Controle de Periféricos - Projeto 13 - Controle Remoto Infravermelho (IR)

Controle de Periféricos - Projeto 13 - Controle Remoto Infravermelho (IR)

Foto - Projeto em funcionamento

Código do projeto:

```
module IR( input wire FPGA_CLK, input wire FPGA_RST, input wire IR,output reg [3:0] DIG, output reg [7:0
        reg [3:0] led_cs;
        reg [7:0] led_db;

        // Sinais de entrada do FPGA
        wire clk = FPGA_CLK;
        wire rst_n = FPGA_RST;

        reg [7:0] led1,led2,led3,led4;
        reg [15:0] irda_data;    // save irda data,than send to 7 segment led
        reg [31:0] get_data;     // use for saving 32 bytes irda data
        reg [5:0]  data_cnt;     // 32 bytes irda data counter
        reg [2:0]  cs,ns;
        reg error_flag;          // 32 bytes data. Durante o sinalizador de erro de dados.

        //--------------------------------------------------------------------------
        reg irda_reg0;       // Para evitar a metaestabilidade e evitar a condução de vários registros,
        reg irda_reg1;       // Isso só pode ser usado, o programa a seguir representa o estado de irda
        reg irda_reg2;       // Para determinar a borda de irda, pressione o registrador novamente, o p
        wire irda_neg_pulse; // Borda de descida de irda
        wire irda_pos_pulse; // Borda de subida de irda
        wire irda_chang;     // Borda de transicao de irda

        reg[15:0] cnt_scan;// Contador de frequência de varredura

        reg [10:0] counter;  // Divisão de frequência 1750 vezes
        reg [8:0]  counter2; // Contador do número de ticks após a divisão de frequência
        wire check_9ms;  // check leader 9ms time
        wire check_4ms;  // check leader 4.5ms time
        wire low;        // check  data="0" time
        wire high;       // check  data="1" time


        // O registrador eh usado aqui
        always @ (posedge clk)
        begin
```

```
        if( !rst_n ) begin
            irda_reg0 <= 1'b0;
            irda_reg1 <= 1'b0;
            irda_reg2 <= 1'b0;
        end

        else begin
            led_cs <= 4'b0000;
            DIG <= led_cs;
            irda_reg0 <= IR;
            irda_reg1 <= irda_reg0;
            irda_reg2 <= irda_reg1;
        end
    end



    // Atribuicoes de variaveis
    assign irda_chang = irda_neg_pulse | irda_pos_pulse;    // Mudanças de sinal recebidas por IR (s
    assign irda_neg_pulse = irda_reg2 & (~irda_reg1);       // IR recebe sinal de irda nas bordas de
    assign irda_pos_pulse = (~irda_reg2) & irda_reg1;       // IR recebe sinal de irda nas bordas de


    //-------------------------------------------------------------------------
    // Valor dividido por 1750 contagens
    always @ (posedge clk)
    begin
        if (!rst_n) begin
            counter <= 11'd0;
        end

        else if( irda_chang ) begin  // Quando o nivel de irda mudar, recomecar a contagem
            counter <= 11'd0;
        end
        else if( counter == 11'd1750 ) begin
            counter <= 11'd0;
        end
        else begin
            counter <= counter + 1'b1;
        end
    end

    //-------------------------------------------------------------------------
    always @ ( posedge clk )
    begin
        if (!rst_n) begin
            counter2 <= 9'd0;
        end
        else if( irda_chang ) begin  //Quando o nivel de irda mudar, recomecar a contagem
            counter2 <= 9'd0;
        end
        else if( counter == 11'd1750 ) begin
            counter2 <= counter2 +1'b1;
```

```
    end
end


assign check_9ms = ((217 < counter2) & (counter2 < 297));
// 257 Para aumentar a estabilidade, faça um certo intervalo
assign check_4ms = ((88 < counter2) & (counter2 < 168));  //128
assign low  = ((6 < counter2) & (counter2 < 26));         // 16
assign high = ((38 < counter2) & (counter2 < 58));        // 48

//--------------------------------------------------------------------------------
// generate statemachine
parameter IDLE       = 3'b000, //Estado inicial
            LEADER_9   = 3'b001, //9ms
             LEADER_4   = 3'b010, //4ms
             DATA_STATE = 3'b100; //transferir dados



//
always @ (posedge clk)
begin
    if( !rst_n ) begin
        cs <= IDLE;
    end
    else begin
        cs <= ns;  //bit de status
