CS223 Digital Design
Final Project Report

Melike Arslan
21601025
09.05.2018
Trainer Pack-43

The project consists of 7 different modules. These modules are:

- gameOfNim_call module
- scoreboard_right module
- scoreboard_left module
- SevSeg_4digit module
- debouncer module
- myGame_module module
- display_8X8 module

gameOfNim_call module: This module is the general top module that calls all other modules which means it acts as a connector for all blocks in the project.

```
module top_callModule(input clk,
        // scoreboard
        input reset, dl, dr, il, ir,
        // rgb display
        input chPlayer, rowSel1, rowSel2, rowSel3, rowSel4, newGame,
        // FPGA pins for 8x8 display
                output reset_out, //shift register's reset
                output OE,     //output enable, active low
                output SH_CP,  //pulse to the shift register
                output ST_CP,  //pulse to store shift register
                output DS,     //shift register's serial input data
                output [7:0] col_select, // active column, active high


        output logic a, b, c, d, e, f, g, dp, [3:0]an
  );
  logic [3:0] leftscoreLeft, leftscoreRight, rightscoreRight , rightscoreLeft;
  scoreboard_right boardright(clk, reset, dr, ir, rightscoreLeft, rightscoreRight);
  scoreboard_left boardleft(clk, reset, dl, il, leftscoreLeft, leftscoreRight);
  SevSeg_4digit segment(clk,
    rightscoreRight, rightscoreLeft, leftscoreRight, leftscoreLeft, //user inputs for each digit (hexadecimal value)
    a, b, c, d, e, f, g, dp, // just connect them to FPGA pins (individual LEDs).
    an // just connect them to FPGA pins (enable vector for 4 digits active low)
    );

  logic debouncerRow1, debouncerRow2, debouncerRow3, debouncerRow4, debouncerNew;
  debouncer(clk, rowSel1, debouncerRow1);
  debouncer(clk, rowSel2, debouncerRow2);
  debouncer(clk, rowSel3, debouncerRow3);
  debouncer(clk, rowSel4, debouncerRow4);
  debouncer(clk, chPlayer, debouncerChange);



  myGame_module game(clk, debouncerChange, debouncerRow1, debouncerRow2, debouncerRow3, debouncerRow4, newGame,
reset_out, OE, SH_CP, ST_CP, DS, col_select);

endmodule
```

scoreboard_right module: This module is the modifier module for the right side of the seven segment player, meaning it includes the incrementing and decrementing method for the right player. If the right player’s score reaches 99 while incrementing, next it turns to 00. If the right player’s score reaches 00 while decrementing, next it turns to 99.

```
module scoreboard_right(input clk, reset, dr, ir,
        output logic [3:0] rightscoreLeft, rightscoreRight
  );

  logic prev_dr = 0;
  logic cur_dr = 0;
  logic prev_ir = 0;
  logic cur_ir = 0;

always_ff@(posedge clk)

    begin
```

```
      prev_dr = cur_dr;
      cur_dr = dr;

      prev_ir = cur_ir;
      cur_ir = ir;

      if(reset)
      begin

        prev_dr = 0;
        cur_dr = 0;
        prev_ir = 0;
        cur_ir = 0;
        rightscoreLeft = 4'b0000;
        rightscoreRight = 4'b0000;

      end
      else
      begin
        if(cur_dr && !prev_dr)
        begin
           if(rightscoreRight == 0 && rightscoreLeft == 0)
           begin
             rightscoreRight = 9;
             rightscoreLeft = 9;
           end

           else if(rightscoreRight == 0)
           begin
             rightscoreLeft = rightscoreLeft-1;
             rightscoreRight = 9;
           end
           else
             rightscoreRight = rightscoreRight -1;
        end

        if(cur_ir && !prev_ir)
        begin
          if(rightscoreRight == 9)
          begin
            rightscoreRight = 0;
            rightscoreLeft = rightscoreLeft+1;
          end
          else
            rightscoreRight = rightscoreRight+1;

          if(rightscoreRight == 9 && rightscoreLeft == 9)
          begin
            rightscoreRight = 0;
            rightscoreLeft = 0;
          end
        end
      end
    end
endmodule
```

scoreboard_left module: The same thing happens as the right scoreboard with the difference of incrementing and decrementing the left player. This actually does the same thing so it might have been more efficient to have only module and calling it twice instead of creating another one for each side.

SevSeg_4digit module: This module was given to us. All it does is connect to the seven segment displayer on the fpga.

debouncer module: Normally when a button is pushed approximately 100 000 highs are being sent. The debouncer module is applied on the buttons we use for each row and what it does is that it reduces the 100 000 inputs into 1 input. It is similar to a clock divider.

```
module debouncer(input clk,
  input PB,
  input PB_state

  );
```

```
    logic [20:0] PB_cnt = 21'b0;

    always @(posedge clk)
        if(PB)
        begin
            if(!PB_state)
                PB_cnt <= PB_cnt + 1'b1;
        end
        else
            PB_cnt <= 21'b0;
        assign PB_state = PB_cnt[20];
endmodule
```

myGame_module module: This module is a rather longer one. So I won’t include all parts of the code, for example the inputs, the outputs and the initializations of the logics.

