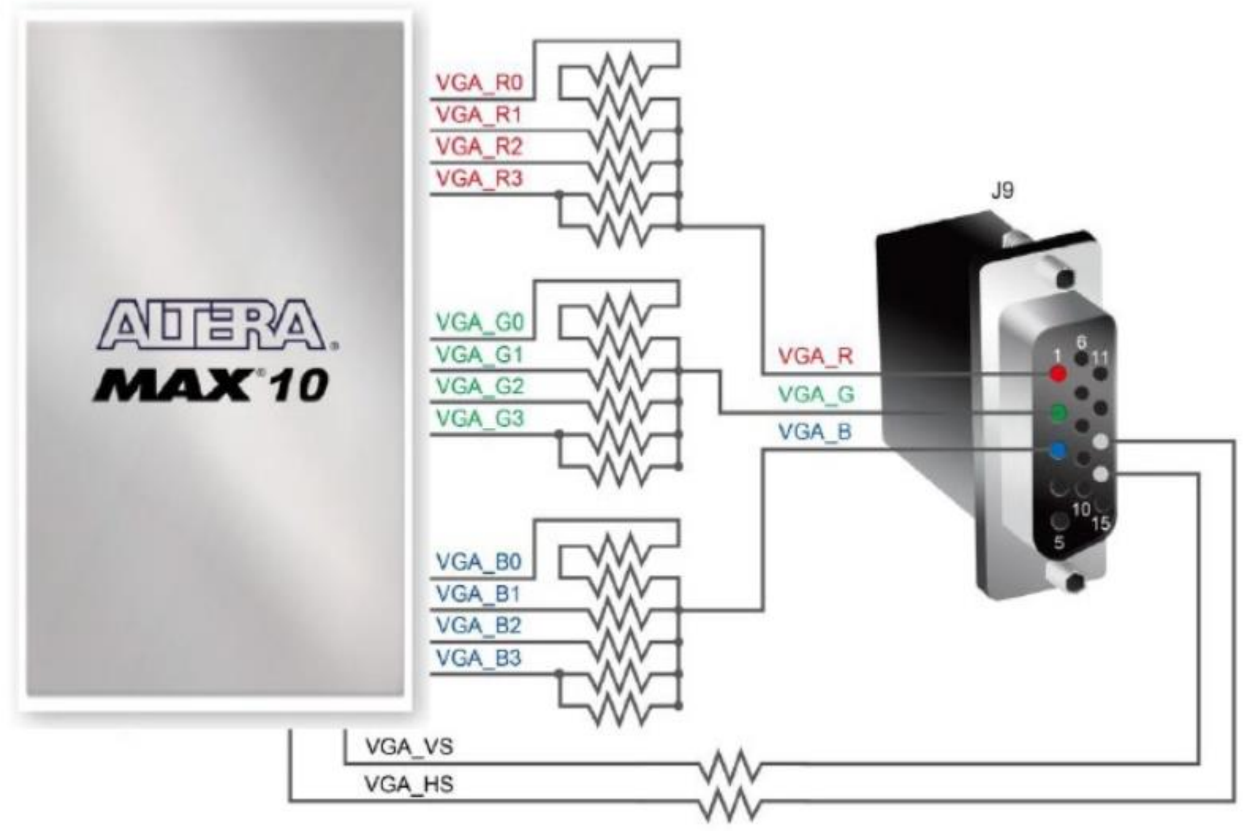
Controlador VGA

J. Emilio Soriano Campos A00829390

Este proyecto consistió en construir un controlador para VGA en VHDL el cual mostrara una cruz y un fondo ambos del color que quisiéramos.



Especificaciones

Las especificaciones para el proyecto eran las que se muestran en la imagen. En resumen, tendríamos un espacio de 800 X 525 pixeles dentro del cual habría un espacio de 640 X 480 pixeles de zona visible mientras que los demás pixeles serian el back y front porch de las sincronías horizontal y vertical.

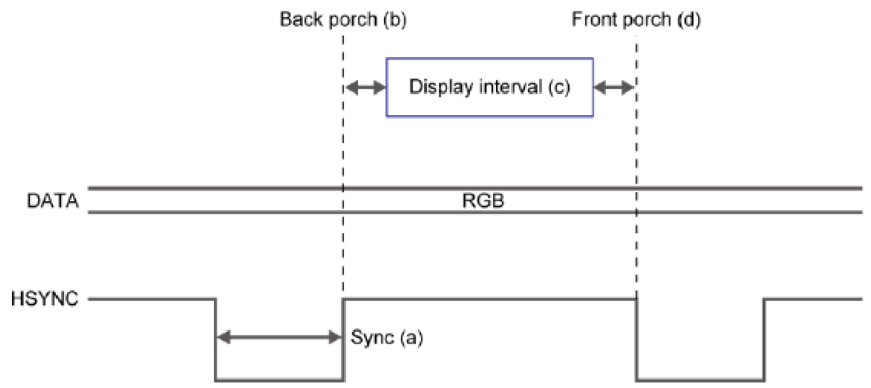
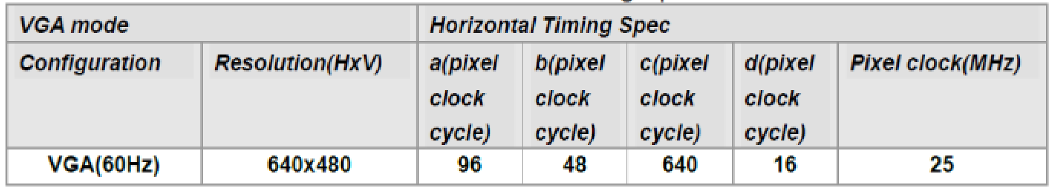


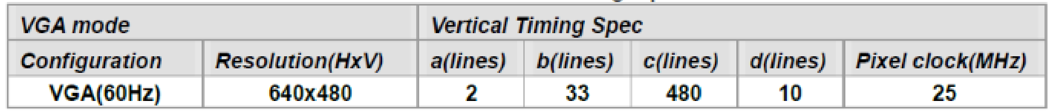
Figure 3-22 VGA horizontal timing specification

Table 3-9 VGA Horizontal Timing Specification



VGA mode		*Horizontal Timing Spec*				
Configuration	*Resolution(HxV)*	*a(pixel clock cycle)*	*b(pixel clock cycle)*	*c(pixel clock cycle)*	*d(pixel clock cycle)*	*Pixel clock(MHz)*
VGA(60Hz)	**640x480**	**96**	**48**	**640**	**16**	**25**

