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

BÁO CÁO BÀI TẬP LỚN

Môn: Thiết kế luận lý với Verilog HDL

GVHD:

TS. Phạm Quốc Cường

Thầy Ngô Đức Minh

Thành viên:

1. Võ Nguyễn Phi Long - 1712033
2. Từ Nguyên Gia Khiêm - 1711754

MỤC LỤC

Phần 1. GIỚI THIỆU

1. **Đề tài 8:** Xây dựng bảng tính điểm cho thi đấu bóng chuyền với các yêu cầu:

- Hiển thị điểm số set đang đấu, tỉ số set hiện tại;
- Cho phép điều chỉnh điểm (tăng, giảm);
- Thông báo đội chiến thắng khi chương trình phát hiện đội chiến thắng;
- Bonus: hiển thị tên đội và báo hiệu đội chiến thắng lên màn hình LCD.

2. **Công cụ sử dụng:**

- Phần mềm Quartus Prime Lite Edition.
- Phần mềm ModelSim – Intel FPGA 10.5b (Quartus Prime 16.1).
- Board De2i – 150.

Phần 2. THIẾT KẾ

Trong luật thi đấu bóng chuyền, mỗi trận đấu sẽ có tất cả 5 hiệp đấu (set). Trong mỗi set, đội nào ghi được nhiều hơn hoặc bằng 25 điểm và cách biệt 2 điểm với đối thủ thì đội đó sẽ giành chiến thắng ở set đó. Đội nào giành chiến thắng được 3 set trước sẽ là đội thắng cuộc. Trong trường hợp sau khi trải qua 4 set đầu mà tỉ số set của hai đội là 2 – 2 thì ở set đấu cuối cùng đội nào ghi được nhiều hơn hoặc bằng 15 điểm và cách biệt 2 điểm với đối thủ sẽ trở thành đội giành chiến thắng chung cuộc.

Để xây dựng một bảng tính điểm cho luật thi đấu bóng chuyền như trên, nhóm chúng em sẽ cần thiết kế những module con sau đây:

- **BackupPoint**: có chức năng đếm lên/xuống, chế độ đến 25 điểm (set thường) và 15 điểm (set thứ 5) theo tín hiệu mode25_15. Được hỗ trợ bởi module BackupDemo***.**
    - **BackupDemo:** đếm điểm, backup điểm mỗi đội và thông báo nếu đội thắng set hiện tại.
- **BackupSet**: chức năng tương tự như module BackupPoint nhưng được áp dụng cho tính điểm set. (Do số set tối đa là 3 nên chưa nhóm nhận thấy chưa cần chức năng Backup như BackupPoint). Được hỗ trợ bởi module SetDemo***.**
    - **SetDemo:** đếm set của một đội và thông báo nếu đội đó thắng trận.
- **Led7_x2** có chức năng phân tách giá trị lớn hơn 4 bit của các số điểm thành những số có độ dài 4 bit để xuất ra led 7 đoạn.
- **decoder_led7** là module nhằm xuất các số điểm ra con led 7 đoạn.
- **LCD_controller** và **LCD_display** là hai module điều khiển việc hiển thị lên LCD 1602.
- **CounterBackup** giúp tổng hợp và kết nối tất cả các sub-module trên.
- **ASS_intf** là module gán chân các tín hiệu xuống board.

Phần 3. HIỆN THỰC

Nhóm chúng em sử dụng hai công cụ tổng hợp chính là ModelSim và Quartus để tiến hành hiện thực các module đã đề ra ở phần 2 bằng ngôn ngữ Verilog HDL. Dưới đây là những đoạn code chúng em đã viết cùng mô hình RTL sinh ra từ phần mềm Quartus:

1. **Module BackupPoint:**

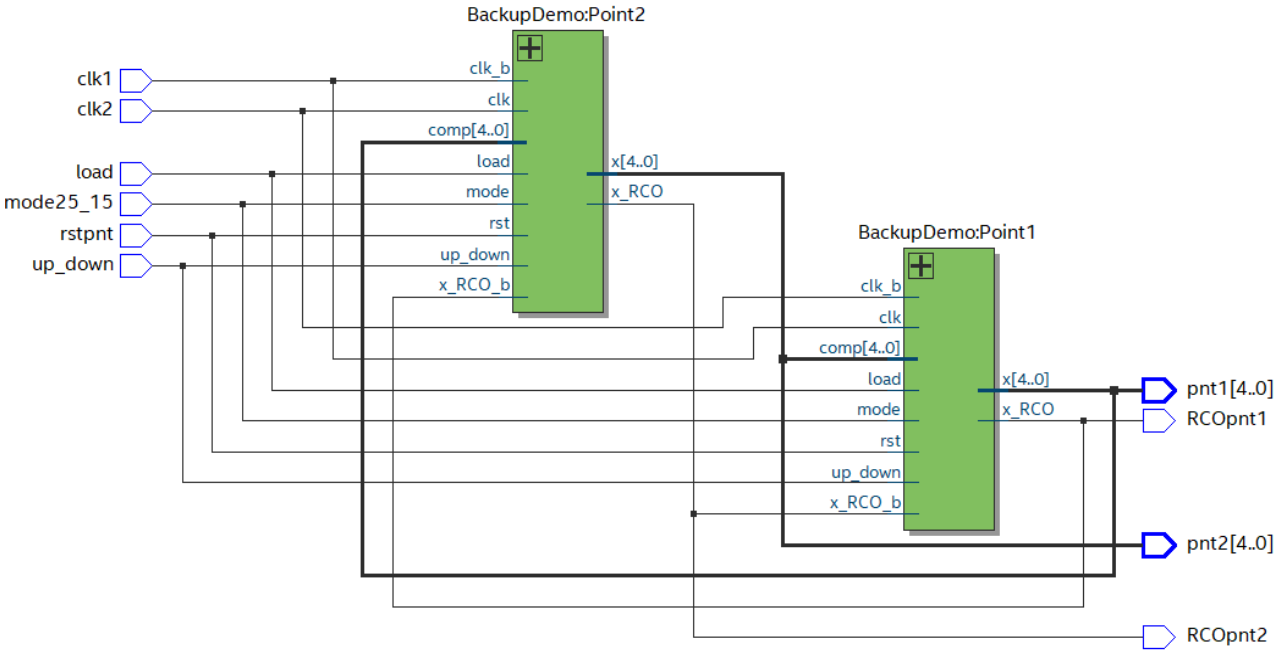
- Code Verilog:

```
module BackupDemo( x_RCO, x, x_b, clk, clk_b, x_RCO_b, load, rst, comp, up_down, mode);
input clk, clk_b, load, rst, up_down, x_RCO_b, mode;
input [4:0]comp;
output reg [4:0]x;
output reg x_RCO;
output reg [4:0]x_b;
   always @(x)
      begin
         if(mode) begin
             if((x >= 25) & (x >= comp + 2)) x_RCO = 1;
             else x_RCO = 0;
             end
         else if (!mode) begin
             if((x >= 15) & (x >= comp + 2)) x_RCO = 1;
             else x_RCO = 0;
             end
         else x_RCO = 0;
      end

   always @(posedge clk or posedge  rst)
         begin
            if (rst)  x <= 0;
            else begin
                  if (load) x <= x_b;
                  else if ((x < 31) & (up_down)) x <= x + 1;
                  else if ((x > 0) & (!up_down)) x <= x - 1;
                  else x <= x;
            end
         end
   always @(posedge rst)
      begin
         if(rst) x_b <= x;
         else x_b <= x_b;
      end
endmodule
```

