Código 1: *Projeto Final- Relogio Digital VGA ANEXO 1 – VHDL MODULE*

```
--Codigo principal--
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use work.enviarS.all;

entity modulo is
         generic (  fclk: integer := 50_000;
                                 twindown: integer := 10
                 );
    Port (
        clk : in  STD_LOGIC;
        rst : in  STD_LOGIC;
        inicio_pausa : in  STD_LOGIC;
        conf : in  STD_LOGIC;
        s : out  STD_LOGIC_VECTOR (6 downto 0);
        anodo : out STD_LOGIC_VECTOR (3 downto 0);
                  Hsync, Vsync: buffer STD_LOGIC;--foi alterado somente para a simulaçao
        R, G : OUT STD_LOGIC_VECTOR(2 DOWNTO 0);
                  B: OUT STD_LOGIC_VECTOR(1 DOWNTO 0)
    );
end modulo;

architecture Behavioral of modulo is
    signal useg, umin, uhor : integer range 0 to 9 := 0;  -- Décimos
    signal dseg, dmin, dhor : integer range 0 to 6 := 0;  -- unidades


         --ESTADOS:
         type state is (rd, continua, pausa);
         signal pr_state, nx_state: state := rd;

         signal inicio_pausa_flag : std_logic := '0';
         signal escrito : std_logic := '0';
         signal enable : std_logic := '0';


         signal inicio_pausa_debouced  : std_logic;

         --rs:
         signal rst_temp: std_logic := '0';
    signal count: integer range 0 to (fclk * twindown) := 0;
         constant max: integer := (fclk * twindown);


begin
        -- segunda coisa do botao:
        process (clk, rst)
                 begin
                         if (clk'event and clk='1') then
                                 if(rst_temp /= rst) then
                                         count <= count + 1;
                                         if (count = max) then
                                                 rst_temp <= rst;
                                                 count <= 0;
                                         end if;
                                 else
                                         count <= 0;
                                 end if;
                         end if;
        end process;

    -- coisa do botao
    process(rst, inicio_pausa_debouced)
            begin
                    if (rst='1') then
                            inicio_pausa_flag <= '0';

                    elsif (inicio_pausa_debouced'event and inicio_pausa_debouced ='0') then
                            inicio_pausa_flag <= not inicio_pausa_flag;
                    end if;


     end process;

    process(rst,clk)
            begin
                    if (clk'event and clk='1') then
                            pr_state <= nx_state;
                    end if;
    end process;

--  tempo ->>. 1 segundo
process(pr_state)

begin
            case pr_state is
                    when continua =>
                            escrito <= '0';
                            enable <= '1';
                      if (inicio_pausa_flag = '1' and inicio_pausa_debouced = '1') then
```

```
                                nx_state <= pausa;
                        else
                                nx_state <= continua;
                        end if;

                when pausa =>
                        escrito <= '0';
                        enable <='0';
                        if(inicio_pausa_flag='0' and inicio_pausa_debouced ='1') then
                                nx_state <= continua;
                        else
                                nx_state <= pausa;
                        end if;

                when rd =>
                        escrito <= '1';
                        enable <='0';
                        if(inicio_pausa_flag='0' and inicio_pausa_debouced ='1') then
                                nx_state <= continua;
                        else
                                nx_state <= rd;
                        end if;
                end case;

end process;
```

```
process (enable, rst_temp,clk)
        variable temp : natural range 0 to 50_000_000 := 0;
begin
        if (rst_temp = '1') then
                                                temp := 0;
                                                useg <= 0;
                                                umin <= 0;
                                                uhor <= 0;
                                                dseg <= 0;
                                                dmin <= 0;
                                                dhor <= 0;
        elsif clk'event and clk = '1' then
                                        if enable = '1' then
                                                temp := temp + 1;
                                                if temp = 50_000_000 then
