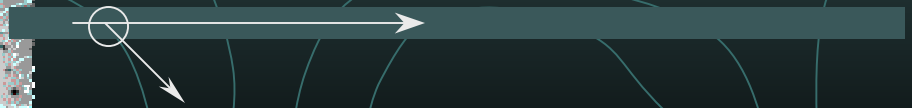
4.3 Modelos de interfaz



Sistemas digitales con VHDL

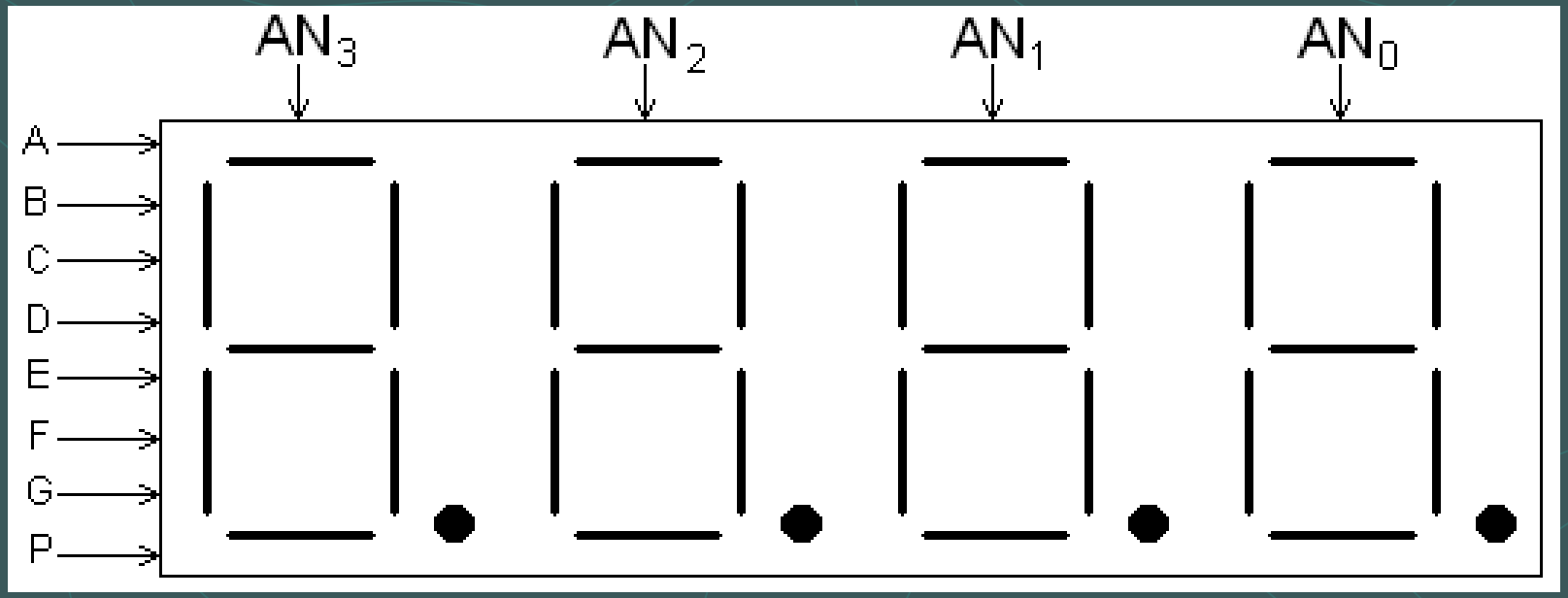
CIO

René de J. Romero Troncoso

Modelos de interfaz

- Manejo de los exhibidores de 7 segmentos
- Manejo de la pantalla VGA 1
- Manejo de la pantalla VGA 2