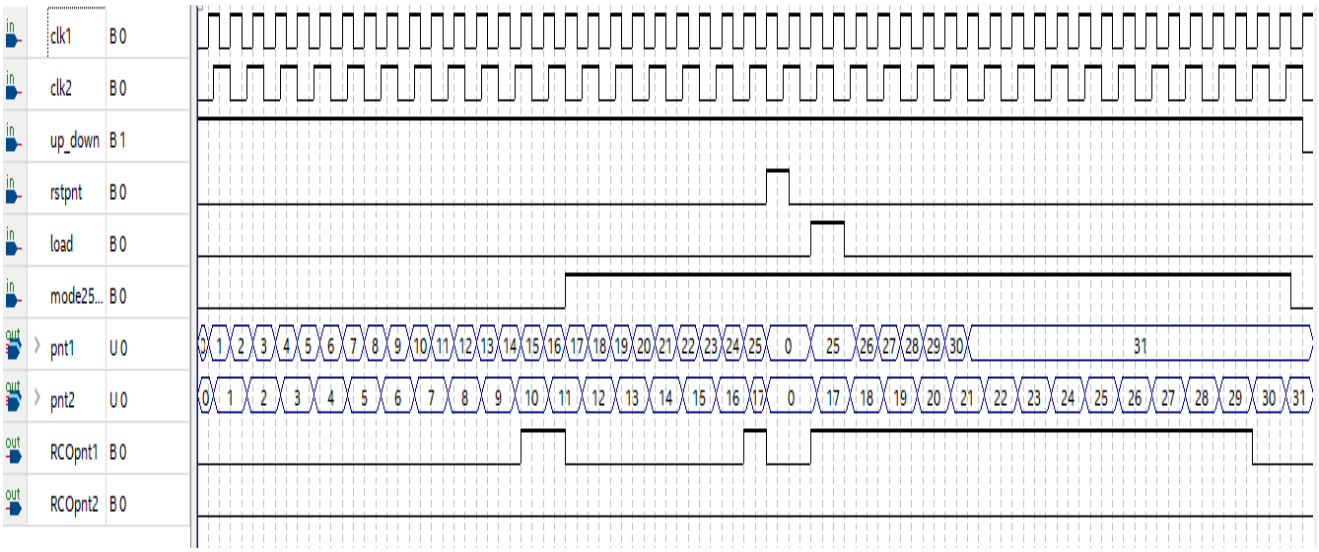
```
module BackupPoint(clk1, clk2, pnt1, pnt2,  RCOpnt1, RCOpnt2, load, rstpnt, up_down, mode25_15);
   input clk1, clk2, load, rstpnt, up_down, mode25_15;
   wire [4:0] pnt1_b, pnt2_b;
   output wire [4:0] pnt1, pnt2;
   output wire RCOpnt1, RCOpnt2;

   BackupDemo Point1( RCOpnt1, pnt1, pnt1_b, clk1, clk2, RCOpnt2, load, rstpnt, pnt2, up_down, mode25_15);
   BackupDemo Point2( RCOpnt2, pnt2, pnt2_b, clk2, clk1, RCOpnt1, load, rstpnt, pnt1, up_down, mode25_15);
endmodule

```

- Thông số:
  - BackupDemo:
    - x_RCO: thông báo nếu đội thắng set hiện tại;
    - x: điểm;
    - x_b: biến phụ kiểu reg hỗ trợ backup;
    - clk: tín hiệu xung clock của điểm đội nhà;
    - clk_b: tín hiệu xung clock của đối thủ;
    - x_RCO_b: tín hiệu RCO của đối thủ;
    - load: tín hiệu cho phép backup;
    - rst: tín hiệu reset (ưu tiên nhất);
    - comp: điểm của đối thủ;
    - up_down: tín hiệu đếm lên (1) hoặc xuống (0);
    - mode: cho phép đếm theo set 25đ (1) hoặc 15đ (0);
  - BackupPoint**:** (dễ dàng suy ra từ BackupDemo)
- Hoạt động:
  - Điểm sẽ được đếm theo chế độ ưu tiên rst (reset) - load (backup) – clk(đếm);
  - Nếu đội nhà có số điểm >= 25 và hơn đối thủ từ 2 điểm trở lên, tín hiệu RCO của đội nhà sẽ lên 1;
  - Khi có tín hiệu rst (posedge) và RCO, tín hiệu x_b sẽ ghi lại giá trị của x;
  - Khi có tín hiệu xung clk (posedge) và load, x sẽ được nạp lại giá trị cũ do x_b đang lưu trữ.

- Mô hình RTL:



- Kết quả mô phỏng



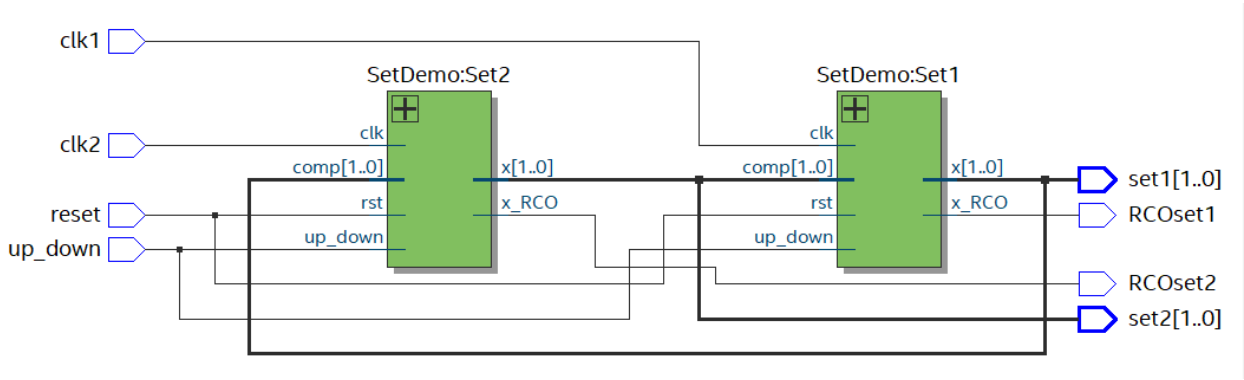
2. Module BackupSet:

- Thông số: Tương tự như BackupPoint, chỉ khác ở các điểm:
  - Không có backup;
  - Điểm tối đa là 3;
  - Khi điểm đội nhà là 3 và lớn hơn đối thủ thì RCO sẽ lên 1;
- Code Verilog:

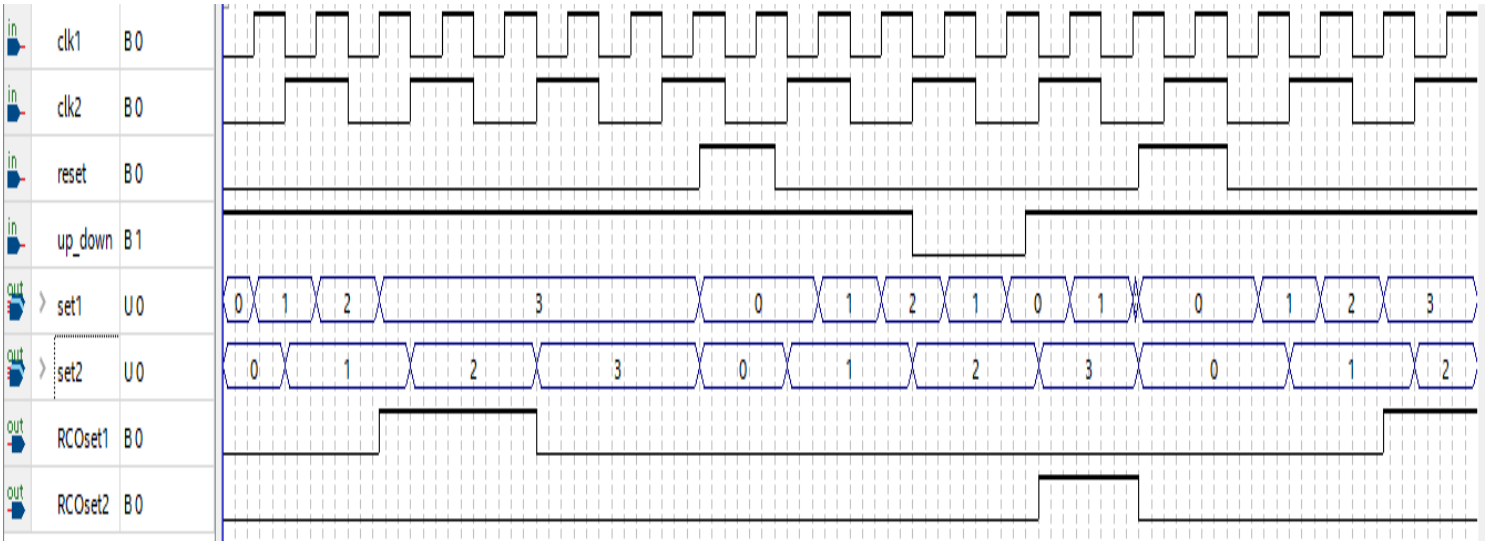
```
module SetDemo( x_RCO, x, clk, rst, comp, up_down);
input clk, rst, up_down;
input [1:0]comp;
output reg [1:0]x;
output reg x_RCO;
   always @(x)
      begin
          if((x == 3) & (x > comp)) x_RCO = 1;
          else x_RCO = 0;
      end

   always @(posedge clk or posedge  rst)
         begin
            if (rst) x <= 0;
            else if (clk)
            begin
               if ((x < 3) & (up_down))  x <= x + 1;
               else if ((x > 0) & (!up_down)) x <= x - 1;
               else x <= x;
            end
            else x <= x;
         end
endmodule
```

```
module BackupSet(clk1, clk2, set1, set2,  RCOset1, RCOset2, reset, up_down);
   input wire clk1, clk2, reset, up_down;
   output wire [1:0]set1, set2;
   output wire RCOset1, RCOset2;
   SetDemo Set1( RCOset1, set1, clk1, reset, set2, up_down);
   SetDemo Set2( RCOset2, set2, clk2, reset, set1, up_down);
endmodule
```

- Mô hình RTL:



- Kết quả mô phỏng:



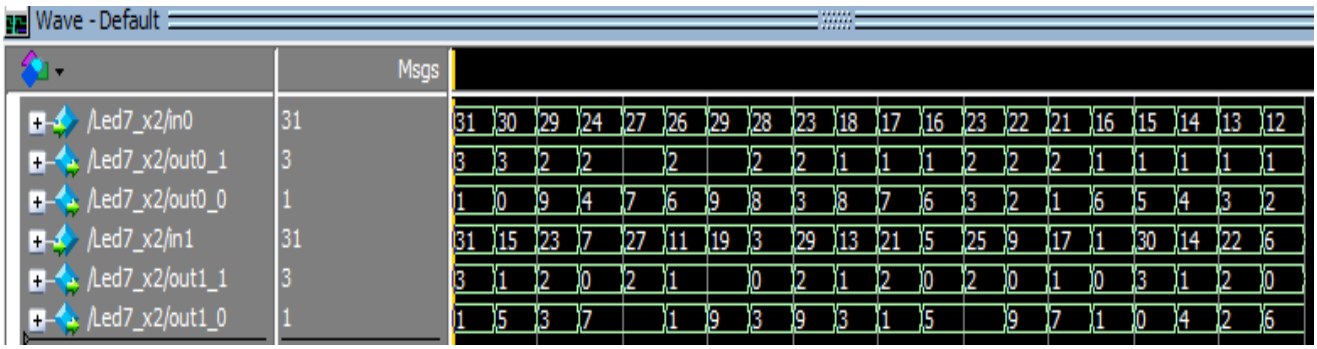
3. Module Led7_x2:

- Thông số:
  - in0: điểm thi đấu (32 > n0 >= 0);
  - out0_1: chữ số hàng chục của in0;
  - out0_0: chữ số hàng đơn vị của in0;
  - Dùng phép chia lấy dư (%) để tính chữ số hàng đơn vị (out0_0) và phép toán (in0 – out0_0)/10;
  - in1, out1_0, out1_1 tương tự.

- Code Verilog:

```
1   module Led7_x2(out0_0, out0_1, out1_0, out1_1, in0, in1);
2      input[4:0] in0, in1;
3      output [3:0] out0_0, out0_1, out1_0, out1_1;
4
5      assign out0_0 =in0 %10;
6
7      assign out1_0 =in1 %10;
8
9      assign out0_1 = (in0 - out0_0)/10;
10
11     assign out1_1 = (in1 - out1_0)/10;
12
13  endmodule
```

- Kết quả mô phỏng:



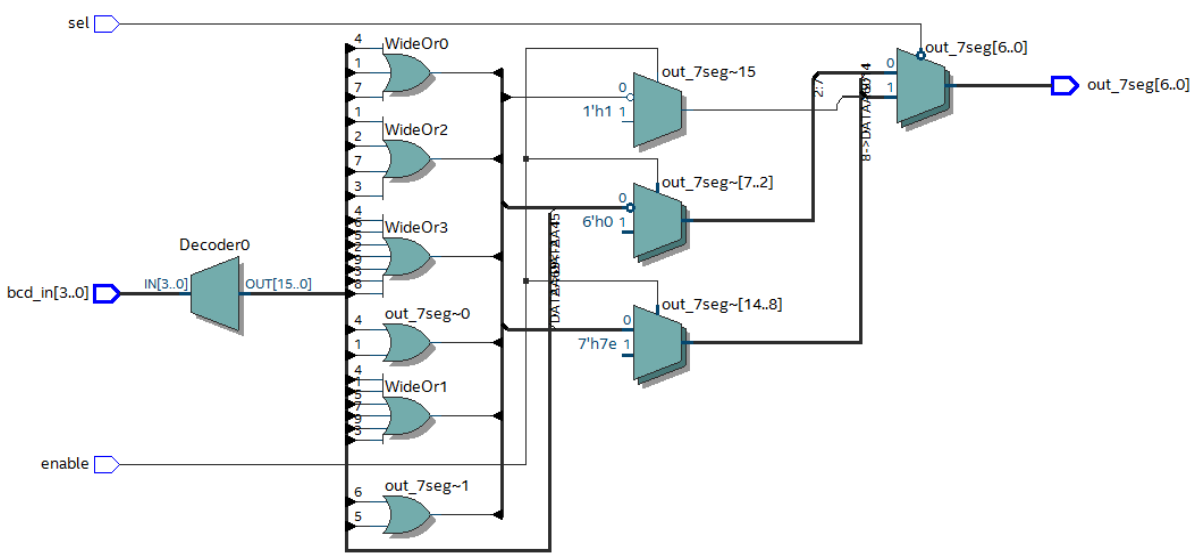
4. Module decoder_led7:

- Thông số:
  - Mode=1 (Cathode chung), khi ngõ ra bằng 1 thì led sáng, ngõ ra bằng 0 thì led tắt. Mode=0 (Anode chung) ngược lại. Dùng cấu trúc "CASE" để hiện thị led theo bcd_in.
  - Khi Enable = 1, điều chỉnh led tắt theo tín hiệu Mode.
  - Led mặc định hiển thị số "0";

