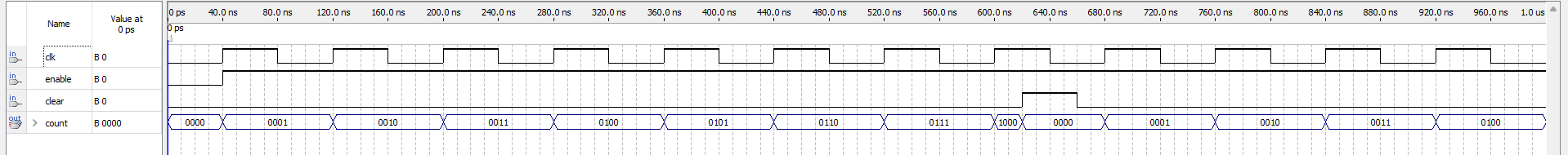
Thực hiện gán chân như câu 1.1

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** | **RTL Viewer** |
| module Counter4Bit(  input wire clk,  input wire enable,  input wire clear,  output reg [3:0] count  );  always @(posedge clk or posedge clear)  begin  if (clear)  count <= 4'b0000;  else if (enable)  count <= count + 1;  end  endmodule |  |

**Kết quả mô phỏng:**

****

**Câu 3.**

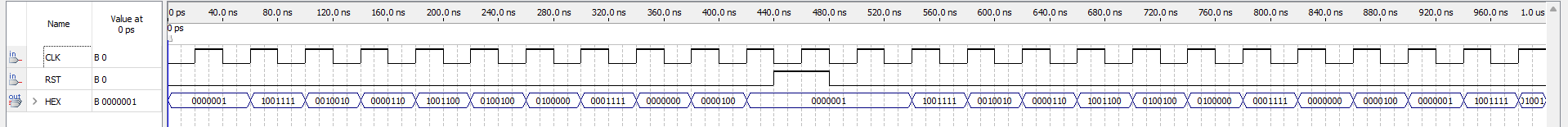
**3.1.Thiết kế mạch đếm BCD 0 đến 9 sao cho giá trị bộ đếm tăng lên 1 khi có cạnh lên của xung clock KEY[3].**

**A diagram of a computer code

Description automatically generated**

|  |  |  |
| --- | --- | --- |
| **Name** | **Code** | **RTL Viewer** |
| Counter\_BCD | module Counter\_BCD(  input wire clk, rst,  output reg [3:0] bcd  );  reg [3:0] count;  always @ (posedge clk or posedge rst)  begin  if (rst)  count <= 4'b0000;  else if (count == 4'b1001)  count <= 4'b0000;  else  count <= count + 1;  end  always @ (posedge clk or posedge rst)  begin  if (rst)  bcd <= 4'b0000;  else  begin  case (count)  4'b0000: bcd <= 4'b0000;  4'b0001: bcd <= 4'b0001;  4'b0010: bcd <= 4'b0010;  4'b0011: bcd <= 4'b0011;  4'b0100: bcd <= 4'b0100;  4'b0101: bcd <= 4'b0101;  4'b0110: bcd <= 4'b0110;  4'b0111: bcd <= 4'b0111;  4'b1000: bcd <= 4'b1000;  4'b1001: bcd <= 4'b1001;  default: bcd <= 4'b0000;  endcase  end  end  endmodule |  |
| bcd\_to\_led7 | module bcd\_to\_led7(  input [3:0] bcd,  output reg [6:0] seg  );  always @(bcd)  begin  case (bcd)  4'b0000 : seg = 7'b0000001;  4'b0001 : seg = 7'b1001111;  4'b0010 : seg = 7'b0010010;  4'b0011 : seg = 7'b0000110;  4'b0100 : seg = 7'b1001100;  4'b0101 : seg = 7'b0100100;  4'b0110 : seg = 7'b0100000;  4'b0111 : seg = 7'b0001111;  4'b1000 : seg = 7'b0000000;  4'b1001 : seg = 7'b0000100;  default : seg = 7'b1111111;  endcase  end  endmodule |  |
| Cau3\_1 | module Cau3\_1(  input CLK,RST,  output [6:0] HEX  );  wire [3:0] BCD;  Counter\_BCD Counter(.clk(CLK),.rst(RST),.bcd(BCD));  bcd\_to\_led7 Decoder(.bcd(BCD),.seg(HEX));  endmodule |  |