Exhibidores de 7 segmentos



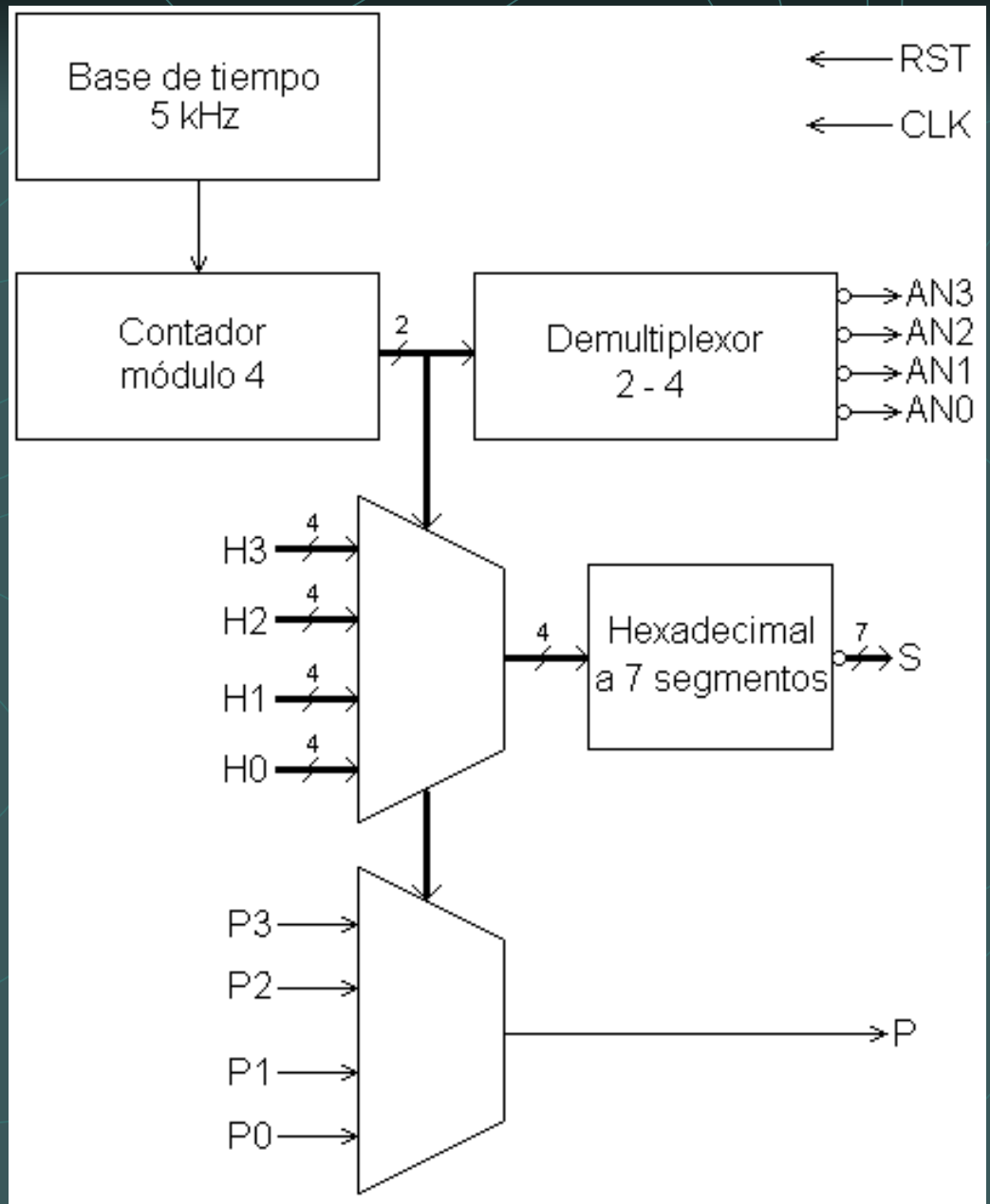
Características generales

- Segmentos comunes
- Ánodos multiplexados
- Señales activas en bajo
- Requiere un sistema digital para su manejo