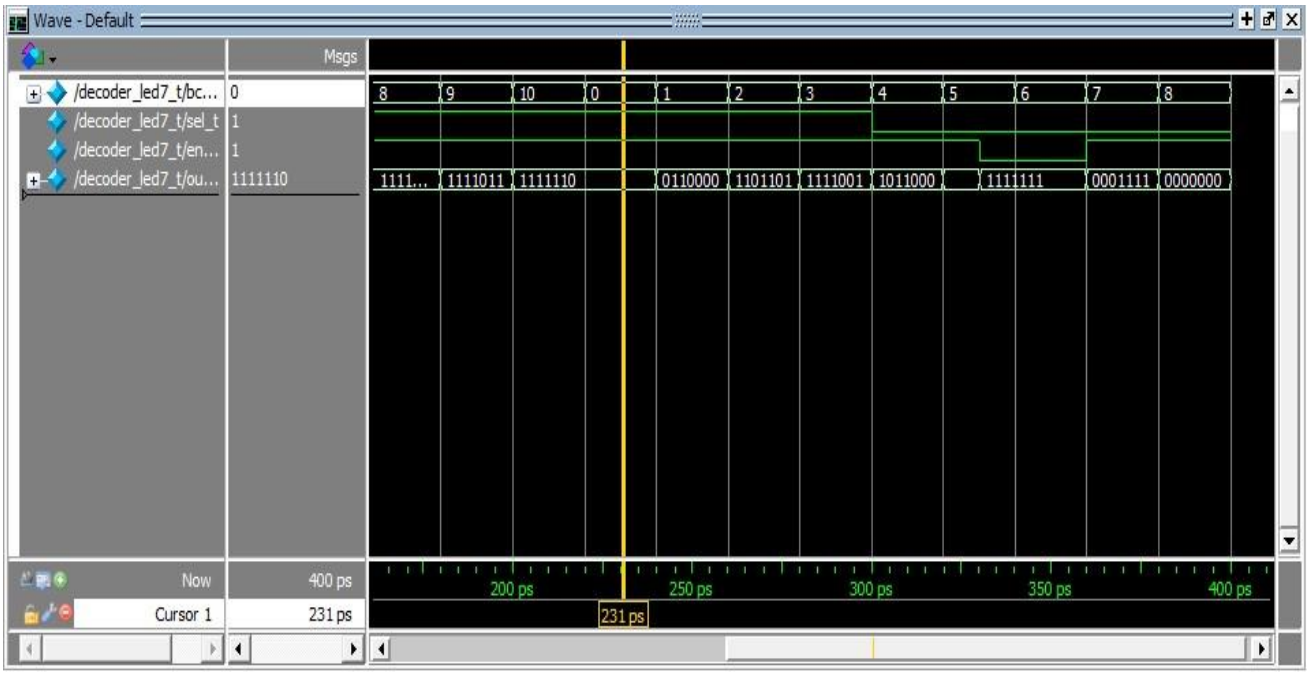
- Code Verilog:

```
module decoder_led7(enable, sel, bcd_in, out_7seg);
   input enable, sel;
   input [3:0]bcd_in;
   output reg [6:0]out_7seg;

   always @(sel) begin
      if(sel) begin
         if(enable) out_7seg = 7'b0000000;
         else case(bcd_in)
               4'b0000: out_7seg = 7'b1111110;
               4'b0001: out_7seg = 7'b0110000;
               4'b0010: out_7seg = 7'b1101101;
               4'b0011: out_7seg = 7'b1111001;
               4'b0100: out_7seg = 7'b0110011;
               4'b0101: out_7seg = 7'b1011011;
               4'b0110: out_7seg = 7'b1011111;
               4'b0111: out_7seg = 7'b1110000;
               4'b1000: out_7seg = 7'b1111111;
               4'b1001: out_7seg = 7'b1111011;
               default: out_7seg = 7'b1111110;
         endcase
      end
      if(~sel) begin //Common Anode
         if(enable) out_7seg = 7'b1111111;
         else case(bcd_in)
               4'b0000: out_7seg = 7'b0000001;
               4'b0001: out_7seg = 7'b1001111;
               4'b0010: out_7seg = 7'b0010010;
               4'b0011: out_7seg = 7'b0000110;
               4'b0100: out_7seg = 7'b1001100;
               4'b0101: out_7seg = 7'b0100100;
               4'b0110: out_7seg = 7'b0100000;
               4'b0111: out_7seg = 7'b0001111;
               4'b1000: out_7seg = 7'b0000000;
               4'b1001: out_7seg = 7'b0000100;
               default: out_7seg = 7'b0000001;
         endcase
      end
   end

endmodule
```

- Mô hình RTL:



- Kết quả mô phỏng:



5. Module LCD_controller:

- Code Verilog:

```
module LCD_Controller ( // Host Side
                        iDATA,iRS,
                        iStart,oDone,
                        iCLK,iRST_N,
                        // LCD Interface
                        LCD_DATA,
                        LCD_RW,
                        LCD_EN,
                        LCD_RS   );
// CLK
parameter   CLK_Divide  =  16;

// Host Side
input [7:0] iDATA;
input iRS,iStart;
input iCLK,iRST_N;
output   reg      oDone;
// LCD Interface
output   [7:0] LCD_DATA;
output   reg   LCD_EN;
output         LCD_RW;
output         LCD_RS;
// Internal Register
reg      [4:0] Cont;
reg      [1:0] ST;
reg      preStart,mStart;

////////////////////////////////////////////
// Only write to LCD, bypass iRS to LCD_RS
assign   LCD_DATA =  iDATA;
assign   LCD_RW      =  1'b0;
assign   LCD_RS      =  iRS;
////////////////////////////////////////////

always@(posedge iCLK or negedge iRST_N)
begin
   if(!iRST_N)
   begin
      oDone <= 1'b0;
      LCD_EN   <= 1'b0;
      preStart<=  1'b0;
      mStart   <= 1'b0;
      Cont  <= 0;
      ST    <= 0;
   end
   else
   begin
      //////   Input Start Detect ///////
      preStart<=  iStart;
      if({preStart,iStart}==2'b01)
      begin
         mStart   <= 1'b1;
         oDone <= 1'b0;
      end
      ////////////////////////////////
      if(mStart)
      begin
         case(ST)
         0: ST <= 1; // Wait Setup
         1: begin
               LCD_EN   <= 1'b1;
               ST    <= 2;
            end
         2: begin
               if(Cont<CLK_Divide)
               Cont  <= Cont+1;
               else
               ST    <= 3;
            end
         3: begin
               LCD_EN   <= 1'b0;
               mStart   <= 1'b0;
               oDone <= 1'b1;
               Cont  <= 0;
               ST    <= 0;
            end
         endcase
      end
   end
end

endmodule
```