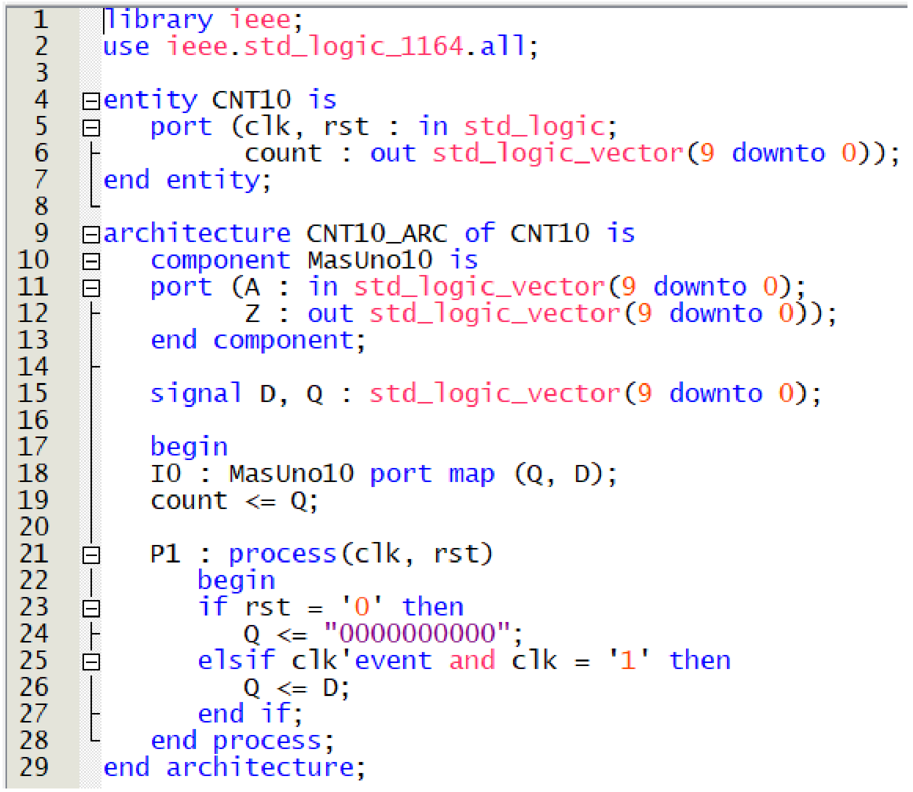
Table 3-10 VGA Vertical Timing Specification



VGA mode		*Vertical Timing Spec*				
Configuration	*Resolution(HxV)*	*a(lines)*	*b(lines)*	*c(lines)*	*d(lines)*	*Pixel clock(MHz)*
VGA(60Hz)	**640x480**	**2**	**33**	**480**	**10**	**25**

Contadores en cascada

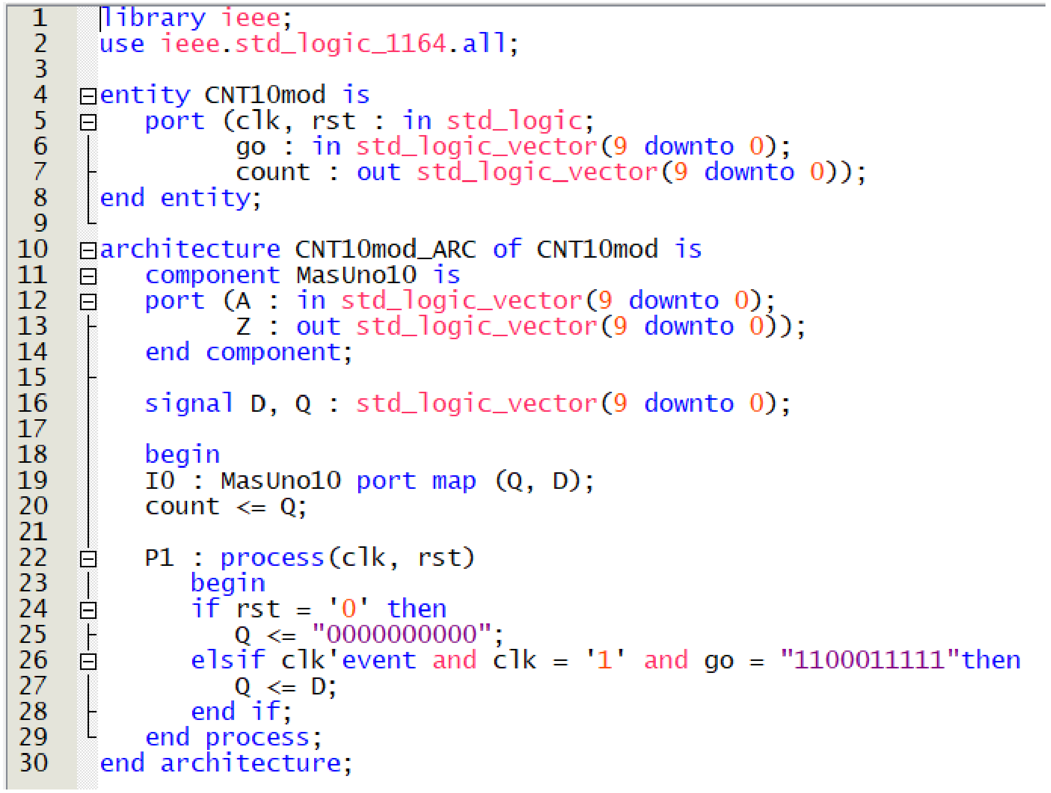
El primer paso para este proyecto fue construir dos contadores: uno de modulo 800 y otro de modulo 525 y que además el contador de modulo 525 solo avanzara cada vez que el contador de modulo 800 llegara a su valor final (799). Para esto hice dos contadores de 10 bits, uno de ellos solo contaría si recibía un 799 en binario.



```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity CNT10 is
    port (clk, rst : in std_logic;
          count : out std_logic_vector(9 downto 0));
end entity;

architecture CNT10_ARC of CNT10 is
    component MasUno10 is
    port (A : in std_logic_vector(9 downto 0);
          Z : out std_logic_vector(9 downto 0));
    end component;

    signal D, Q : std_logic_vector(9 downto 0);

    begin
    I0 : MasUno10 port map (Q, D);
    count <= Q;

    P1 : process(clk, rst)
       begin
       if rst = '0' then
          Q <= "0000000000";
       elsif clk'event and clk = '1' then
          Q <= D;
       end if;
    end process;
end architecture;
```