Terminales asignadas en el FPGA

Señal	Terminal FPGA
AN3	E13
AN2	F14
AN1	G14
AN0	D14
A	E14
B	G13
C	N15
D	P15
E	R16
F	F13
G	N16
P	P16

Modelo de interfaz dinámica



Generador de base de tiempo

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity Modulo_10000 is
   port(
      RST : in  std_logic; -- Reset maestro
      CLK : in  std_logic; -- Reloj maestro
      Q   : out std_logic  -- Cero
      );
   end Modulo_10000;

architecture Cascada of Modulo_10000 is

-- Estados internos
signal Qp, Qn : std_logic_vector(13 downto 0);

-- Detector de 9999
signal Z      : std_logic;

-- Incrementador
signal I      : std_logic_vector(13 downto 0);
```

```
begin

   Detector: process(Qp)
   begin
      Z <=                                    NOT(Qp(13)) OR     Qp(12) OR
               Qp(11)  OR NOT(Qp(10)) OR NOT(Qp(9))  OR NOT(Qp(8)) OR
               Qp(7)   OR     Qp(6)   OR     Qp(5)   OR     Qp(4)  OR
            NOT(Qp(3))  OR NOT(Qp(2))  OR NOT(Qp(1))  OR NOT(Qp(0));
   end process Detector;

   Incrementador: process(Qp)
   begin
      I <= Qp + 1;
   end process Incrementador;

   Anulador: process(Z,I)
   begin
      for j in 0 to 13 loop
         Qn(j) <= Z AND I(j);
      end loop;
      Q <= NOT(Z);
   end process Anulador;

   Secuencial: process(RST,CLK)
   begin
      if (RST='1') then
         Qp <= (others => '0');
      elsif (CLK'event and CLK='1') then
         Qp <= Qn;
      end if;
   end process Secuencial;

end Cascada;
```

Contador módulo 4

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity Modulo_4 is
   port(
      RST : in  std_logic;                        -- Reset maestro
      CLK : in  std_logic;                        -- Reloj maestro
      H   : in  std_logic;                        -- Habilitacion de entrada
      Q   : out std_logic_vector(1 downto 0)  -- Salida
      );
   end Modulo_4;

architecture Cascada of Modulo_4 is

-- Estados internos
signal Qp, Qn : std_logic_vector(1 downto 0);
```

```
begin

   Combinacional: process(H,Qp)
   begin
      if (H='1') then
         Qn <= Qp + 1;
      else
         Qn <= Qp;
      end if;
      Q <= Qp;
   end process Combinacional;

   Secuencial: process(RST,CLK)
   begin
      if (RST='1') then
         Qp <= (others => '0');
      elsif (CLK'event and CLK='1') then
         Qp <= Qn;
      end if;
   end process Secuencial;

end Cascada;
```

Demultiplexor 2 - 4

```
library IEEE;
use IEEE.std_logic_1164.all;

entity Demux_2_4 is
   port(
      Q : in  std_logic_vector(1 downto 0); -- Anodo activo
      Y : out std_logic_vector(3 downto 0)  -- Anodos
      );
   end Demux_2_4;

architecture Simple of Demux_2_4 is
begin
   process(Q)
   begin
      case Q is
         when "00"   => Y <= "1110";
         when "01"   => Y <= "1101";
         when "10"   => Y <= "1011";
         when others => Y <= "0111";
      end case;
   end process;
end Simple;
```

Multiplexor cuádruple 4 - 1

```
library IEEE;
use IEEE.std_logic_1164.all;

entity Mux4_4_1 is
   port(
      Q  : in  std_logic_vector(1 downto 0); -- Selector
      H3 : in  std_logic_vector(3 downto 0); -- Entrada 3
      H2 : in  std_logic_vector(3 downto 0); -- Entrada 2
      H1 : in  std_logic_vector(3 downto 0); -- Entrada 1
      H0 : in  std_logic_vector(3 downto 0); -- Entrada 0
      Y  : out std_logic_vector(3 downto 0)  -- Salida
      );
   end Mux4_4_1;

architecture Seleccion of Mux4_4_1 is
begin
   process(H3,H2,H1,H0,Q)
   begin
      case Q is
         when "00"   => Y <= H0;
         when "01"   => Y <= H1;
         when "10"   => Y <= H2;
         when others => Y <= H3;
      end case;
   end process;
end Seleccion;
```