                                                        temp := 0;
                                                        useg <= useg + 1;
                                                              if useg = 9 then
                                                                      useg <= 0;
                                                                      dseg <= dseg + 1;
                                                                      if dseg = 5 then
                                                                              dseg <= 0;
                                                                              umin <= umin +1;
                                                                              if umin = 9 then
                                                                                      dmin <= dmin +1;
                                                                                      umin <= 0;
                                                                                      if dmin = 5  then
                                                                                              uhor <= uhor + 1;
                                                                                              dmin <= 0;
                                                                                              if (uhor = 9) then
                                                                                                      dhor <= dhor + 1;
                                                                                                      uhor <= 0;
                                                                                              elsif(uhor = 3 and dhor = 2) then
                                                                                                      uhor <=0;
                                                                                                      dhor <= 0;
                                                                                              end
                                                                                              ↪  if;
                                                                                      end if;
                                                                              end if;
                                                                      end if;
                                                              end if;
                                                end if;
                                        end if;

                        end if;
         end process;


display_ctrl : entity work.DisplayController
        Port map (
            clk => clk,
            conf => conf,
            umin => umin,
            dmin => dmin,
            uhor => uhor,
            dhor => dhor,
            useg => useg,
            dseg => dseg,
            s => s,
            anodo => anodo,
```

```
                        enable => escrito
);

        ligarBotao : entity work.controle_botao
generic map (
    fclk => fclk,
    twindown => twindown
)
port map (
    bt => inicio_pausa,
    clk => clk,
    temp_out => inicio_pausa_debouced
);

          vga :entity work.vga_componente_dig
                generic map (
                          -- Personalize os parâmetros se necessário
                          lado_grande => 60,
                          lado_curto => 20,
                          distancia_a_x => 0,
                          distancia_a_y => 0,
                          distancia_b_x => 40,
                          distancia_b_y => 20,
                          distancia_c_x => 40,
                          distancia_c_y => 80,
                          distancia_d_x => 0,
                          distancia_d_y => 140,
                          distancia_e_x => 0,
                          distancia_e_y => 80,
                          distancia_f_x => 0,
                          distancia_f_y => 20,
                          distancia_g_x => 0,
                          distancia_g_y => 80,
                          distancia_dez_uni => 80,
                          distancia_uni_dez => 200
                )
                port map (
                          clk       => clk,               -- Relógio de 50 MHz
                          Hsync     => Hsync,             -- Sinal de sincronização horizontal
                          Vsync     => Vsync,             -- Sinal de sincronização vertical
                          R         => R,                 -- Canal vermelho da saída VGA
                          G         => G,                 -- Canal verde da saída VGA
                          B         => B,                 -- Canal azul da saída VGA