<= Contador normal

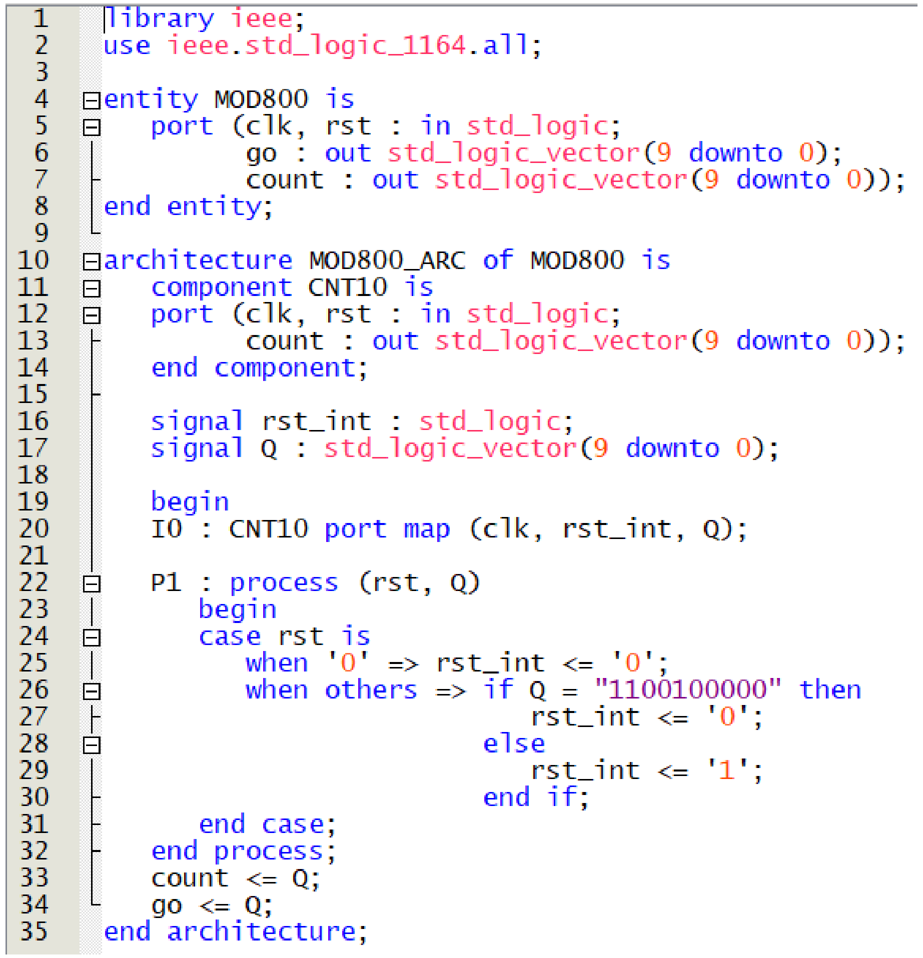
=> Contador modificado



```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity CNT10mod is
    port (clk, rst : in std_logic;
          go : in std_logic_vector(9 downto 0);
          count : out std_logic_vector(9 downto 0));
end entity;

architecture CNT10mod_ARC of CNT10mod is
    component MasUno10 is
    port (A : in std_logic_vector(9 downto 0);
          Z : out std_logic_vector(9 downto 0));
    end component;

    signal D, Q : std_logic_vector(9 downto 0);

    begin
    I0 : MasUno10 port map (Q, D);
    count <= Q;

    P1 : process(clk, rst)
       begin
       if rst = '0' then
          Q <= "0000000000";
       elsif clk'event and clk = '1' and go = "1100011111"then
          Q <= D;
       end if;
    end process;
end architecture;
```

Contadores en cascada

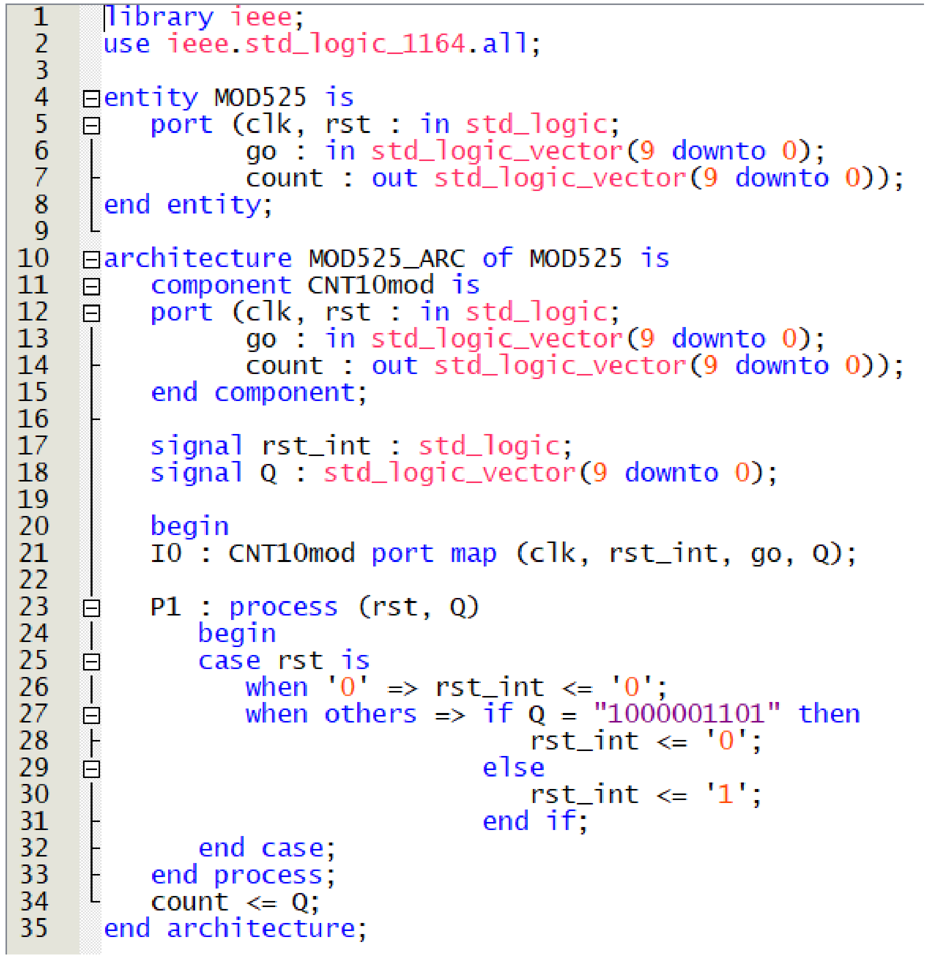
Con ambos contadores de 10 bits listos ahora solo tenia que restringir uno a que contara hasta 800 y otro que solo contara hasta 525.



```
library ieee;
use ieee.std_logic_1164.all;

entity MOD800 is
   port (clk, rst : in std_logic;
         go : out std_logic_vector(9 downto 0);
         count : out std_logic_vector(9 downto 0));
end entity;

architecture MOD800_ARC of MOD800 is
   component CNT10 is
   port (clk, rst : in std_logic;
         count : out std_logic_vector(9 downto 0));
   end component;

   signal rst_int : std_logic;
   signal Q : std_logic_vector(9 downto 0);

   begin
   I0 : CNT10 port map (clk, rst_int, Q);

   P1 : process (rst, Q)
      begin
      case rst is
         when '0' => rst_int <= '0';
         when others => if Q = "1100100000" then
                           rst_int <= '0';
                        else
                           rst_int <= '1';
                        end if;

      end case;
   end process;
   count <= Q;
   go <= Q;
end architecture;
```