Multiplexor 4 - 1

```
library IEEE;
use IEEE.std_logic_1164.all;

entity Mux_4_1 is
   port(
      Q  : in  std_logic_vector(1 downto 0); -- Selector
      P3 : in  std_logic;                    -- Entrada 3
      P2 : in  std_logic;                    -- Entrada 2
      P1 : in  std_logic;                    -- Entrada 1
      P0 : in  std_logic;                    -- Entrada 0
      PD : out std_logic                     -- Salida
      );
   end Mux_4_1;

architecture Seleccion of Mux_4_1 is
begin
   process(P3,P2,P1,P0,Q)
   begin
      case Q is
         when "00"   => PD <= NOT(P0);
         when "01"   => PD <= NOT(P1);
         when "10"   => PD <= NOT(P2);
         when others => PD <= NOT(P3);
      end case;
   end process;
end Seleccion;
```

Decodificador de hexadecimal a 7 segmentos

```
library IEEE;
use IEEE.std_logic_1164.all;

entity Hexadecimal_7s is
   port(
      H : in  std_logic_vector(3 downto 0); -- Hexadecimal
      S : out std_logic_vector(7 downto 1)  -- 7 segmentos
      );
   end Hexadecimal_7s;
```

```
architecture Seleccion of Hexadecimal_7s is
begin
   process(H)
   begin
      case H is
         -- Segmento          abcdefg
         when "0000" => S <= "0000001";
         when "0001" => S <= "1001111";
         when "0010" => S <= "0010010";
         when "0011" => S <= "0000110";
         when "0100" => S <= "1001100";
         when "0101" => S <= "0100100";
         when "0110" => S <= "0100000";
         when "0111" => S <= "0001111";
         when "1000" => S <= "0000000";
         when "1001" => S <= "0000100";
         when "1010" => S <= "0001000";
         when "1011" => S <= "1100000";
         when "1100" => S <= "0110001";
         when "1101" => S <= "1000010";
         when "1110" => S <= "0110000";
         when others => S <= "0111000";
      end case;
   end process;
end Seleccion;
```

Control completo

```
library IEEE;
use IEEE.std_logic_1164.all;

entity Control_Exhibidores_7s is
   port(
      RST : in  std_logic;                       -- Reset maestro
      CLK : in  std_logic;                       -- Reloj maestro
      H3  : in  std_logic_vector(3 downto 0); -- Entrada 3
      H2  : in  std_logic_vector(3 downto 0); -- Entrada 2
      H1  : in  std_logic_vector(3 downto 0); -- Entrada 1
      H0  : in  std_logic_vector(3 downto 0); -- Entrada 0
      P   : in  std_logic_vector(3 downto 0); -- Entrada puntos
      A   : out std_logic_vector(3 downto 0); -- Anodos
      S   : out std_logic_vector(7 downto 1); -- 7 segmentos
      PD  : out std_logic                        -- Punto decimal
      );
   end Control_Exhibidores_7s;

architecture Completa of Control_Exhibidores_7s is
```

```
-- Base de tiempo de 5 kHz
component Modulo_10000
   port(
      RST : in  std_logic; -- Reset maestro
      CLK : in  std_logic; -- Reloj maestro
      Q   : out std_logic  -- Cero
      );
   end component;

-- Contador modulo 4
component Modulo_4
   port(
      RST : in  std_logic;                    -- Reset maestro
      CLK : in  std_logic;                    -- Reloj maestro
      H   : in  std_logic;                    -- Habilitacion de entrada
      Q   : out std_logic_vector(1 downto 0)  -- Salida
      );
   end component;

-- Demultiplexor 2 a 4
component Demux_2_4
   port(
      Q : in  std_logic_vector(1 downto 0); -- Anodo activo
      Y : out std_logic_vector(3 downto 0)  -- Anodos
      );
   end component;
```

```
-- Multiplexor cuadruple de 4 a 1
component Mux4_4_1
   port(
      Q  : in  std_logic_vector(1 downto 0); -- Selector
      H3 : in  std_logic_vector(3 downto 0); -- Entrada 3
      H2 : in  std_logic_vector(3 downto 0); -- Entrada 2
      H1 : in  std_logic_vector(3 downto 0); -- Entrada 1
      H0 : in  std_logic_vector(3 downto 0); -- Entrada 0
      Y  : out std_logic_vector(3 downto 0)  -- Salida
      );
   end component;

-- Multiplexor de 4 a 1
component Mux_4_1
   port(
      Q  : in  std_logic_vector(1 downto 0); -- Selector
      P3 : in  std_logic;                    -- Entrada 3
      P2 : in  std_logic;                    -- Entrada 2
      P1 : in  std_logic;                    -- Entrada 1
      P0 : in  std_logic;                    -- Entrada 0
      PD : out std_logic                     -- Salida
      );
   end component;

-- Decodificador de hexadecimal a 7 segmentos
component Hexadecimal_7s
   port(
      H : in  std_logic_vector(3 downto 0); -- Hexadecimal
      S : out std_logic_vector(7 downto 1)  -- 7 segmentos
      );
   end component;
```