6. Module LCD_display:

- Code Verilog:

```
module LCD_display( win1, win2,
                    // Host Side
                    iCLK, iRST_N,
                    // LCD Side
                    LCD_DATA,LCD_RW,LCD_EN,LCD_RS );
// Host Side
input    iCLK, iRST_N;
input    win1, win2;
// LCD Side
output   [7:0] LCD_DATA;
output         LCD_RW,LCD_EN,LCD_RS;
// Internal Wires/Registers
reg    [5:0]    LUT_INDEX;
reg    [8:0]    LUT_DATA;
reg    [5:0]    mLCD_ST;
reg    [17:0]   mDLY;
reg             mLCD_Start;
reg    [7:0]    mLCD_DATA;
reg             mLCD_RS;
wire            mLCD_Done;

parameter   LCD_INTIAL    = 0;
parameter   LCD_LINE1     = 5;
parameter   LCD_CH_LINE   = LCD_LINE1+16;
parameter   LCD_LINE2     = LCD_LINE1+16+1;
parameter   LUT_SIZE      = LCD_LINE1+32+1;

always@(posedge iCLK or negedge iRST_N)
begin
   if(!iRST_N)
   begin
      LUT_INDEX   <= 0;
      mLCD_ST     <= 0;
      mDLY        <= 0;
      mLCD_Start  <= 0;
      mLCD_DATA   <= 0;
      mLCD_RS     <= 0;
   end
   else
   begin
      if(LUT_INDEX<LUT_SIZE)
      begin
         case(mLCD_ST)
         0: begin
               mLCD_DATA   <= LUT_DATA[7:0];
               mLCD_RS     <= LUT_DATA[8];
               mLCD_Start  <= 1;
```

```
48                  mLCD_ST      <= 1;
49              end
50          1: begin
51                  if(mLCD_Done)
52                  begin
53                      mLCD_Start  <= 0;
54                      mLCD_ST     <= 2;
55                  end
56              end
57          2: begin
58                  if(mDLY<18'h3FFFE)
59                  mDLY  <= mDLY+1;
60                  else
61                  begin
62                      mDLY  <= 0;
63                      mLCD_ST  <= 3;
64                  end
65              end
66          3: begin
67                  LUT_INDEX   <= LUT_INDEX+1;
68                  mLCD_ST  <= 0;
69              end
70          endcase
71      end
72  end
73  end
74
75  always
76  begin
77  if({win1, win2} == 2'b10)
78  begin
79      case(LUT_INDEX)
80      // Initial
81      LCD_INTIAL+0:   LUT_DATA <= 9'h038;
82      LCD_INTIAL+1:   LUT_DATA <= 9'h00C;
83      LCD_INTIAL+2:   LUT_DATA <= 9'h001;
84      LCD_INTIAL+3:   LUT_DATA <= 9'h006;
85      LCD_INTIAL+4:   LUT_DATA <= 9'h080;
86      // Line 1
87      LCD_LINE1+0:    LUT_DATA <= 9'h120;
88      LCD_LINE1+1:    LUT_DATA <= 9'h154;   //T
89      LCD_LINE1+2:    LUT_DATA <= 9'h165;   //e
90      LCD_LINE1+3:    LUT_DATA <= 9'h161;   //a
91      LCD_LINE1+4:    LUT_DATA <= 9'h16D;   //m
92      LCD_LINE1+5:    LUT_DATA <= 9'h131;   //1
93      LCD_LINE1+6:    LUT_DATA <= 9'h120;   //
94      LCD_LINE1+7:    LUT_DATA <= 9'h1B0;   //-
```

```
 95        LCD_LINE1+8:    LUT_DATA <= 9'h120;   //
 96        LCD_LINE1+9:    LUT_DATA <= 9'h154;   //T
 97        LCD_LINE1+10:   LUT_DATA <= 9'h165;   //e
 98        LCD_LINE1+11:   LUT_DATA <= 9'h161;   //a
 99        LCD_LINE1+12:   LUT_DATA <= 9'h16D;   //m
100        LCD_LINE1+13:   LUT_DATA <= 9'h132;   //2
101        LCD_LINE1+14:   LUT_DATA <= 9'h120;
102        LCD_LINE1+15:   LUT_DATA <= 9'h120;
103        // Change Line
104        LCD_CH_LINE:    LUT_DATA <= 9'h0C0;
105        // Line 2
106        LCD_LINE2+0:    LUT_DATA <= 9'h120;
107        LCD_LINE2+1:    LUT_DATA <= 9'h157;   //w
108        LCD_LINE2+2:    LUT_DATA <= 9'h169;   //i
109        LCD_LINE2+3:    LUT_DATA <= 9'h16E;   //n
110        LCD_LINE2+4:    LUT_DATA <= 9'h16E;   //n
111        LCD_LINE2+5:    LUT_DATA <= 9'h165;   //e
112        LCD_LINE2+6:    LUT_DATA <= 9'h172;   //r
113        LCD_LINE2+7:    LUT_DATA <= 9'h120;
114        LCD_LINE2+8:    LUT_DATA <= 9'h120;
115        LCD_LINE2+9:    LUT_DATA <= 9'h120;
116        LCD_LINE2+10:   LUT_DATA <= 9'h120;
117        LCD_LINE2+11:   LUT_DATA <= 9'h120;
118        LCD_LINE2+12:   LUT_DATA <= 9'h120;
119        LCD_LINE2+13:   LUT_DATA <= 9'h120;
120        LCD_LINE2+14:   LUT_DATA <= 9'h120;
121        LCD_LINE2+15:   LUT_DATA <= 9'h120;
122        default:        LUT_DATA <= 9'h120;
123      endcase
124    end
125    else if({win1, win2} == 2'b01)
126  begin
127      case(LUT_INDEX)
128      // Initial
129      LCD_INTIAL+0:   LUT_DATA <= 9'h038;
130      LCD_INTIAL+1:   LUT_DATA <= 9'h00C;
131      LCD_INTIAL+2:   LUT_DATA <= 9'h001;
132      LCD_INTIAL+3:   LUT_DATA <= 9'h006;
133      LCD_INTIAL+4:   LUT_DATA <= 9'h080;
134      // Line 1
135      LCD_LINE1+0:    LUT_DATA <= 9'h120;
136      LCD_LINE1+1:    LUT_DATA <= 9'h154;   //T
137      LCD_LINE1+2:    LUT_DATA <= 9'h165;   //e
138      LCD_LINE1+3:    LUT_DATA <= 9'h161;   //a
139      LCD_LINE1+4:    LUT_DATA <= 9'h16D;   //m
140      LCD_LINE1+5:    LUT_DATA <= 9'h131;   //1
141      LCD_LINE1+6:    LUT_DATA <= 9'h120;   //
```

```
142         LCD_LINE1+7:    LUT_DATA <= 9'h1B0;   //-
143         LCD_LINE1+8:    LUT_DATA <= 9'h120;   //
144         LCD_LINE1+9:    LUT_DATA <= 9'h154;   //T
145         LCD_LINE1+10:   LUT_DATA <= 9'h165;   //e
146         LCD_LINE1+11:   LUT_DATA <= 9'h161;   //a
147         LCD_LINE1+12:   LUT_DATA <= 9'h16D;   //m
148         LCD_LINE1+13:   LUT_DATA <= 9'h132;   //2
149         LCD_LINE1+14:   LUT_DATA <= 9'h120;
150         LCD_LINE1+15:   LUT_DATA <= 9'h120;
151         // Change Line
152         LCD_CH_LINE:    LUT_DATA <= 9'h0C0;
153         // Line 2
154         LCD_LINE2+0:    LUT_DATA <= 9'h120;
155         LCD_LINE2+1:    LUT_DATA <= 9'h120;
156         LCD_LINE2+2:    LUT_DATA <= 9'h120;
157         LCD_LINE2+3:    LUT_DATA <= 9'h120;
158         LCD_LINE2+4:    LUT_DATA <= 9'h120;
159         LCD_LINE2+5:    LUT_DATA <= 9'h120;
160         LCD_LINE2+6:    LUT_DATA <= 9'h120;
161         LCD_LINE2+7:    LUT_DATA <= 9'h120;
162         LCD_LINE2+8:    LUT_DATA <= 9'h120;
163         LCD_LINE2+9:    LUT_DATA <= 9'h157;   //W
164         LCD_LINE2+10:   LUT_DATA <= 9'h169;   //i
165         LCD_LINE2+11:   LUT_DATA <= 9'h16E;   //n
166         LCD_LINE2+12:   LUT_DATA <= 9'h16E;   //n
167         LCD_LINE2+13:   LUT_DATA <= 9'h165;   //e
168         LCD_LINE2+14:   LUT_DATA <= 9'h172;   //r
169         LCD_LINE2+15:   LUT_DATA <= 9'h120;
170         default:        LUT_DATA <= 9'h120;
171       endcase
172     end
173     else
174       case(LUT_INDEX)
175       // Initial
176       LCD_INTIAL+0:   LUT_DATA <= 9'h038;
177       LCD_INTIAL+1:   LUT_DATA <= 9'h00C;
178       LCD_INTIAL+2:   LUT_DATA <= 9'h001;
179       LCD_INTIAL+3:   LUT_DATA <= 9'h006;
180       LCD_INTIAL+4:   LUT_DATA <= 9'h080;
181       // Line 1
182       LCD_LINE1+0:    LUT_DATA <= 9'h120;
183       LCD_LINE1+1:    LUT_DATA <= 9'h154;   //T
184       LCD_LINE1+2:    LUT_DATA <= 9'h165;   //e
185       LCD_LINE1+3:    LUT_DATA <= 9'h161;   //a
186       LCD_LINE1+4:    LUT_DATA <= 9'h16D;   //m
187       LCD_LINE1+5:    LUT_DATA <= 9'h131;   //1
188       LCD_LINE1+6:    LUT_DATA <= 9'h120;   //
```