<= Contador de modulo 800

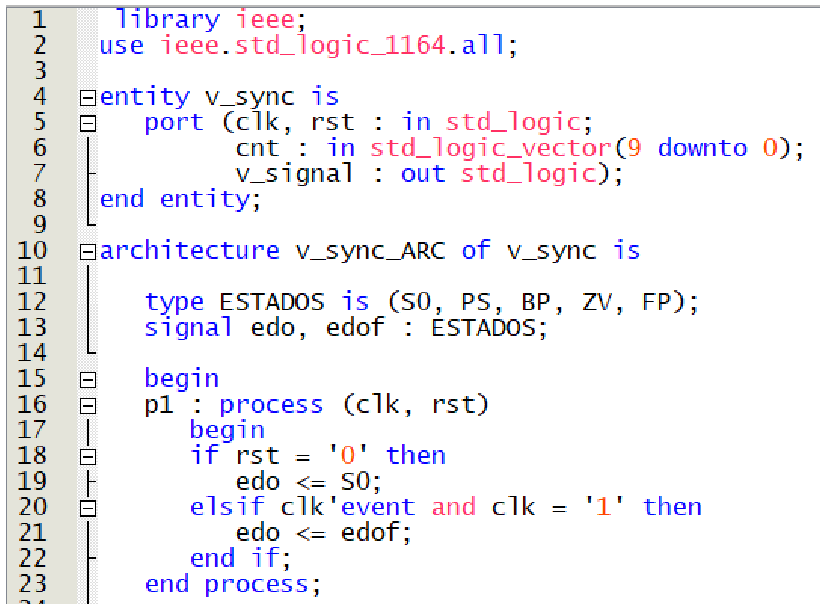
=> Contador de modulo 525



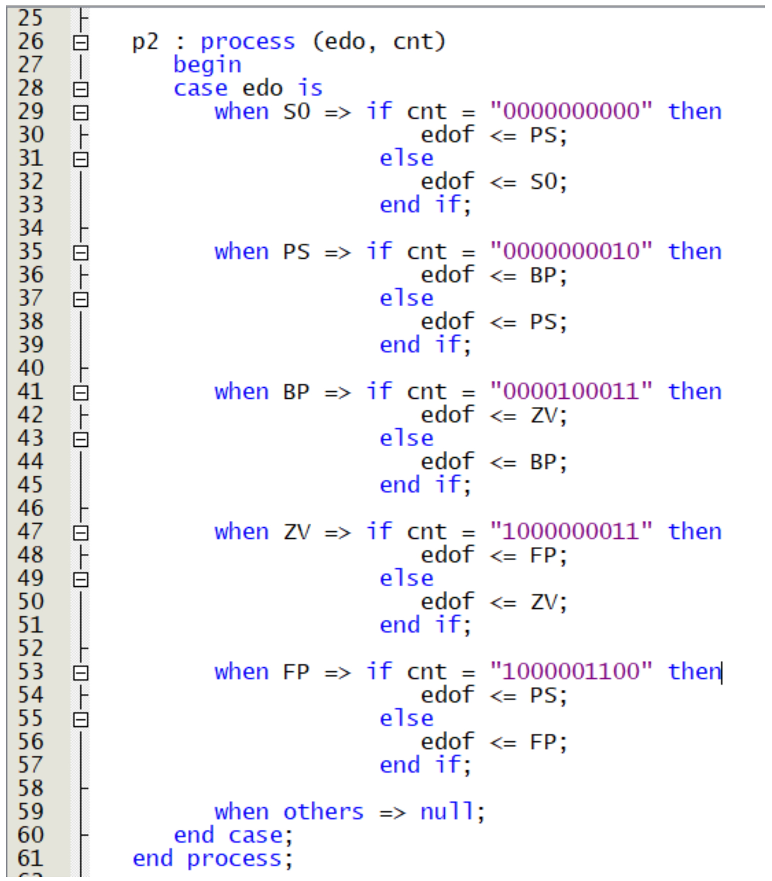
```
library ieee;
use ieee.std_logic_1164.all;

entity MOD525 is
   port (clk, rst : in std_logic;
         go : in std_logic_vector(9 downto 0);
         count : out std_logic_vector(9 downto 0));
end entity;

architecture MOD525_ARC of MOD525 is
   component CNT10mod is
   port (clk, rst : in std_logic;
         go : in std_logic_vector(9 downto 0);
         count : out std_logic_vector(9 downto 0));
   end component;

   signal rst_int : std_logic;
   signal Q : std_logic_vector(9 downto 0);

   begin
   I0 : CNT10mod port map (clk, rst_int, go, Q);

   P1 : process (rst, Q)
      begin
      case rst is
         when '0' => rst_int <= '0';
         when others => if Q = "1000001101" then
                           rst_int <= '0';
                        else
                           rst_int <= '1';
                        end if;
      end case;
   end process;
   count <= Q;
end architecture;
```

Sincronía vertical

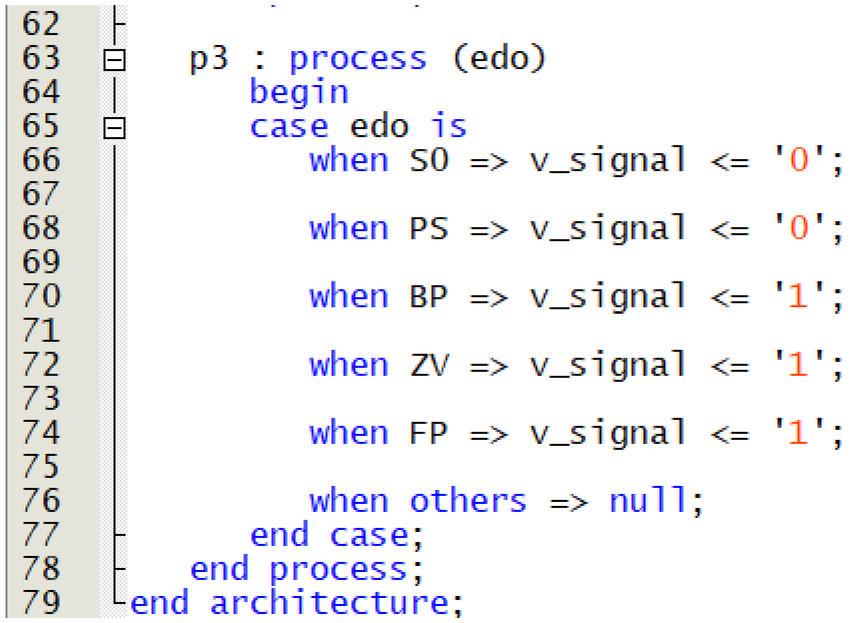
Para saber en donde me encontraba verticalmente necesitaba construir una maquina de estados que tuviera 4 estados mas un estado IDLE. Estos estados dependerían del valor del contador modulo 525.



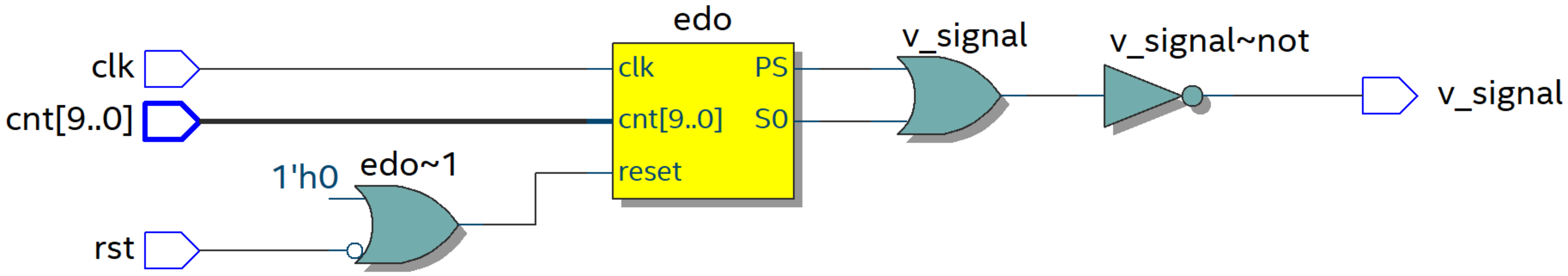
```
 library ieee;
use ieee.std_logic_1164.all;

entity v_sync is
   port (clk, rst : in std_logic;
         cnt : in std_logic_vector(9 downto 0);
         v_signal : out std_logic);
end entity;

architecture v_sync_ARC of v_sync is

   type ESTADOS is (S0, PS, BP, ZV, FP);
   signal edo, edof : ESTADOS;

   begin
   p1 : process (clk, rst)
      begin
      if rst = '0' then
         edo <= S0;
      elsif clk'event and clk = '1' then
         edo <= edof;
      end if;
   end process;
```