This part of the occurs when the new game switch is turned on:

```
        if(newGame)
        begin
            image_red = {8'b00000011, 8'b00000011, 8'b00110011, 8'b00110011, 8'b00110011, 8'b00000011, 8'b00000011, 8'b00000000};
            image_green = {8'b00000000, 8'b00000000, 8'b00000000, 8'b00000000, 8'b00000000, 8'b00000000, 8'b00000000, 8'b00000000};
            image_blue = {8'b00000000, 8'b00001100, 8'b00001100, 8'b11001100, 8'b00001100, 8'b00001100, 8'b00000000, 8'b00000000};

            cnt1 = 0;
            cnt2 = 0;
            cnt3 = 0;
            cnt4 = 0;

            prev_rowSel1 = 0;
            prev_rowSel2 = 0;
            prev_rowSel3 = 0;
            prev_rowSel4 = 0;
            prev_chPl = 0;

            curr_rowSel1 = 0;
            curr_rowSel2 = 0;
            curr_rowSel3 = 0;
            curr_rowSel4 = 0;
            curr_chPl = 0;

            row1Boolean = 1;
            row2Boolean = 1;
            row3Boolean = 1;
            row4Boolean = 1;

            leftPlayer = 4'b0000;
```

The following part occurs for each case when a button is pressed but only with fewer cases:

```
if (curr_rowSel4 && !prev_rowSel4 && row4Boolean && image_red[0][0] != 0)
            begin
                row3Boolean = 0;
                row2Boolean = 0;
                row1Boolean = 0;
                cnt4++;
                case(cnt4)
                    1:
                    begin
                      image_red[6][0] = 0;
                     image_red[6][1] = 0;
                    end
                    2:
                    begin
                        image_red[5][0] = 0;
                        image_red[5][1] = 0;
                    end
                    3:
                    begin
                        image_red[4][0]=0;
                        image_red[4][1]=0;

                    end
                    4:
                    begin
                        image_red[3][0] = 0;
                        image_red[3][1] = 0;
                    end
                    5:
                    begin
                        image_red[2][0] = 0;
                        image_red[2][1] = 0;
                    end
                    6:
                    begin
                        image_red[1][0] = 0;
                        image_red[1][1] = 0;
                    end
                    7:
                    begin
                        image_red[0][0] = 0;
                        image_red[0][1] = 0;
                        leftPlayer++;
                        row3Boolean = 1;
                        row2Boolean = 1;
                        row1Boolean = 1;

                    end
                endcase
            end
```

This part is for changing the player in each turn:

```
      else if ((curr_chPl && !prev_chPl) || (cnt1 == 1 || cnt2 == 3 || cnt3 == 5 || cnt4 == 7))
      begin

        row1Boolean = 1;
        row4Boolean = 1;
        row3Boolean = 1;
        row2Boolean = 1;
        leftPlayer++;


      end
```