                          secUnits  => useg,              -- Unidade dos segundos
                          secTens   => dseg,              -- Dezena dos segundos
                          minUnits  => umin,              -- Unidade dos minutos
                          minTens   => dmin,              -- Dezena dos minutos
                          hourUnits => uhor,              -- Unidade das horas
                          hourTens  => dhor,               -- Dezena das horas
                          escrito => escrito
                );

end Behavioral;

--Codigo do componente: controle do Display--
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
use work.enviarS.all;

entity DisplayController is
    Port (
        clk : in STD_LOGIC;
        conf : in STD_LOGIC;
        umin, dmin, uhor, dhor, useg, dseg : in INTEGER range 0 to 9; -- Entradas de dados para os displays
        s : out STD_LOGIC_VECTOR(6 downto 0);
        anodo : out STD_LOGIC_VECTOR(3 downto 0);
                  enable: in std_logic
    );
end DisplayController;



architecture Behavioral of DisplayController is
    signal temp_2 : natural range 0 to 12_500 := 0;
    signal display : natural range 0 to 3 := 0;
         signal temp_3     : natural range 0 to 50_000_000*10:= 0;
    signal char_index : natural range 0 to 15 := 0; -- Índice para rolar os caracteres
          -- Função para retornar os caracteres da mensagem "Relogio Digital"
    function get_char(index: natural) return STD_LOGIC_VECTOR is
        variable s: STD_LOGIC_VECTOR(6 downto 0);
    begin
        case index is
            when 0 => S := "0111001";--R
                                when 1 => S := "0110000";--E
                                when 2 => S := "1110001";--L
                                when 3 => S := "0000001";--O
```

```
                                when 4 => S := "0100000";--G
                                when 5 => S := "1001111";--I
                                when 6 => S := "0000001";--0
                                when 7 => S := "1111111";--espaço
                                when 8 => S := "1000010";--D
                                when 9 => S := "1001111";--I
            when 10 => s := "0100000"; -- G
            when 11 => s := "1001111"; -- I
            when 12 => s := "1110000"; -- T
            when 13 => s := "0001000"; -- A
            when 14 => s := "1110001"; -- L
            when others => s := "1111111"; -- Espaço (valor padrão)
        end case;
        return s;
    end function;

begin
 -- Processo para controlar o tempo de rolagem do texto
    process(clk)
    begin
       if clk'event and clk = '1' then
            temp_3 <= temp_3 + 1;

            if temp_3 = 50_000*100*10 then -- Espera antes de rolar o texto
                temp_3 <= 0;
                char_index <= (char_index + 1) mod 16;
            end if;
        end if;
    end process;

    process(clk)
    begin
        if clk'event and clk = '1' then
            temp_2 <= temp_2 + 1;
            if temp_2 = 12_500 then
                temp_2 <= 0;
                display <= display + 1;
                                    if display=3 then
                                     display<=0;
                                     end if;
            end if;
                        if enable = '1' then
                                case display is
                                        when 0 =>
                                                anodo <= "0111"; -- Primeiro display ativo
                                                s <= get_char((char_index) mod 16);
                                        when 1 =>
                                                anodo <= "1011"; -- Segundo display ativo
                                                s <= get_char((char_index + 1) mod 16);
                                        when 2 =>
                                                anodo <= "1101"; -- Terceiro display ativo
                                                s <= get_char((char_index + 2) mod 16);
                                        when others =>
                                                anodo <= "1110"; -- Quarto display ativo
                                                s <= get_char((char_index + 3) mod 16);
                          end case;

                        else
            if conf = '1' then
                case display is
                    when 0 =>
                        anodo <= "1110";
                        s <= bcd_to_SSD(umin);
                    when 1 =>
                        anodo <= "1101";
                        s <= bcd_to_SSD(dmin);
                    when 2 =>