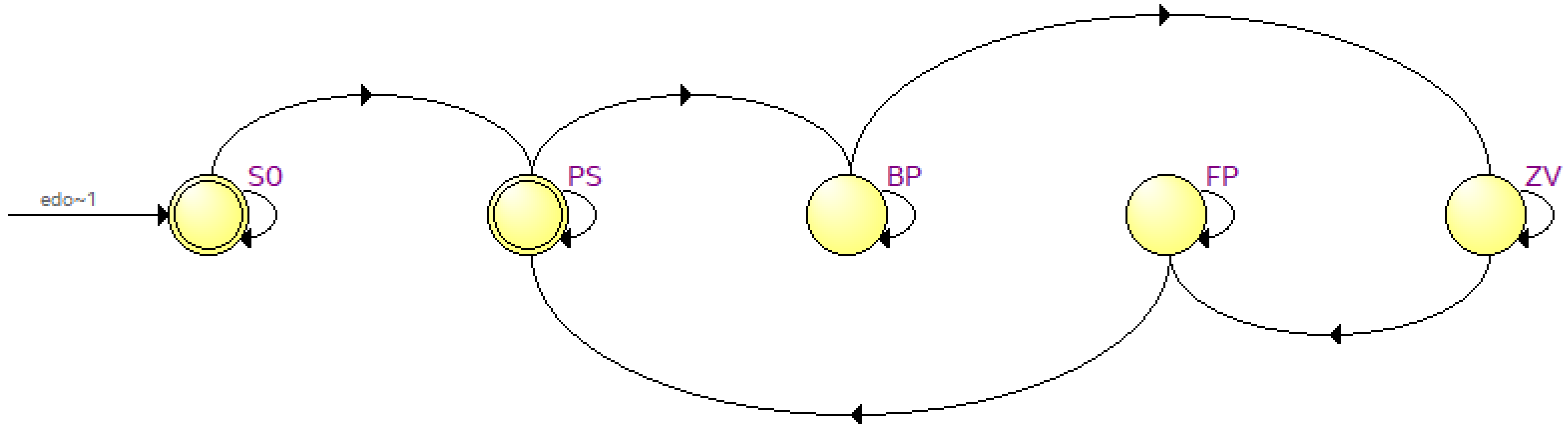


```
   p2 : process (edo, cnt)
      begin
      case edo is
         when S0 => if cnt = "0000000000" then
                         edof <= PS;
                     else
                         edof <= S0;
                     end if;

         when PS => if cnt = "0000000010" then
                         edof <= BP;
                     else
                         edof <= PS;
                     end if;

         when BP => if cnt = "0000100011" then
                         edof <= ZV;
                     else
                         edof <= BP;
                     end if;

         when ZV => if cnt = "1000000011" then
                         edof <= FP;
                     else
                         edof <= ZV;
                     end if;

         when FP => if cnt = "1000001100" then
                         edof <= PS;
                     else
                         edof <= FP;
                     end if;

         when others => null;
      end case;
   end process;
```



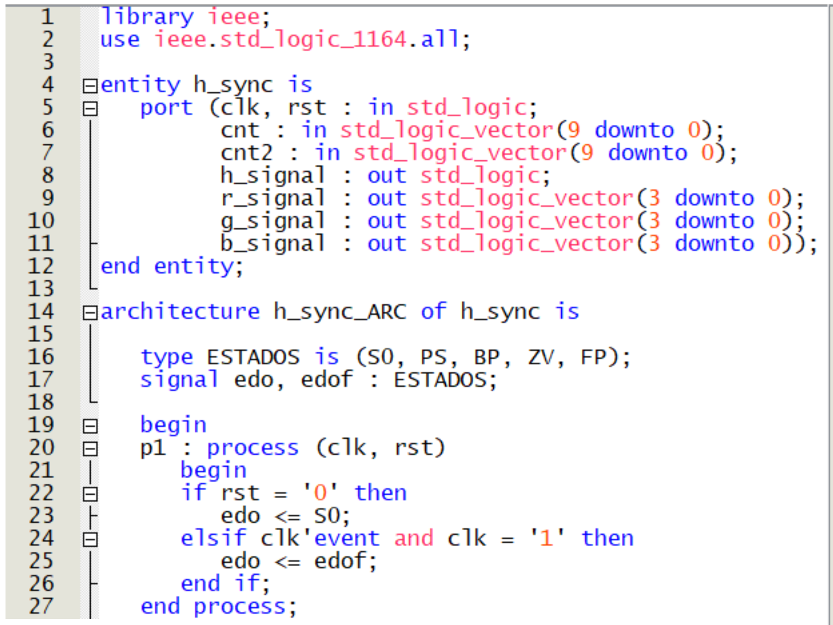
```
   p3 : process (edo)
      begin
      case edo is
         when S0 => v_signal <= '0';

         when PS => v_signal <= '0';

         when BP => v_signal <= '1';

         when ZV => v_signal <= '1';

         when FP => v_signal <= '1';

         when others => null;
      end case;
   end process;
end architecture;
```


edo
clk
cnt[9..0]
1'h0
edo~1
rst
clk
PS
cnt[9..0]
S0
reset
v_signal
v_signal~not
v_signal