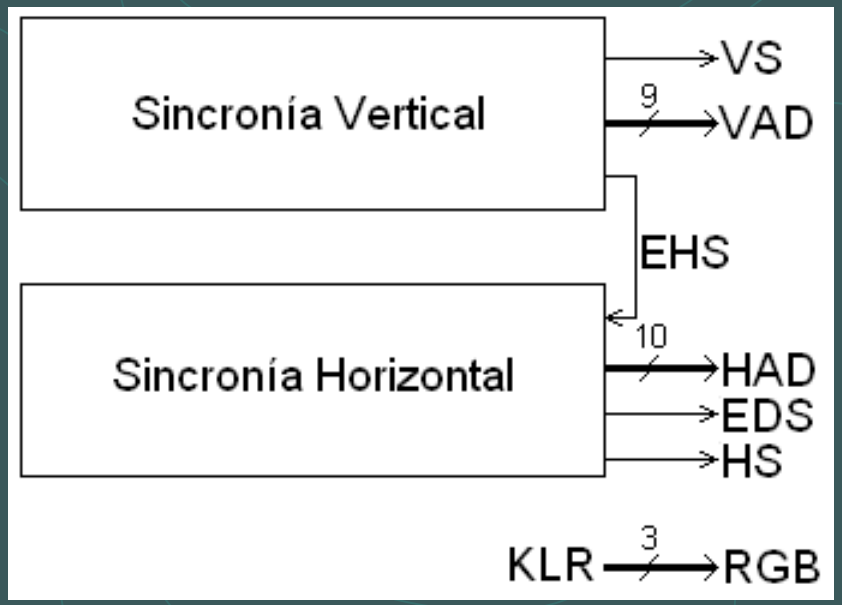
```
-- Habilitacion interna
signal ENI : std_logic;

-- Contador modulo 4
signal Q   : std_logic_vector(1 downto 0);

-- Multiplexor de los segmentos
signal M   : std_logic_vector(3 downto 0);

begin

   -- Base de tiempo
   Bloque_01: Modulo_10000   port map(RST,CLK,ENI);

   -- Contador modulo 4
   Bloque_02: Modulo_4       port map(RST,CLK,ENI,Q);

   -- Demultiplexor 2 a 4
   Bloque_03: Demux_2_4      port map(Q,A);

   -- Multiplexor de los segmentos
   Bloque_04: Mux4_4_1       port map(Q,H3,H2,H1,H0,M);

   -- Multiplexor del punto decimal
   Bloque_05: Mux_4_1        port map(Q,P(3),P(2),P(1),P(0),PD);

   -- Decodificador de hexadecimal a 7 segmentos
   Bloque_06: Hexadecimal_7s port map(M,S);

end Completa;
```