**Kết quả mô phỏng:**

****

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

**A diagram of a block diagram

Description automatically generated**

|  |  |
| --- | --- |
| **Code** | **RTL Viewer** |
| module Cau3\_2(  input CLOCK\_50,reset\_n,  output [7:0] HEX0  );  wire clock\_1s;  wire [3:0] bcd4;  clock\_divider Clock\_1s(.clk\_50(CLOCK\_50),.clk\_1Hz(clock\_1s));  Counter\_BCD COUNTER(.clk(clock\_1s),.rst(reset\_n),.bcd(bcd4));  bcd\_to\_led7 DECODER(.bcd(bcd4),.seg(HEX0));  endmodule |  |
| module clock\_divider (  input wire clk\_50,  output reg clk\_1Hz  );  reg [24:0] count;  parameter DIVIDER = 25000000;  always @ (posedge clk\_50)  begin  if (count == DIVIDER - 1)  begin  count <= 0;  clk\_1Hz <= ~clk\_1Hz;  end  else begin  count <= count + 1;  end  end  endmodule |  |

**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.**

|  |  |  |
| --- | --- | --- |
| **Name** | **Code** | **RTL Viewer** |
| Clock\_1hz | module clock\_1Hz (  input wire clk\_50,  output reg clk\_1Hz  );  reg [24:0] count;  parameter DIVIDER = 25000000;  always @ (posedge clk\_50)  begin  if (count == DIVIDER - 1)  begin  count <= 0;  clk\_1Hz <= ~clk\_1Hz;  end  else begin  count <= count + 1;  end  end  endmodule |  |
| counter | module counter(  input wire reset,  input wire clk1Hz,  output reg [4:0] count,  output reg [3:0] bcd\_high,  output reg [3:0] bcd\_low  );  always @(posedge clk1Hz or posedge reset)  begin  if(reset)  count <= 0;  else if (count == 20)  count <= 0;  else  count <= count + 1;  bcd\_low <= count % 10;  bcd\_high <= count / 10;  end  endmodule |  |
| decoder | module decoder(  input [3:0] bcd\_l,  input [3:0] bcd\_h,  output reg [6:0] HEX0,  output reg [6:0] HEX1  );  always @(\*) begin  case(bcd\_l)  4'b0000: HEX0 = 7'b1000000;  4'b0001: HEX0 = 7'b1111001;  4'b0010: HEX0 = 7'b0100100;  4'b0011: HEX0 = 7'b0110000;  4'b0100: HEX0 = 7'b0011001;  4'b0101: HEX0 = 7'b0010010;  4'b0110: HEX0 = 7'b0000010;  4'b0111: HEX0 = 7'b1111000;  4'b1000: HEX0 = 7'b0000000;  4'b1001: HEX0 = 7'b0010000;  default: HEX0 = 7'b1111111;  endcase  end  always @(\*) begin  case(bcd\_h)  4'b0000: HEX1 = 7'b1000000;  4'b0001: HEX1 = 7'b1111001;  4'b0010: HEX1 = 7'b0100100;  default: HEX1 = 7'b1111111;  endcase  end  endmodule |  |
| Cau5 | module Cau5(  input wire CLOCK\_50HZ, RESET,  output [6:0] hex0,hex1  );  wire clk\_1s, BCD\_H, BCD\_L ;  clock\_1Hz CLOCK (.clk\_50(CLOCK\_50HZ), .clk\_1Hz(clk\_1s));  counter COUNTER(.reset(RESET), .clk1Hz(clk\_1s), .bcd\_high(BCD\_H) ,.bcd\_low(BCD\_L));  decoder DECODER(.bcd\_h(BCD\_H), .bcd\_l(BCD\_L), .HEX0(hex0),.HEX1(hex1));  endmodule |  |