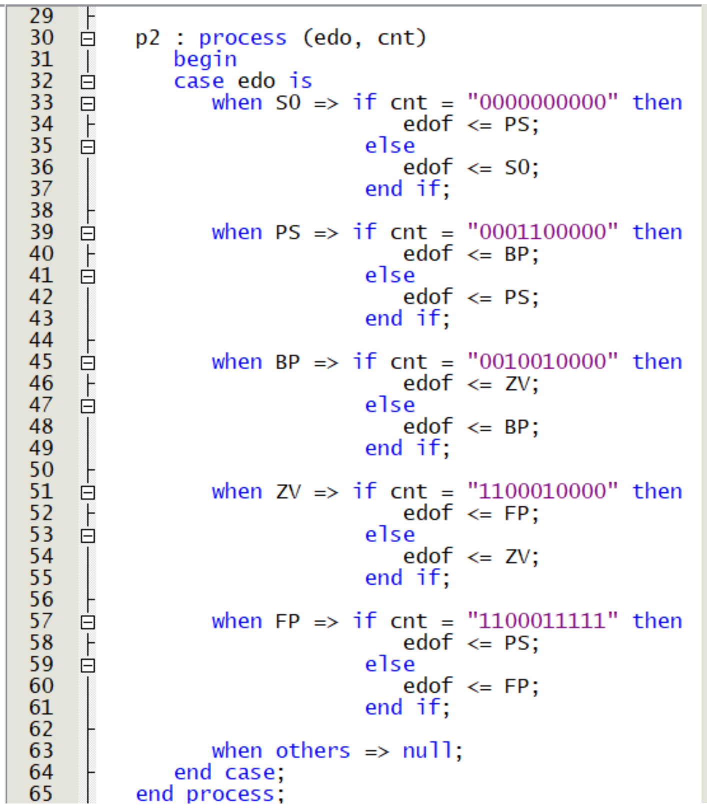

edo~1
S0
PS
BP
FP
ZV

Sincronía horizontal

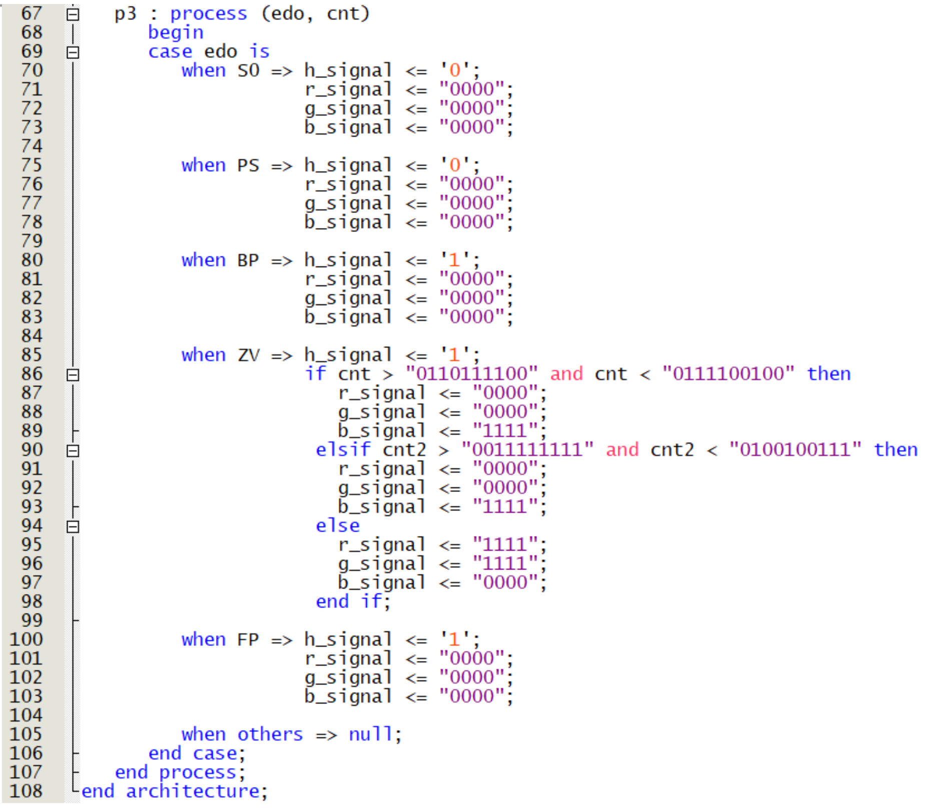
Para la sincronía horizontal no solo necesitaría saber donde me encontraba horizontalmente sino que también verticalmente ya que esta maquina de estados definiría la salida de los valores para las entradas de RGB. Esta maquina de estados definía sus estados con el contador de modulo 800 pero definía lo que pintaría con ese mismo contador mas el contador de modulo 525.



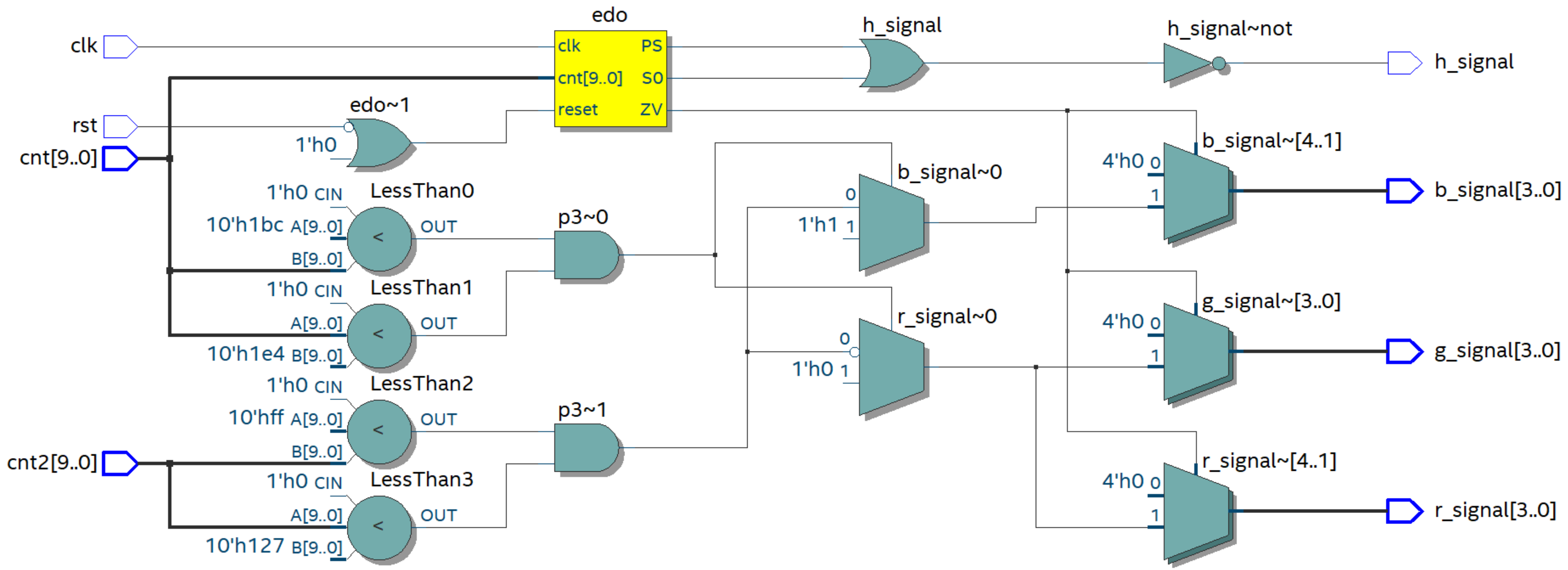
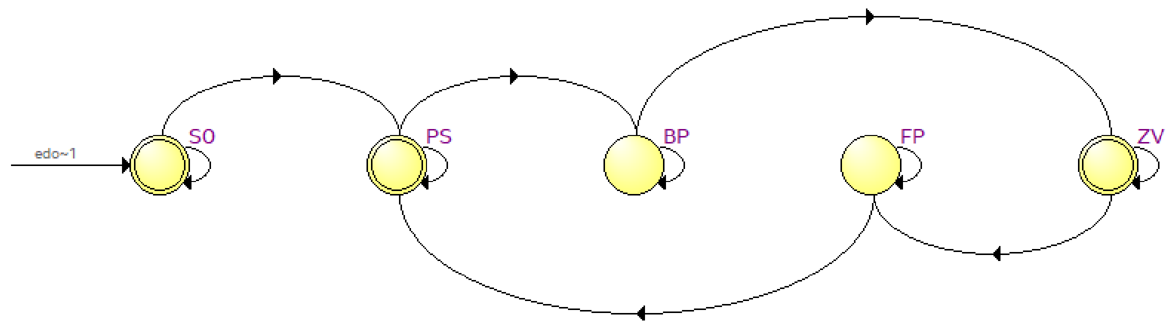
```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity h_sync is
   port (clk, rst : in std_logic;
         cnt : in std_logic_vector(9 downto 0);
         cnt2 : in std_logic_vector(9 downto 0);
         h_signal : out std_logic;
         r_signal : out std_logic_vector(3 downto 0);
         g_signal : out std_logic_vector(3 downto 0);
         b_signal : out std_logic_vector(3 downto 0));
end entity;

architecture h_sync_ARC of h_sync is

   type ESTADOS is (S0, PS, BP, ZV, FP);
   signal edo, edof : ESTADOS;

   begin
   p1 : process (clk, rst)
      begin
      if rst = '0' then
         edo <= S0;
      elsif clk'event and clk = '1' then
         edo <= edof;
      end if;
   end process;
```



```vhdl

p2 : process (edo, cnt)
   begin
   case edo is
      when S0 => if cnt = "0000000000" then
                     edof <= PS;
                  else
                     edof <= S0;
                  end if;

      when PS => if cnt = "0001100000" then
                     edof <= BP;
                  else
                     edof <= PS;
                  end if;

      when BP => if cnt = "0010010000" then
                     edof <= ZV;
                  else
                     edof <= BP;
                  end if;

      when ZV => if cnt = "1100010000" then
                     edof <= FP;
                  else
                     edof <= ZV;
                  end if;

      when FP => if cnt = "1100011111" then
                     edof <= PS;
                  else
                     edof <= FP;
                  end if;

      when others => null;
   end case;
end process;
```