                        anodo <= "1011";
                        s <= bcd_to_SSD(uhor);
                    when others =>
                        anodo <= "0111";
                        s <= bcd_to_SSD(dhor);
                end case;
            else
                case display is
                    when 0 =>
                        anodo <= "1110";
                        s <= bcd_to_SSD(useg);
                    when 1 =>
                        anodo <= "1101";
                        s <= bcd_to_SSD(dseg);
                    when 2 =>
                        anodo <= "1011";
                        s <= bcd_to_SSD(umin);
                    when others =>
                        anodo <= "0111";
                        s <= bcd_to_SSD(dmin);
```

```
                end case;
            end if;
                                end if;
        end if;
    end process;
end Behavioral;

--Codigo do componente: VGA--
LIBRARY ieee;
USE ieee.std_logic_1164.all;
use work.enviarS.all;

ENTITY vga_componente_dig IS
    GENERIC (
        --- Horizontal (em pixels)
        Ha: INTEGER := 120; -- Hsync Pulse
        Hb: INTEGER := 120+64; -- Hsync Pulse + Back Porch
        Hc: INTEGER := 120+64+800; -- Hsync Pulse + Back Porch + Active Video
        Hd: INTEGER := 120+64+800+56; -- Hsync Pulse + Back Porch + Active Video + Front Porch

        -- Vertical (em linhas)
        Va: INTEGER := 6;    -- Vsync Pulse
        Vb: INTEGER := 6+23;  -- Vsync Pulse + Back Porch
        Vc: INTEGER := 6+23+600; -- Vsync Pulse + Back Porch + Active Video
        Vd: INTEGER := 6+23+600+37; -- Vsync Pulse + Back Porch + Active Video + Front Porch

    lado_grande: integer := 20 * 3;
    lado_curto: integer := 20;

    -- Posição inicial de dígitos (ajuste conforme necessário)
    distancia_a_x: integer := 0;
    distancia_a_y: integer := 0;
    distancia_b_x: integer := 20 * 2;
    distancia_b_y: integer := 20 * 1;
    distancia_c_x: integer := 20 * 2;
    distancia_c_y: integer := 20 * 4;
    distancia_d_x: integer := 0;
    distancia_d_y: integer := 20 * 7;
    distancia_e_x: integer := 0;
    distancia_e_y: integer := 20 * 4;
    distancia_f_x: integer := 0;
    distancia_f_y: integer := 20 * 1;
    distancia_g_x: integer := 0;
    distancia_g_y: integer := 20 * 4;

    distancia_dez_uni: integer := 20 * 4;
    distancia_uni_dez: integer := 20 * 10
);

PORT (
    clk: IN STD_LOGIC;                      -- 50MHz in our board

    Hsync, Vsync: buffer STD_LOGIC;--mudado
    R, G : OUT STD_LOGIC_VECTOR(2 DOWNTO 0);
              B: OUT STD_LOGIC_VECTOR(1 DOWNTO 0);
    nblanck, nsync: OUT STD_LOGIC;
              secUnits: in NATURAL RANGE 0 TO 9;
              secTens: in NATURAL RANGE 0 TO 5;
              minUnits: in NATURAL RANGE 0 TO 9;
              minTens:  in NATURAL RANGE 0 TO 7;
              hourUnits: in NATURAL RANGE 0 TO 9;
              hourTens: in NATURAL RANGE 0 TO 2;
              escrito: in std_logic
    );
END vga_componente_dig;

ARCHITECTURE vga_componente_dig OF vga_componente_dig IS
    SIGNAL Hactive, Vactive, dena: STD_LOGIC;

    SIGNAL digit_secU, digit_secT, digit_minU, digit_minT, digit_hourU, digit_hourT: STD_LOGIC_VECTOR(6 DOWNTO 0);
    SIGNAL X: INTEGER RANGE 0 TO Hd;
    SIGNAL Y: INTEGER RANGE 0 TO Vd;

         TYPE tipopontos IS ARRAY (0 to 11, 0 TO 26) OF std_logic;
        signal pontos:tipopontos := (
        ('1','1','1','0','1','1','1','0','1','0','0','0','1','1','1','0','1','1','1','0','0','1','0','0','1','1','1'),
        ('1','0','1','0','1','0','0','0','1','0','0','0','1','0','1','0','1','0','0','0','0','0','0','0','1','0','1'),
        ('1','1','0','0','1','1','0','0','1','0','0','0','1','0','1','0','1','1','1','0','0','1','0','0','1','0','1'),
        ('1','0','1','0','1','0','0','0','1','0','0','0','1','0','1','0','1','0','1','0','0','1','0','0','1','0','1'),
        ('1','0','1','0','1','1','1','0','1','1','1','0','1','1','1','0','1','1','1','0','0','1','0','0','1','1','1'),
        ('0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'),
        ('0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0','0'),
        ('1','1','0','0','0','1','0','0','1','1','1','0','0','1','0','0','1','1','1','0','1','1','1','0','1','0','0'),
        ('1','0','1','0','0','0','0','0','1','0','0','0','0','0','0','0','0','1','0','0','1','0','1','0','1','0','0'),
        ('1','0','1','0','0','1','0','0','1','1','1','0','0','1','0','0','0','1','0','0','1','0','1','0','1','0','0'),
        ('1','0','1','0','0','1','0','0','1','0','1','0','0','1','0','0','0','1','0','0','1','1','1','0','1','0','0'),
```

```
    ('1','1','0','0','0','1','0','0','1','1','1','0','0','1','0','0','0','1','0','0','1','0','1','0','1','1','1'));