**Kết quả mô phỏng**

**A screenshot of a computer

Description automatically generated**

**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.**

**Mạch CLOCK:**

|  |
| --- |
| module Cau6(  input clk,  input rst,  input [4:0] swh,  input [5:0] swm,  input [5:0] sws,  output reg [4:0] H,  output reg [5:0] M,  output reg [5:0] S,  output reg error, count\_enable,  output reg [3:0] S\_bcd\_h, S\_bcd\_l,M\_bcd\_h,M\_bcd\_l,H\_bcd\_h,H\_bcd\_l  );  always @\* begin  if (swh != 6'b000000 || swm != 6'b000000 || sws != 6'b000000) begin  count\_enable = 1; // Enable counting if all switches are not 1  error = (swh > 23 || swm > 59 || sws > 59) ? 1 : 0;  end  else begin  count\_enable = 0; // Disable counting if any switch is 1  error = 0;  end  end  always @(posedge(clk) or posedge(rst)) begin  if (rst == 1'b1) begin  S = 0;  M = 0;  H = 0;  end  else if(clk == 1'b1) begin  if(count\_enable == 1) begin  S = sws;  M = swm;  H = swh;  S = sws + 1; //increment sec  if(S == 60) begin //check for max value of sec  S = 0; //reset seconds  M = swm + 1; //increment minutes  if(M == 60) begin //check for max value of min  M = 0; //reset minutes  H = swh + 1; //increment hours  if(H == 24) begin //check for max value of hours  H = 0; //reset hours  end  end  end  if (S > 59) S = 59;  if (M > 59) M = 59;  if (H > 23) H = 23;  end  else if(count\_enable == 0) begin  S = S + 1; //increment sec  if(S == 60) begin //check for max value of sec  S = 0; //reset seconds  M = M + 1; //increment minutes  if(M == 60) begin //check for max value of min  M = 0; //reset minutes  H = H + 1; //increment hours  if(H == 24) begin //check for max value of hours  H = 0; //reset hours  end  end  end  end  end  S\_bcd\_l <= S%10;  S\_bcd\_h <= S/10;  M\_bcd\_l <= M%10;  M\_bcd\_h <= M/10;  H\_bcd\_l <= H%10;  H\_bcd\_h <= H/10;  end  endmodule |