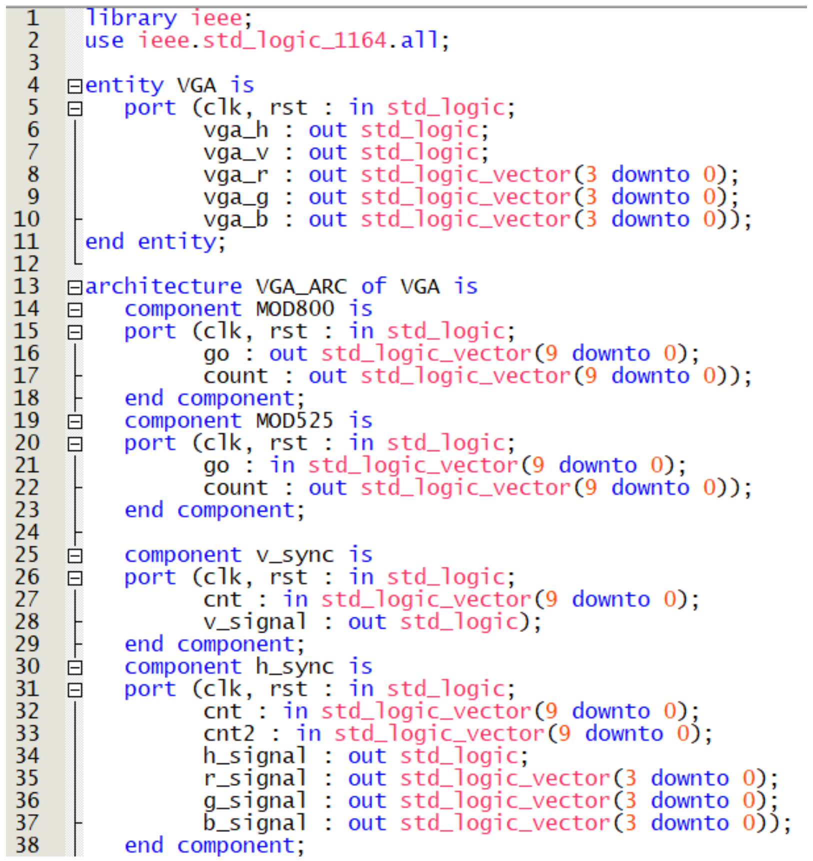


```vhdl
   p3 : process (edo, cnt)
      begin
      case edo is
         when S0 => h_signal <= '0';
                    r_signal <= "0000";
                    g_signal <= "0000";
                    b_signal <= "0000";

         when PS => h_signal <= '0';
                    r_signal <= "0000";
                    g_signal <= "0000";
                    b_signal <= "0000";

         when BP => h_signal <= '1';
                    r_signal <= "0000";
                    g_signal <= "0000";
                    b_signal <= "0000";

         when ZV => h_signal <= '1';
                    if cnt > "0110111100" and cnt < "0111100100" then
                       r_signal <= "0000";
                       g_signal <= "0000";
                       b_signal <= "1111";
                     elsif cnt2 > "0011111111" and cnt2 < "0100100111" then
                       r_signal <= "0000";
                       g_signal <= "0000";
                       b_signal <= "1111";
                     else
                       r_signal <= "1111";
                       g_signal <= "1111";
                       b_signal <= "0000";
                     end if;

         when FP => h_signal <= '1';
                    r_signal <= "0000";
                    g_signal <= "0000";
                    b_signal <= "0000";

         when others => null;
      end case;
   end process;
end architecture;
```