-- Pontos de nício (Horizontal e Vertical)
TYPE ponto IS ARRAY (0 TO 1) OF integer RANGE 0 TO Hd;
SIGNAL ponto_inicio: ponto := ((hb+10)+110, vb+160);

TYPE umdigito IS ARRAY (0 TO 6, 0 TO 2) OF INTEGER RANGE 0 TO Hd;
CONSTANT digito: umdigito := (
    (0, distancia_a_x, distancia_a_y),  -- Segmento a
    (1, distancia_b_x, distancia_b_y),  -- Segmento b
    (1, distancia_c_x, distancia_c_y),  -- Segmento c
    (0, distancia_d_x, distancia_d_y),  -- Segmento d
    (1, distancia_e_x, distancia_e_y),  -- Segmento e
    (1, distancia_f_x, distancia_f_y),  -- Segmento f
    (0, distancia_g_x, distancia_g_y)   -- Segmento g

);

TYPE posicaodigito IS ARRAY (0 TO 5) OF INTEGER RANGE 0 TO Hd;
CONSTANT posicoes: posicaodigito := (
    0, -- dezena de hora
    distancia_dez_uni,  -- unidade de hora
    distancia_uni_dez, -- dezena de minuto
    distancia_dez_uni + distancia_uni_dez, -- unidade de minuto
    2 * distancia_uni_dez, -- dezena de segundo
    2 * distancia_uni_dez + distancia_dez_uni -- unidade de segundo
    );

BEGIN
    ----------------------------------------------------
    -- Part 1: CONTROL GENERATOR
    ----------------------------------------------------
    nblanck <= '1';  -- No direct blanking
    nsync <= '0';    -- No sync on green



    -- Horizontal signals generation:
    PROCESS(clk)
        VARIABLE Hcount: INTEGER RANGE 0 TO Hd;
    BEGIN
        IF (clk'EVENT AND clk = '1') THEN
            Hcount := Hcount + 1;
            IF (Hcount = Ha) THEN
                Hsync <= '1';
            ELSIF (Hcount = Hb) THEN
                Hactive <= '1';
            ELSIF (Hcount = Hc) THEN
                Hactive <= '0';
            ELSIF (Hcount = Hd) THEN
                Hsync <= '0';
                Hcount := 0;
            END IF;
        END IF;
        X <= Hcount;
    END PROCESS;

    -- Vertical signals generation:
    PROCESS(Hsync)
        VARIABLE Vcount: INTEGER RANGE 0 TO Vd;
    BEGIN
        IF (Hsync'EVENT AND Hsync = '0') THEN
            Vcount := Vcount + 1;
            IF (Vcount = Va) THEN
                Vsync <= '1';
            ELSIF (Vcount = Vb) THEN
                Vactive <= '1';
            ELSIF (Vcount = Vc) THEN
                Vactive <= '0';
            ELSIF (Vcount = Vd) THEN
                Vsync <= '0';
                Vcount := 0;
            END IF;
        END IF;
        Y <= Vcount;
    END PROCESS;

    -- Display enable generation:
    dena <= Hactive AND Vactive;

    ----------------------------------------------------
    -- Part 3: IMAGE GENERATOR
    ----------------------------------------------------
PROCESS(Hsync, Vsync, Vactive, dena, X, Y,secUnits,secTens,minUnits,minTens,hourUnits,hourTens,ponto_inicio)
        VARIABLE i, j, i1, j1: INTEGER RANGE 0 TO 26;
```

```
BEGIN
            R <= "000";
            G <= "000";
            B <= "00";
    -- Make BCD to SSD conversion:
    digit_secU <= bcd_to_SSD(secUnits);
    digit_secT <= bcd_to_SSD(secTens);
    digit_minU <= bcd_to_SSD(minUnits);
    digit_minT <= bcd_to_SSD(minTens);
    digit_hourU <= bcd_to_SSD(hourUnits);
    digit_hourT <= bcd_to_SSD(hourTens);

              IF (dena = '1') THEN
                    if ( escrito = '1') then

                                    FOR j IN 0 TO 26 LOOP
                                          FOR i IN 0 TO 11 LOOP
                                          j1 := j;
                                          i1 := i;
                                                    IF (((ponto_inicio(0) + j1*lado_curto) <= X) AND X < ((ponto_inicio(0) +
                                                    ↪  (j1+1)*lado_curto)))THEN
                                                              if ((ponto_inicio(1)+i1*lado_curto <= Y) and (Y <
                                                              ↪  ponto_inicio(1)+lado_curto+i1*lado_curto)) then
                                                              R <= (OTHERS => pontos(i, j));
                                                    END IF;
                                                END IF;
                                          END LOOP;
                                          END LOOP;


                  else


                          -- Ponto ":" entre unidade de hora e dezena de minuto
                  --IF X >= ponto_inicio(0) and x < ponto_inicio(0)+ 50*lado_curto AND Y >= ponto_inicio(1) and y <
                  ↪  ponto_inicio(1)+ 11*lado_curto then
                  IF ((X >= ponto_inicio(0) + lado_curto*8) AND
                           (X <  ponto_inicio(0) + lado_curto*9)) THEN
                           -- Ponto superior
                           IF ((Y >= ponto_inicio(1)/2 + lado_grande * 2 + lado_curto ) AND