        end
end



always @ ( * )
begin
        case (cs)
            IDLE:
              if (~irda_reg1)
                 ns = LEADER_9;
              else
                 ns = IDLE;

            LEADER_9:
              if (irda_pos_pulse)   //leader 9ms check
                 begin
                    if (check_9ms)
                      ns = LEADER_4;
                    else
                      ns = IDLE;
                 end
              else  // Completar a declaracao if-else para evitar gerar travas (problemas)
                 ns = LEADER_9;

            LEADER_4:
```

```
                if (irda_neg_pulse)  // leader 4.5ms check
                   begin
                      if (check_4ms)
                        ns = DATA_STATE;
                      else
                        ns = IDLE;
                   end
                else
                   ns = LEADER_4;

             DATA_STATE:
                if ((data_cnt == 6'd32) & irda_reg2 & irda_reg1)
                   ns = IDLE;
                else if (error_flag)
                   ns = IDLE;
                else
                   ns = DATA_STATE;

             // Caso 'default'
             default: ns = IDLE;
          endcase
    end

    // --> A saída na máquina de estados é descrita por um circuito sequencial
    always @ (posedge clk)
    begin
          if( !rst_n ) begin
              data_cnt <= 6'd0;
              get_data <= 32'd0;
              error_flag <= 1'b0;
          end

       else if ( cs == IDLE ) begin
              data_cnt <= 6'd0;
              get_data <= 32'd0;
              error_flag <= 1'b0;
       end

       else if ( cs == DATA_STATE ) begin
              if( irda_pos_pulse ) begin // low 0.56ms check
                  if( !low ) begin //error
                    error_flag <= 1'b1;
                  end
              end

              else if( irda_neg_pulse ) begin //check 0.56ms/1.68ms data 0/1
                  if ( low ) begin
                      get_data[0] <= 1'b0;
                  end

                  else if( high ) begin
                      get_data[0] <= 1'b1;
                  end
```

```
                else begin
                  error_flag <= 1'b1;
                end

                get_data[31:1] <= get_data[30:0];
                data_cnt <= data_cnt + 1'b1;
            end
        end
end



always @ (posedge clk)
begin
    if (!rst_n) begin
        irda_data <= 16'd0;
    end

    else if ((data_cnt ==6'd32) & irda_reg1) begin
        led1 <= get_data[7:0];   // Código inverso de dados
        led2 <= get_data[15:8];  //código de dados
        led3 <= get_data[23:16]; //Código de usuário
        led4 <= get_data[31:24];
    end
end

// Exibe as teclas pressionadas no controle remoto no tubo digital
always@(led2)
begin
    case(led2)
    // Valor do código da placa de controle remoto 0
    8'b01101000: led_db = 8'b1100_0000; // Mostrar o numero 0

    // Valor do código do botao 1 do controle remoto
    8'b00110000: led_db = 8'b1111_1001; // Mostrar o numero  1

    // Valor do código do botao 2 do controle remoto
    8'b00011000: led_db = 8'b1010_0100;  // Mostrar o numero 2

    // Valor do código do botao 3 do controle remoto
    8'b01111010: led_db = 8'b1011_0000;  // Mostrar o numero 3

    // Valor do código do botao 4 do controle remoto
    8'b00010000: led_db = 8'b1001_1001;  // Mostrar o numero 4

    // Valor do código do botao 5 do controle remoto
    8'b00111000: led_db = 8'b1001_0010;  // Mostrar o numero 5

    // Valor do código do botao 6 do controle remoto
    8'b01011010: led_db = 8'b1000_0010;  // Mostrar o numero  6

    // Valor do código do botao 7 do controle remoto
    8'b01000010: led_db = 8'b1111_1000;  // Mostrar o numero 7
```

```
            // Valor do código do botao 8 do controle remoto
            8'b01001010: led_db = 8'b1000_0000;  // Mostrar o numero 8

            // Valor do código do botao 9 do controle remoto
            8'b01010010: led_db = 8'b1001_0000;  // Mostrar o numero 9

            // Caso padrao: Quando nenhuma tecla é pressionada, exibe F
            default: led_db = 8'b1000_1110;

             endcase

             // Atribuicao
             SEG <= led_db;
        end
endmodule
```