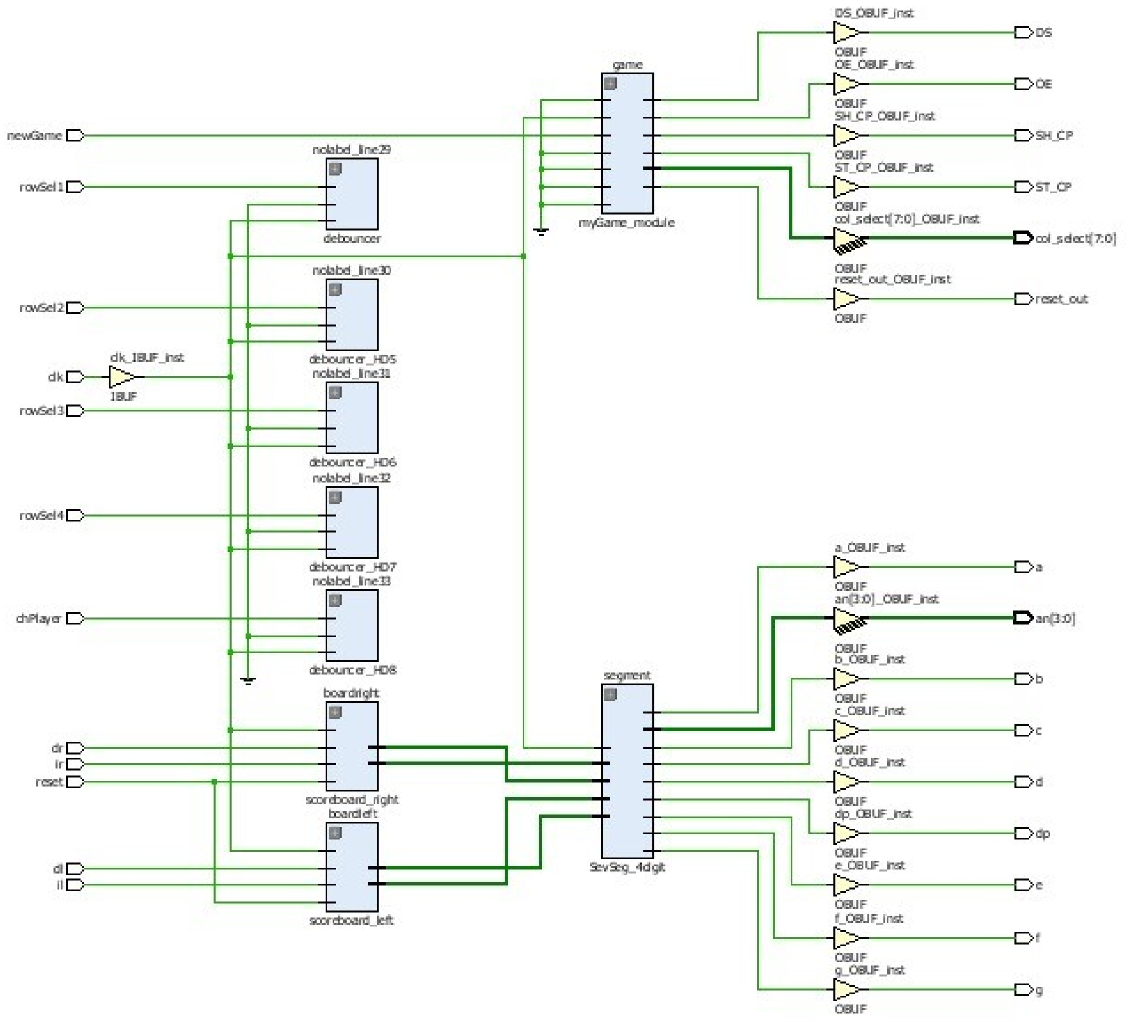
This part is for illustrating the winner in each turn:

```
      else if (image_red[0][0] == 0 && image_blue[1][2] == 0 && image_red[2][4] == 0  && image_blue[3][6] == 0)
      begin
        if(leftPlayer%2==1)
          begin
            image_red = {8'b00010000,
                  8'b00111000,
                  8'b01111100,
                  8'b11111110,
                  8'b00111000,
                  8'b00111000,
                  8'b00111000,
                  8'b00111000};

            image_green = {8'b00000000, 8'b00000000, 8'b00000000, 8'b00000000, 8'b00000000, 8'b00000000, 8'b00000000,
8'b00000000};
            image_blue = {8'b00010000, 8'b00111000, 8'b01111100, 8'b11111110, 8'b00111000, 8'b00111000, 8'b00111000,
8'b00111000};
          end
        else
          begin
            image_red = {8'b00000000, 8'b00000000, 8'b00000000, 8'b00000000, 8'b00000000, 8'b00000000, 8'b00000000,
8'b00000000};
            image_green = {8'b00111000, 8'b00111000, 8'b00111000, 8'b00111000, 8'b11111110, 8'b01111100, 8'b00111000,
8'b00010000};
            image_blue = {8'b00111000, 8'b00111000, 8'b00111000, 8'b00111000, 8'b11111110, 8'b01111100, 8'b00111000,
8'b00010000};
          end
      end
```

```
        end
    end

    // This module displays 8x8 image on LED display module.
    display_8x8 display_8x8_0(
        .clk(clk),

        // RGB data for display current column
        .red_vect_in(image_red[col_num]),
        .green_vect_in(image_green[col_num]),
        .blue_vect_in(image_blue[col_num]),

        .col_data_capture(), // unused
        .col_num(col_num),

        // FPGA pins for display
        .reset_out(reset_out),
        .OE(OE),
        .SH_CP(SH_CP),
        .ST_CP(ST_CP),
        .DS(DS),
        .col_select(col_select)
    );


endmodule
```

<u>display_8X8 module:</u> This code was also given and what it does is turns the leds on the rgb display to high according to our array.

The schematic for the general call module:



newGame
rowSel1
rowSel2
clk
clk_IBUF_inst
IBUF
rowSel3
rowSel4
chPlayer
dr
ir
reset
dl
il
nolabel_line29
debouncer
nolabel_line30
debouncer_HD5
nolabel_line31
debouncer_HD6
nolabel_line32
debouncer_HD7
nolabel_line33
debouncer_HD8
boardright
scoreboard_right
boardleft
scoreboard_left
game
myGame_module
segment
SevSeg_4digit
DS_OBUF_inst
OBUF
DS
OE_OBUF_inst
OBUF
OE
SH_CP_OBUF_inst
OBUF
SH_CP
ST_CP_OBUF_inst
OBUF
ST_CP
col_select[7:0]_OBUF_inst
OBUF
col_select[7:0]
reset_out_OBUF_inst
OBUF
reset_out
a_OBUF_inst
OBUF
a
an[3:0]_OBUF_inst
OBUF
an[3:0]
b_OBUF_inst
OBUF
b
c_OBUF_inst
OBUF
c
d_OBUF_inst
OBUF
d
dp_OBUF_inst
OBUF
dp
e_OBUF_inst
OBUF
e
f_OBUF_inst
OBUF
f
g_OBUF_inst
OBUF
g