                                    (Y < ponto_inicio(1)/2 + lado_grande * 3 )) THEN
                                    R <= "111"; -- Cor para o ponto
                                    G <= "000";
                                    B <= "00";
                           END IF;

                           -- Ponto inferior
                           IF ((Y >= ponto_inicio(1)/2 + lado_grande * 3 + lado_curto ) AND
                                    (Y < ponto_inicio(1)/2 + lado_grande * 4 )) THEN
                                    R <= "111"; -- Cor para o ponto
                                    G <= "000";
                                    B <= "00";
                           END IF;
                  END IF;

                  -- Ponto ":" entre unidade de minuto e dezena de segundo
                  IF ((X >= ponto_inicio(0)+lado_curto*18) AND
                           (X < ponto_inicio(0) + lado_curto*19)) THEN
                           -- Ponto superior
                           IF ((Y >= ponto_inicio(1)/2 + lado_grande * 2 + lado_curto) AND
                                    (Y < ponto_inicio(1)/2+ lado_grande * 3)) THEN
                                    R <= "111"; -- Cor para o ponto
                                    G <= "000";
                                    B <= "00";
                           END IF;

                           -- Ponto inferior
                           IF ((Y >= ponto_inicio(1)/2 + lado_grande * 3 + lado_curto ) AND
                                    (Y < ponto_inicio(1)/2 + lado_grande * 4 )) THEN
                                    R <= "111"; -- Cor para o ponto
                                    G <= "000";
                                    B <= "00";
                           END IF;

                              END IF;

    FOR i IN 6 downTO 0 LOOP

       FOR j IN 0 TO 5 LOOP

             IF ((ponto_inicio(0) + digito(i, 1) + posicoes(j))) <= X AND
                 X < ((ponto_inicio(0) + digito(i, 1) + lado_grande * (1 - digito(i, 0)) + lado_curto * digito(i, 0)) +
                 ↪  posicoes(j)) THEN
                 IF ((ponto_inicio(1) + digito(i, 2))) <= Y AND
                     Y < (ponto_inicio(1) + digito(i, 2) + lado_grande * digito(i, 0) + lado_curto * (1 - digito(i, 0))) THEN
```

```
                                                IF (j = 0) THEN
    R <= (OTHERS => (NOT digit_hourT(6 - i)));
                                                        G <= (OTHERS => '0');
                                                              B <= (OTHERS => '0');

ELSIF (j = 1) THEN
    R <= (OTHERS => (NOT digit_hourU(6 - i)));
                                                        G <= (OTHERS => '0');
                                                              B <= (OTHERS => '0');

ELSIF (j = 2) THEN

                                                              R <= (OTHERS => (NOT digit_minT(6 - i)));
                                                              G <= (OTHERS => '0');
                                                              B <= (OTHERS => '0');


ELSIF (j = 3) THEN
    R <= (OTHERS => (NOT digit_minU(6 - i)));
                                                        G <= (OTHERS => '0');
                                                              B <= (OTHERS => '0');

ELSIF (j = 4) THEN
    R <= (OTHERS => (NOT digit_secT(6 - i)));
                                                        G <= (OTHERS => '0');
                                                              B <= (OTHERS => '0');

ELSE
    R<= (OTHERS => (NOT digit_secU(6 - i)));
                                                        G <= (OTHERS => '0');
                                                              B <= (OTHERS => '0');
                                                end if;
                                                    END IF;

                                                END IF;
              END LOOP;

            END LOOP;

                          END IF;
        ELSE
            G <= (OTHERS => '0');
            R <= (OTHERS => '0');
            B <= (OTHERS => '0');
        END IF;
    END PROCESS;
END vga_componente_dig;



--Codigo do componente: controle do botao--
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity controle_botao is
    generic (
        fclk : integer := 50_000;
        twindown : integer := 10
    );
    port (
        bt : in std_logic;
        clk : in std_logic;
        temp_out : out std_logic
    );
end controle_botao;

architecture Behavioral of controle_botao is
    signal temp : std_logic := '0';
    signal count : integer range 0 to (fclk * twindown) := 0;
    constant max : integer := fclk * twindown;