**Mạch Decoder**

|  |
| --- |
| module decoder(  input [3:0] s\_bcd\_l,s\_bcd\_h,m\_bcd\_l,m\_bcd\_h,h\_bcd\_l,h\_bcd\_h,  output reg [6:0] HEX0,HEX1,HEX2,HEX3,HEX4,HEX5  );  always @(\*) begin  case(s\_bcd\_l)  4'b0000: HEX0 = 7'b1000000;  4'b0001: HEX0 = 7'b1111001;  4'b0010: HEX0 = 7'b0100100;  4'b0011: HEX0 = 7'b0110000;  4'b0100: HEX0 = 7'b0011001;  4'b0101: HEX0 = 7'b0010010;  4'b0110: HEX0 = 7'b0000010;  4'b0111: HEX0 = 7'b1111000;  4'b1000: HEX0 = 7'b0000000;  4'b1001: HEX0 = 7'b0010000;  default: HEX0 = 7'b1111111;  endcase  end  always @(\*) begin  case(s\_bcd\_h)  4'b0000: HEX1 = 7'b1000000;  4'b0001: HEX1 = 7'b1111001;  4'b0010: HEX1 = 7'b0100100;  4'b0011: HEX1 = 7'b0110000;  4'b0100: HEX1 = 7'b0011001;  4'b0101: HEX1 = 7'b0010010;  4'b0110: HEX1 = 7'b0000010;  4'b0111: HEX1 = 7'b1111000;  4'b1000: HEX1 = 7'b0000000;  4'b1001: HEX1 = 7'b0010000;  default: HEX1 = 7'b1111111;  endcase  end  always @(\*) begin  case(m\_bcd\_l)  4'b0000: HEX2 = 7'b1000000;  4'b0001: HEX2 = 7'b1111001;  4'b0010: HEX2 = 7'b0100100;  4'b0011: HEX2 = 7'b0110000;  4'b0100: HEX2 = 7'b0011001;  4'b0101: HEX2 = 7'b0010010;  4'b0110: HEX2 = 7'b0000010;  4'b0111: HEX2 = 7'b1111000;  4'b1000: HEX2 = 7'b0000000;  4'b1001: HEX2 = 7'b0010000;  default: HEX2 = 7'b1111111;  endcase  end  always @(\*) begin  case(m\_bcd\_h)  4'b0000: HEX3 = 7'b1000000;  4'b0001: HEX3 = 7'b1111001;  4'b0010: HEX3 = 7'b0100100;  4'b0011: HEX3 = 7'b0110000;  4'b0100: HEX3 = 7'b0011001;  4'b0101: HEX3 = 7'b0010010;  4'b0110: HEX3 = 7'b0000010;  4'b0111: HEX3 = 7'b1111000;  4'b1000: HEX3 = 7'b0000000;  4'b1001: HEX3 = 7'b0010000;  default: HEX3 = 7'b1111111;  endcase  end  always @(\*) begin  case(h\_bcd\_l)  4'b0000: HEX4 = 7'b1000000;  4'b0001: HEX4 = 7'b1111001;  4'b0010: HEX4 = 7'b0100100;  4'b0011: HEX4 = 7'b0110000;  4'b0100: HEX4 = 7'b0011001;  4'b0101: HEX4 = 7'b0010010;  4'b0110: HEX4 = 7'b0000010;  4'b0111: HEX4 = 7'b1111000;  4'b1000: HEX4 = 7'b0000000;  4'b1001: HEX4 = 7'b0010000;  default: HEX4 = 7'b1111111;  endcase  end  always @(\*) begin  case(h\_bcd\_h)  4'b0000: HEX5 = 7'b1000000;  4'b0001: HEX5 = 7'b1111001;  4'b0010: HEX5 = 7'b0100100;  4'b0011: HEX5 = 7'b0110000;  4'b0100: HEX5 = 7'b0011001;  4'b0101: HEX5 = 7'b0010010;  4'b0110: HEX5 = 7'b0000010;  4'b0111: HEX5 = 7'b1111000;  4'b1000: HEX5 = 7'b0000000;  4'b1001: HEX5 = 7'b0010000;  default: HEX5 = 7'b1111111;  endcase  end  endmodule |

**Mạch hoàn chỉnh**

|  |
| --- |
| module Cau6\_full(  input CLK, RST,  input [4:0] SWH,  input [5:0] SWM,  input [5:0] SWS,  output ERROR, CE,  output [4:0] Hours,  output [5:0] Minutes,  output [5:0] Seconds,  output [6:0] LED0,LED1,LED2,LED3,LED4,LED5  );  wire [3:0] S\_BCD\_L,S\_BCD\_H,M\_BCD\_L,M\_BCD\_H,H\_BCD\_L,H\_BCD\_H;  Cau6 CLOCK(.clk(CLK),.rst(RST),.swh(SWH),.swm(SWM),.sws(SWS),  .error(ERROR),.count\_enable(CE),  .S\_bcd\_l(S\_BCD\_L),.S\_bcd\_h(S\_BCD\_H),  .M\_bcd\_l(M\_BCD\_L),.M\_bcd\_h(M\_BCD\_H),  .H\_bcd\_l(H\_BCD\_L),.H\_bcd\_h(H\_BCD\_H),  .H(Hours),.M(Minutes),.S(Seconds)  );  decoder LED\_DECODER(.s\_bcd\_l(S\_BCD\_L),.s\_bcd\_h(S\_BCD\_H),  .m\_bcd\_l(M\_BCD\_L),.m\_bcd\_h(M\_BCD\_H),  .h\_bcd\_l(H\_BCD\_L),.h\_bcd\_h(H\_BCD\_H),  .HEX0(LED0),.HEX1(LED1),.HEX2(LED2),  .HEX3(LED3),.HEX4(LED4),.HEX5(LED5)  );  endmodule |

**RTL Viewer**

**A computer screen shot of a computer code

Description automatically generated**