Ejemplo de síntesis

```
library IEEE;
use IEEE.std_logic_1164.all;

entity Ejemplo_Exhibidores is
   port(
      RST : in  std_logic;                       -- Reset maestro
      CLK : in  std_logic;                       -- Reloj maestro
      H1  : in  std_logic_vector(3 downto 0); -- Entrada 1
      H0  : in  std_logic_vector(3 downto 0); -- Entrada 0
      P   : in  std_logic;                       -- Entrada punto
      A   : out std_logic_vector(3 downto 0); -- Anodos
      S   : out std_logic_vector(7 downto 1); -- 7 segmentos
      PD  : out std_logic                        -- Punto decimal
      );
   end Ejemplo_Exhibidores;

architecture Muestra of Ejemplo_Exhibidores is
```

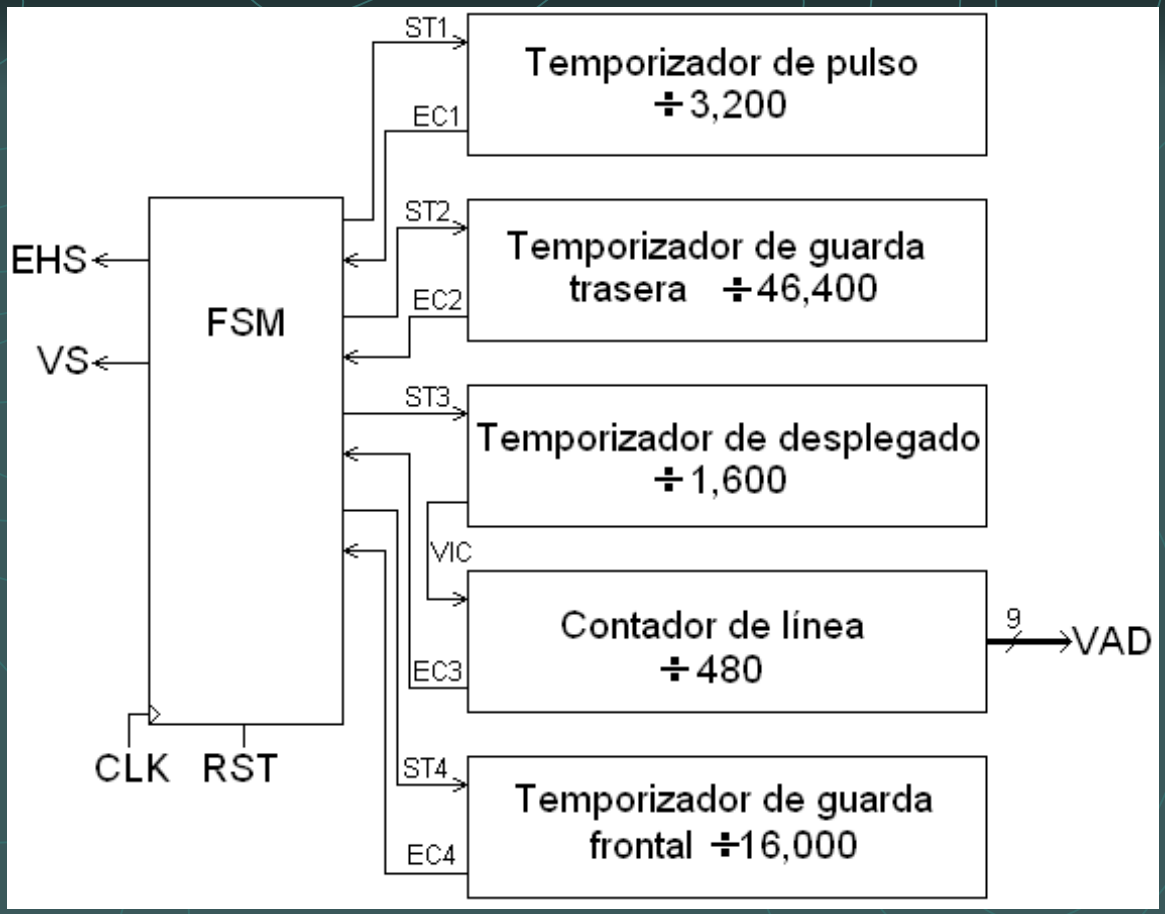
```
-- Control de los exhibidores de 7 segmentos
component Control_Exhibidores_7s
   port(
      RST : in  std_logic;                     -- Reset maestro
      CLK : in  std_logic;                     -- Reloj maestro
      H3  : in  std_logic_vector(3 downto 0); -- Entrada 3
      H2  : in  std_logic_vector(3 downto 0); -- Entrada 2
      H1  : in  std_logic_vector(3 downto 0); -- Entrada 1
      H0  : in  std_logic_vector(3 downto 0); -- Entrada 0
      P   : in  std_logic_vector(3 downto 0); -- Entrada puntos
      A   : out std_logic_vector(3 downto 0); -- Anodos
      S   : out std_logic_vector(7 downto 1); -- 7 segmentos
      PD  : out std_logic                      -- Punto decimal
      );
   end component;

-- Vector de puntos decimales
signal D      : std_logic_vector(3 downto 0);

-- Complemento de dos exhibidores
signal H3, H2 : std_logic_vector(3 downto 0);

begin

   -- Interconectividad
   D  <= P & P & P & P;
   H3 <= NOT(H1);
   H2 <= NOT(H0);

   -- Control de los exhibidores de 7 segmentos
   DUT_01: Control_Exhibidores_7s port map(RST,CLK,H3,H2,H1,H0,D,A,S,PD);

end Muestra;
```