begin
    process (clk)
    begin
        if (clk'event and clk='1') then
            if (temp /= bt) then
                count <= count + 1;
                if (count = max) then
                    temp <= bt;
                    count <= 0;
                end if;
            else
                count <= 0;
            end if;
        end if;
    end process;

    temp_out <= temp;


end Behavioral;
```

Código 2: *Controlador de porta de Garagem – VHDL TEST BENCH*

```
--------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date:   13:55:38 12/06/2024
-- Design Name:
-- Module Name:   /home/eloiza/Relogio_Digital_pf/Relogio_Digital_pf_tb.vhd
-- Project Name:  Relogio_Digital
--------------------------------------------------------------------------------

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY Relogio_Digital_pf_tb IS
END Relogio_Digital_pf_tb;

ARCHITECTURE behavior OF Relogio_Digital_pf_tb IS
      -- Component declaration for the Unit Under Test (UUT)
    component garagem
        generic ( fclk: integer := 1;
                                                twindown: integer := 1;
                                                tempo_max_1: integer := 200;
                                                freq : integer := 1
                                                );

    -- Declaração do componente do módulo
    COMPONENT modulo
    PORT(
         clk : IN  std_logic;
         rst : IN  std_logic;
         inicio_pausa : IN  std_logic;
         conf : IN  std_logic;
         s : OUT  std_logic_vector(6 downto 0);
         anodo : OUT  std_logic_vector(3 downto 0);
         Hsync : buffer  std_logic;  -- Atualizado para OUT
         Vsync : buffer  std_logic;  -- Atualizado para OUT
         R : OUT  std_logic_vector(2 downto 0);
         G : OUT  std_logic_vector(2 downto 0);
         B : OUT  std_logic_vector(1 downto 0)
        );

    END COMPONENT;


    -- Declaração de sinais de entrada
    signal clk : std_logic := '0';
    signal rst : std_logic := '0';
    signal inicio_pausa : std_logic := '0';
    signal conf : std_logic := '0';

    -- Declaração de sinais de saída
    signal s : std_logic_vector(6 downto 0);
    signal anodo : std_logic_vector(3 downto 0);
    signal Hsync : std_logic;
    signal Vsync : std_logic;
    signal R : std_logic_vector(2 downto 0);
    signal G : std_logic_vector(2 downto 0);
    signal B : std_logic_vector(1 downto 0);

    -- Sinais intermediários para evitar conflitos
    signal Hsync_internal : std_logic;
    signal Vsync_internal : std_logic;

    -- Variável de controle de simulação
    signal simulacao : integer range 0 to 3 := 0;

    -- Período do clock
    constant clk_period : time := 10 ns;

BEGIN

    -- Instância da Unidade em Teste (UUT)
    uut: modulo PORT MAP (
          clk => clk,
          rst => rst,
          inicio_pausa => inicio_pausa,
          conf => conf,
          s => s,
          anodo => anodo,
          Hsync => Hsync_internal,  -- Usando sinais intermediários
          Vsync => Vsync_internal,  -- Usando sinais intermediários
          R => R,
          G => G,
          B => B
        );
```

```
-- Mapear sinais intermediários para os sinais do testbench
Hsync <= Hsync_internal;
Vsync <= Vsync_internal;

-- Processo para gerar o clock
clk_process : process
begin
    clk <= '0';
    wait for clk_period / 2;
    clk <= '1';
    wait for clk_period / 2;
end process;

-- Processo de estímulos
stimulus_process : process
begin
    case simulacao is
        -- Simulação 0: Teste básico do relógio
        when 0 =>
            rst <= '1';
            wait for clk_period * 20;
            rst <= '0';
            wait for clk_period * 10;
            inicio_pausa <= '1';
            wait for clk_period * 20;
            inicio_pausa <= '0';
            wait for clk_period * 20;

        -- Simulação 1: Verificação dos sinais de sincronização VGA
        when 1 =>
            rst <= '1';
            wait for clk_period * 10;
            rst <= '0';
            wait for clk_period * 20;
            -- Testa Hsync e Vsync
            wait for clk_period * 20;

        -- Simulação 2: Teste da máquina de estados
        when 2 =>
            rst <= '1';
            wait for clk_period * 10;
            rst <= '0';
            wait for clk_period * 20;
            inicio_pausa <= '1';
            wait for clk_period * 20;
            inicio_pausa <= '0';
            wait for clk_period * 20;
            conf <= '1';
            wait for clk_period * 20;
            conf <= '0';
            wait for clk_period * 20;

        -- Fim das simulações
        when others =>
            wait;
    end case;
    wait;
end process;

END behavior;
```