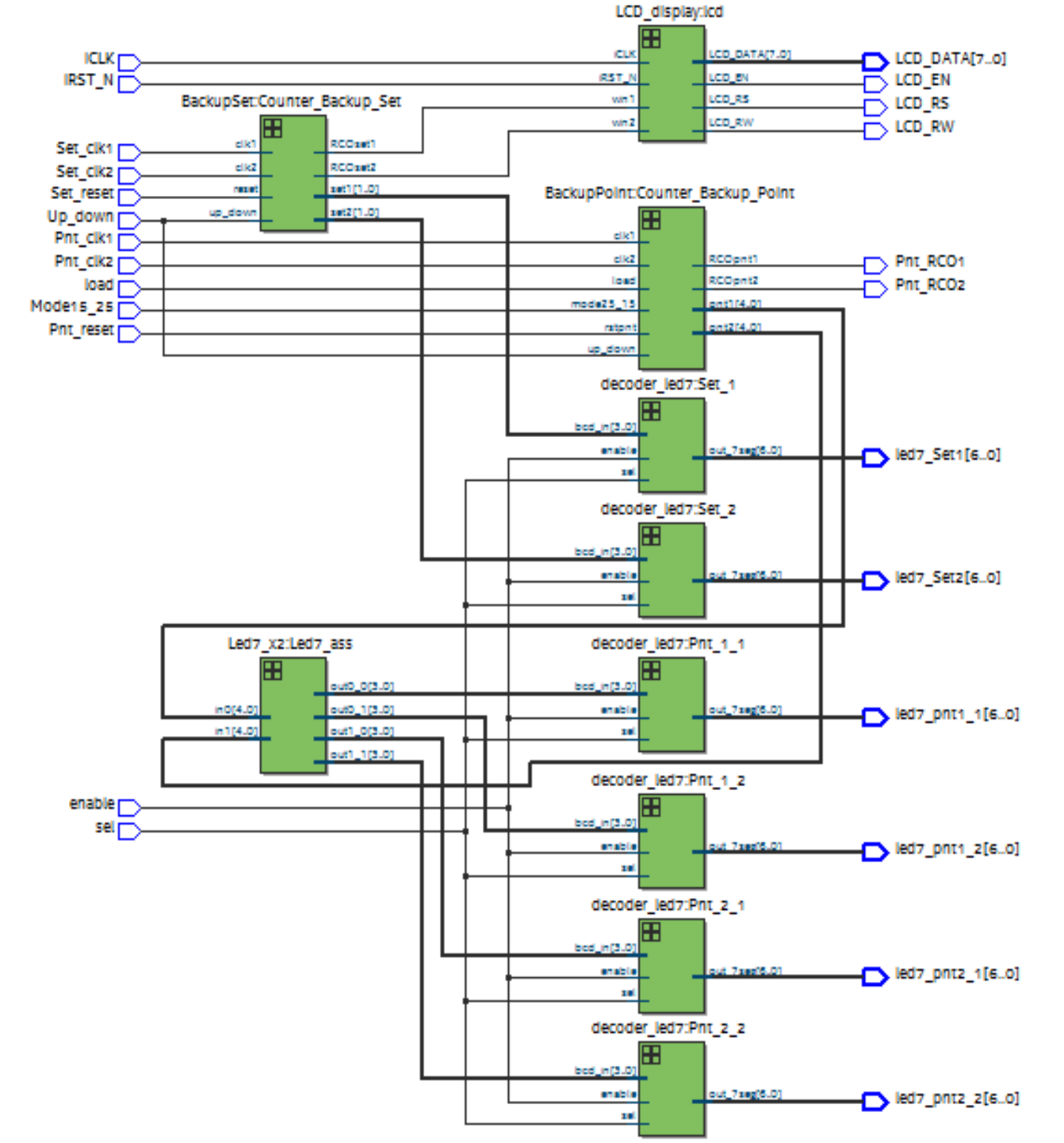
```
189         LCD_LINE1+7:    LUT_DATA <= 9'h1B0;   //-
190         LCD_LINE1+8:    LUT_DATA <= 9'h120;   //
191         LCD_LINE1+9:    LUT_DATA <= 9'h154;   //T
192         LCD_LINE1+10:   LUT_DATA <= 9'h165;   //e
193         LCD_LINE1+11:   LUT_DATA <= 9'h161;   //a
194         LCD_LINE1+12:   LUT_DATA <= 9'h16D;   //m
195         LCD_LINE1+13:   LUT_DATA <= 9'h132;   //2
196         LCD_LINE1+14:   LUT_DATA <= 9'h120;
197         LCD_LINE1+15:   LUT_DATA <= 9'h120;
198         // Change Line
199         LCD_CH_LINE:    LUT_DATA <= 9'h0C0;
200         // Line 2
201         LCD_LINE2+0:    LUT_DATA <= 9'h120;
202         LCD_LINE2+1:    LUT_DATA <= 9'h120;
203         LCD_LINE2+2:    LUT_DATA <= 9'h120;
204         LCD_LINE2+3:    LUT_DATA <= 9'h120;
205         LCD_LINE2+4:    LUT_DATA <= 9'h120;
206         LCD_LINE2+5:    LUT_DATA <= 9'h120;
207         LCD_LINE2+6:    LUT_DATA <= 9'h120;
208         LCD_LINE2+7:    LUT_DATA <= 9'h120;
209         LCD_LINE2+8:    LUT_DATA <= 9'h120;
210         LCD_LINE2+9:    LUT_DATA <= 9'h120;
211         LCD_LINE2+10:   LUT_DATA <= 9'h120;
212         LCD_LINE2+11:   LUT_DATA <= 9'h120;
213         LCD_LINE2+12:   LUT_DATA <= 9'h120;
214         LCD_LINE2+13:   LUT_DATA <= 9'h120;
215         LCD_LINE2+14:   LUT_DATA <= 9'h120;
216         LCD_LINE2+15:   LUT_DATA <= 9'h120;
217         default:        LUT_DATA <= 9'h120;
218       endcase
219   end
220
221   LCD_Controller        u0 (   // Host Side
222                         .iDATA(mLCD_DATA),
223                         .iRS(mLCD_RS),
224                         .iStart(mLCD_Start),
225                         .oDone(mLCD_Done),
226                         .iCLK(iCLK),
227                         .iRST_N(iRST_N),
228                         // LCD Interface
229                         .LCD_DATA(LCD_DATA),
230                         .LCD_RW(LCD_RW),
231                         .LCD_EN(LCD_EN),
232                         .LCD_RS(LCD_RS)    );
233
234   endmodule
```