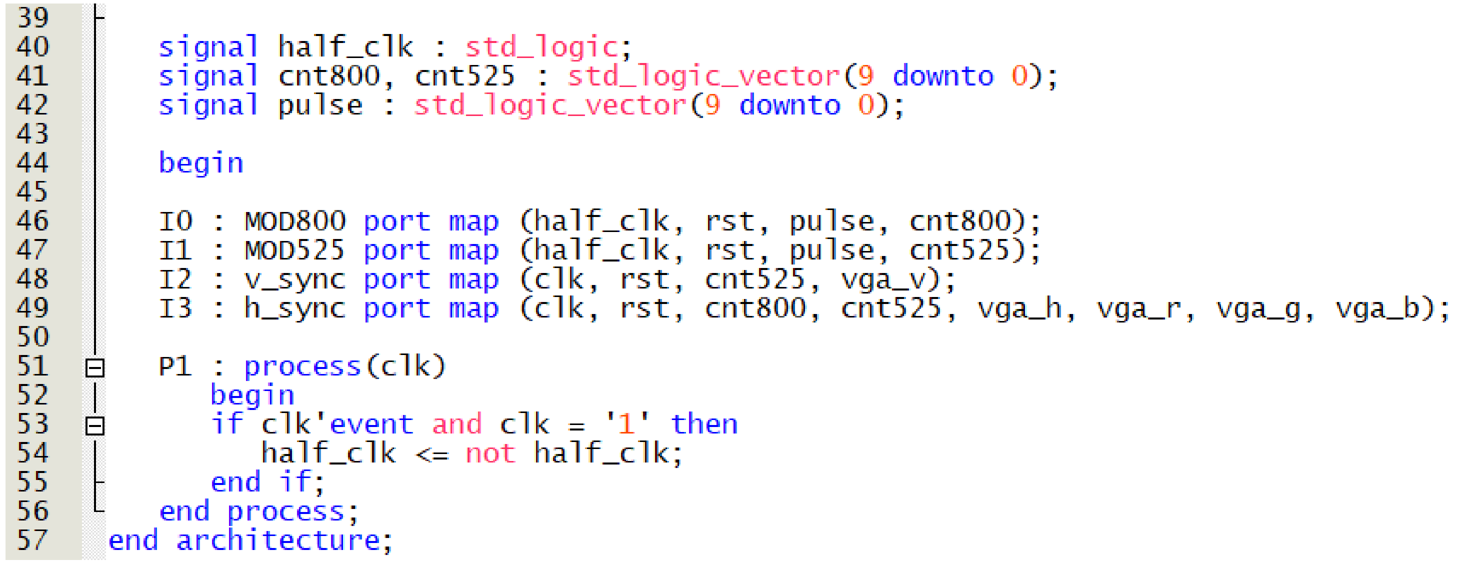

clk
rst
cnt[9..0]
cnt2[9..0]
edo
clk
cnt[9..0]
reset
PS
S0
ZV
edo~1
1'h0
h_signal
h_signal~not
h_signal
LessThan0
1'h0 CIN
10'h1bc A[9..0]
B[9..0]
OUT
LessThan1
1'h0 CIN
A[9..0]
10'h1e4 B[9..0]
OUT
LessThan2
1'h0 CIN
10'hff A[9..0]
B[9..0]
OUT
LessThan3
1'h0 CIN
A[9..0]
10'h127 B[9..0]
OUT
p3~0
p3~1
b_signal~0
0
1'h1 1
r_signal~0
0
1'h0 1
b_signal~[4..1]
4'h0 0
1
b_signal[3..0]
g_signal~[3..0]
4'h0 0
1
g_signal[3..0]
r_signal~[4..1]
4'h0 0
1
r_signal[3..0]

edo~1
S0
PS
BP
FP
ZV

Top-level design

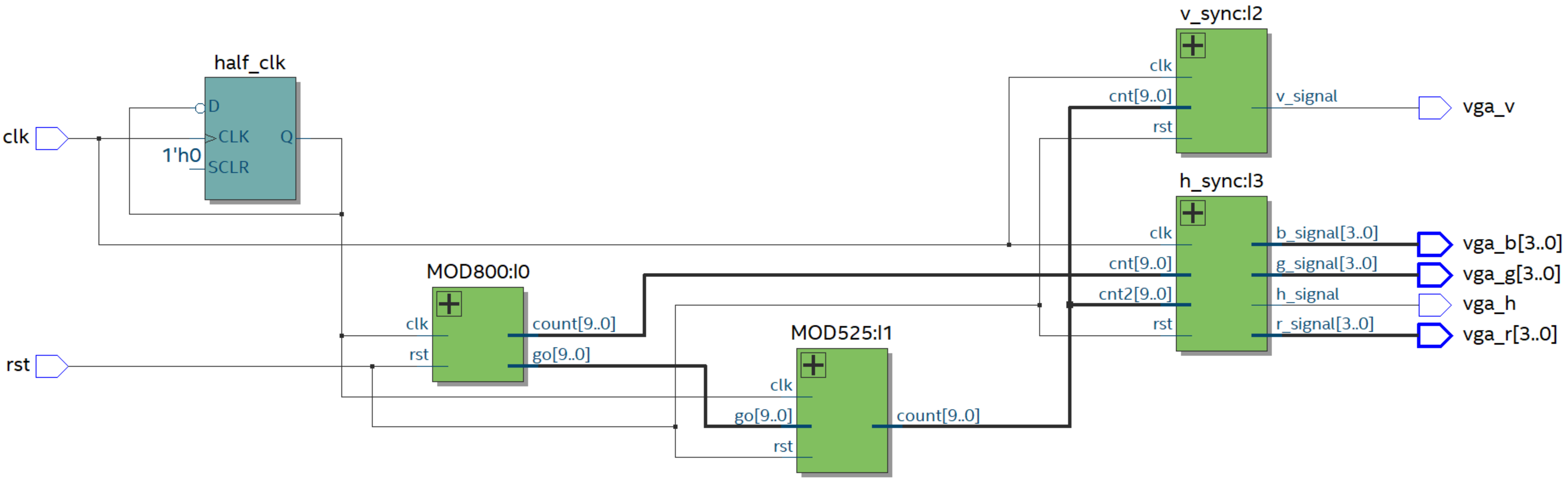
Una vez que tenia todos los componentes solo fue cuestión de conectarlos. También fue necesario hacer un divisor de señal de reloj ya que la tarjeta solo tiene dos señales de 50 MHz y una de 10 MHz y yo necesitaba una de 25 MHz pero no considere necesario hacer otro componente para ese paso y en cambio esta instanciado al final de la arquitectura.



```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity VGA is
    port (clk, rst : in std_logic;
            vga_h : out std_logic;
            vga_v : out std_logic;
            vga_r : out std_logic_vector(3 downto 0);
            vga_g : out std_logic_vector(3 downto 0);
            vga_b : out std_logic_vector(3 downto 0));
end entity;

architecture VGA_ARC of VGA is
    component MOD800 is
    port (clk, rst : in std_logic;
            go : out std_logic_vector(9 downto 0);
            count : out std_logic_vector(9 downto 0));
    end component;
    component MOD525 is
    port (clk, rst : in std_logic;
            go : in std_logic_vector(9 downto 0);
            count : out std_logic_vector(9 downto 0));
    end component;

    component v_sync is
    port (clk, rst : in std_logic;
            cnt : in std_logic_vector(9 downto 0);
            v_signal : out std_logic);
    end component;
    component h_sync is
    port (clk, rst : in std_logic;
            cnt : in std_logic_vector(9 downto 0);
            cnt2 : in std_logic_vector(9 downto 0);
            h_signal : out std_logic;
            r_signal : out std_logic_vector(3 downto 0);
            g_signal : out std_logic_vector(3 downto 0);
            b_signal : out std_logic_vector(3 downto 0));
    end component;
```



```vhdl

    signal half_clk : std_logic;
    signal cnt800, cnt525 : std_logic_vector(9 downto 0);
    signal pulse : std_logic_vector(9 downto 0);

    begin

    I0 : MOD800 port map (half_clk, rst, pulse, cnt800);
    I1 : MOD525 port map (half_clk, rst, pulse, cnt525);
    I2 : v_sync port map (clk, rst, cnt525, vga_v);
    I3 : h_sync port map (clk, rst, cnt800, cnt525, vga_h, vga_r, vga_g, vga_b);

    P1 : process(clk)
        begin
        if clk'event and clk = '1' then
            half_clk <= not half_clk;
        end if;
    end process;
end architecture;
```


half_clk
D
CLK
Q
1'h0
SCLR
clk
rst
MOD800:I0
clk
rst
count[9..0]
go[9..0]
MOD525:I1
clk
go[9..0]
rst
count[9..0]
v_sync:I2
clk
cnt[9..0]
rst
v_signal
vga_v
h_sync:I3
clk
cnt[9..0]
cnt2[9..0]
rst
b_signal[3..0]
g_signal[3..0]
h_signal
r_signal[3..0]
vga_b[3..0]
vga_g[3..0]
vga_h
vga_r[3..0]

Top-level design

La demostración del funcionamiento de este programa esta en el siguiente link: https://youtu.be/9SZKdfC1Xu0