7. Module Counter_Backup:

- Code Verilog:

```
module Counter_Backup(Pnt_clk1, Pnt_clk2, Set_clk1, Set_clk2, Pnt_RCO1, Pnt_RCO2, Set_RCO1, Set_RCO2, Pnt_reset, Set_reset,
                      led7_pnt1_1, led7_pnt1_2,led7_pnt2_1, led7_pnt2_2, led7_Set1, led7_Set2,
                      Mode15_25, enable, sel, load, Up_down,
                      iCLK, iRST_N, LCD_DATA, LCD_RW, LCD_EN, LCD_RS);
input wire Pnt_clk1, Pnt_clk2, Set_clk1, Set_clk2, Pnt_reset, Set_reset, Mode15_25, enable, sel, load, Up_down;
input wire iCLK, iRST_N;
wire [4:0] Pnt1, Pnt2;
wire [1:0] Set1, Set2;
wire [3:0] Pnt1_1, Pnt1_2, Pnt2_1, Pnt2_2;
output wire Set_RCO1, Set_RCO2;
output wire [7:0] LCD_DATA;
output wire LCD_RW, LCD_EN, LCD_RS;
output wire Pnt_RCO1, Pnt_RCO2;
output wire [6:0]led7_pnt1_1, led7_pnt1_2,led7_pnt2_1, led7_pnt2_2, led7_Set1, led7_Set2;
BackupPoint Counter_Backup_Point(Pnt_clk1, Pnt_clk2, Pnt1, Pnt2, Pnt_RCO1, Pnt_RCO2, load, Pnt_reset, Up_down, Mode15_25);
BackupSet Counter_Backup_Set(Set_clk1, Set_clk2, Set1, Set2,  Set_RCO1, Set_RCO2, Set_reset, Up_down);
Led7_x2 Led7_ass(Pnt1_1, Pnt1_2, Pnt2_1, Pnt2_2, Pnt1, Pnt2);
decoder_led7 Pnt_1_2(enable, sel, Pnt1_2, led7_pnt1_2);
decoder_led7 Pnt_1_1(enable, sel, Pnt1_1, led7_pnt1_1);
decoder_led7 Pnt_2_1(enable, sel, Pnt2_1, led7_pnt2_1);
decoder_led7 Pnt_2_2(enable, sel, Pnt2_2, led7_pnt2_2);
decoder_led7 Set_1(enable, sel, Set1, led7_Set1);
decoder_led7 Set_2(enable, sel, Set2, led7_Set2);
LCD_display lcd(Set_RCO1, Set_RCO2, iCLK, iRST_N, LCD_DATA, LCD_RW, LCD_EN, LCD_RS);
endmodule
```

- Mô hình RTL:


LCD_display:lcd
ICLK
IRST_N
LCD_DATA[7..0]
LCD_EN
LCD_RS
LCD_RW
BackupSet:Counter_Backup_Set
Set_clk1
Set_clk2
Set_reset
Up_down
BackupPoint:Counter_Backup_Point
Pnt_clk1
Pnt_clk2
load
Mode1s_2s
Pnt_reset
Pnt_RCO1
Pnt_RCO2
decoder_led7:Set_1
led7_Set1[6..0]
decoder_led7:Set_2
led7_Set2[6..0]
Led7_x2:Led7_ass
decoder_led7:Pnt_1_1
led7_pnt1_1[6..0]
decoder_led7:Pnt_1_2
led7_pnt1_2[6..0]
enable
sel
decoder_led7:Pnt_2_1
led7_pnt2_1[6..0]
decoder_led7:Pnt_2_2
led7_pnt2_2[6..0]

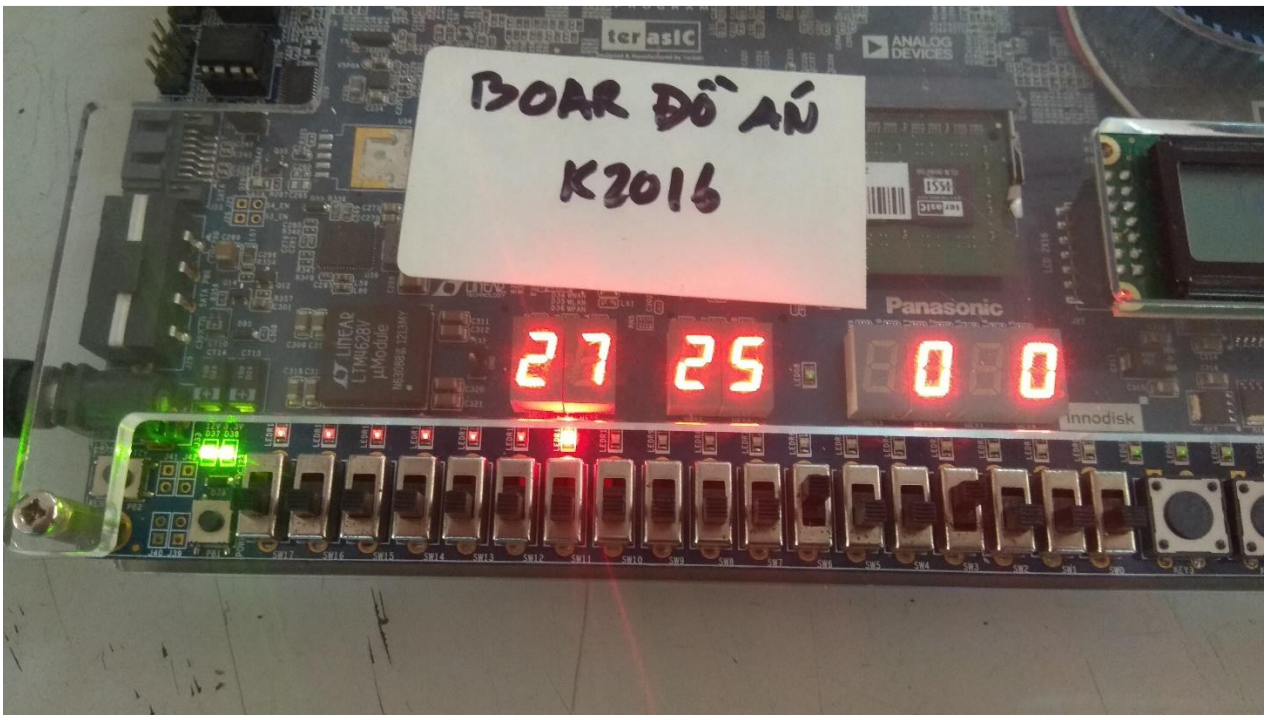
8. Module ASS_intf:

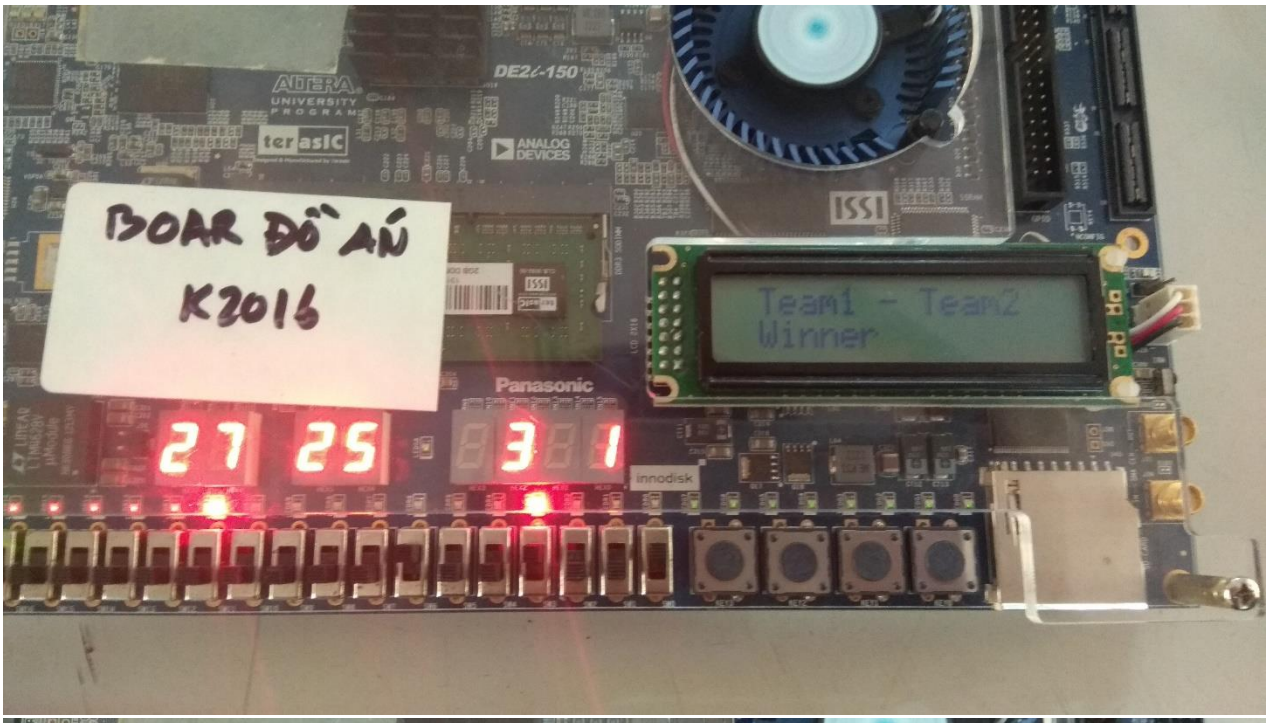
- Code Verilog:

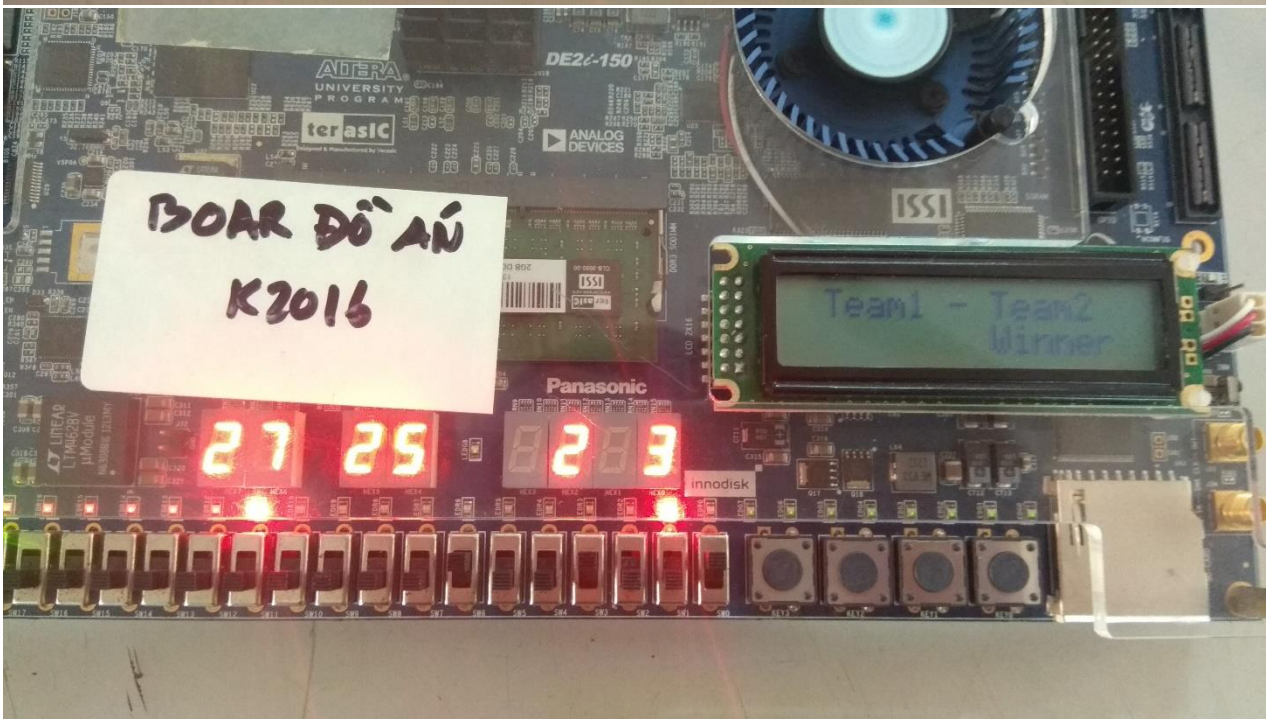
```
module ASS_intf(input [3:0]KEY, input [7:0]SW, input CLOCK_50,
                output [11:0]LEDR, output [6:0]HEX7, HEX6, HEX5, HEX4, HEX2, HEX0,
                output [7:0]LCD_DATA, output LCD_RW, LCD_EN, LCD_RS);
Counter_Backup (
  .Pnt_clk1(KEY[3]),
  .Pnt_clk2(KEY[2]),
  .Set_clk1(KEY[1]),
  .Set_clk2(KEY[0]),
  .Pnt_RCO1(LEDR[11]),
  .Pnt_RCO2(LEDR[8]),
  .Set_RCO1(LEDR[3]),
  .Set_RCO2(LEDR[1]),
  .Pnt_reset(SW[2]),
  .Set_reset(SW[1]),
  .led7_pnt1_1({HEX6[0], HEX6[1], HEX6[2], HEX6[3], HEX6[4], HEX6[5], HEX6[6]}),
  .led7_pnt1_2({HEX7[0], HEX7[1], HEX7[2], HEX7[3], HEX7[4], HEX7[5], HEX7[6]}),
  .led7_pnt2_1({HEX4[0], HEX4[1], HEX4[2], HEX4[3], HEX4[4], HEX4[5], HEX4[6]}),
  .led7_pnt2_2({HEX5[0], HEX5[1], HEX5[2], HEX5[3], HEX5[4], HEX5[5], HEX5[6]}),
  .led7_Set1({HEX2[0], HEX2[1], HEX2[2], HEX2[3], HEX2[4], HEX2[5], HEX2[6]}),
  .led7_Set2({HEX0[0], HEX0[1], HEX0[2], HEX0[3], HEX0[4], HEX0[5], HEX0[6]}),
  .Mode15_25(SW[6]),
  .enable(SW[5]),
  .sel(SW[4]),
  .load(SW[7]),
  .Up_down(SW[3]),
  .iCLK(CLOCK_50),
  .iRST_N(SW[0]),
  .LCD_DATA(LCD_DATA),
  .LCD_RW(LCD_RW),
  .LCD_EN(LCD_EN),
  .LCD_RS(LCD_RS));
endmodule
```

Phần 4: THỬ NGHIỆM

Dưới đây là một số hình ảnh được chụp trong quá trình chạy sau khi chương trình đã được nạp xuống board DE2i-150:







Link youtube demo mạch:
https://www.youtube.com/watch?v=YMtOaGZeGAU&feature=youtu.be&fbclid=IwAR2yLn7mCXLfmCWorJ7DGUJ5ADrLbLeB4S7dChm3etJVYKiywpv2E-EXokw

Phần 5: KẾT LUẬN

Nhìn chung nhóm đã thực hiện được những yêu cầu của đề tài như đếm điểm lên/xuống, hiển thị điểm, thông báo đội chiến thắng cũng như tính năng backup khi tính điểm sai và hiển thị đội chiến thắng trên LCD. Tuy nhiên cũng có một số điểm chưa được hoàn thiện: chưa thể đồng bộ đếm điểm và set, hiển thị LCD còn đơn giản, chỉ backup được sau khi 1 đội chiến thắng,.. Những hạn chế đó nhóm em hi vọng sẽ khắc phục được trong những lần cải tiến sau.

Bảng phân chia công việc:

STT	MSSV	Họ và tên	Công việc	Mức độ hoàn thành
1	1712033	Võ Nguyễn Phi Long	Đóng góp ý tưởng, viết module, mô phỏng.	100%
2	1711754	Từ Nguyên Gia Khiêm	Đóng góp ý tưởng, tổng hợp module, nạp board, viết báo cáo.	100%