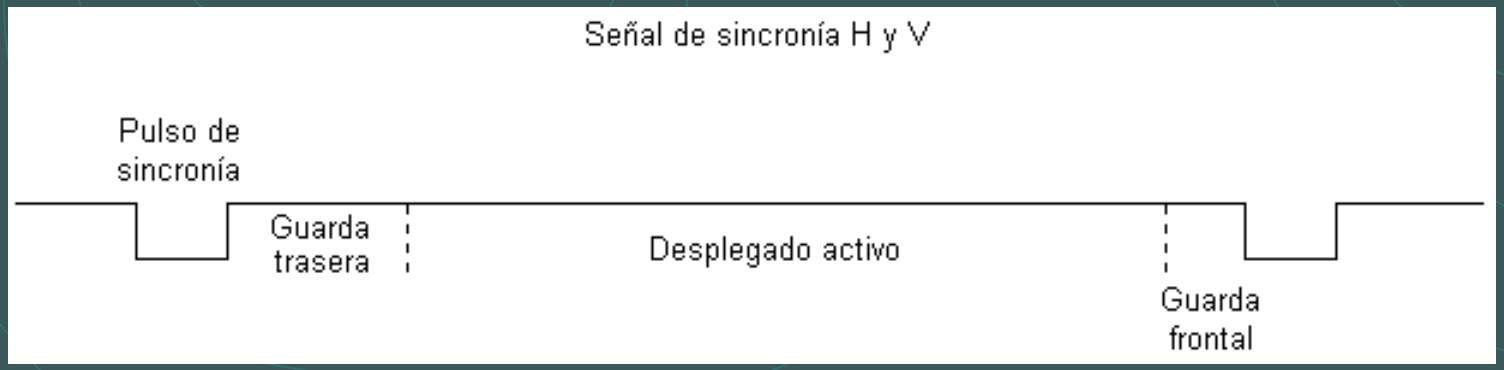
Desplegado VGA 1
Pantalla del color seleccionado



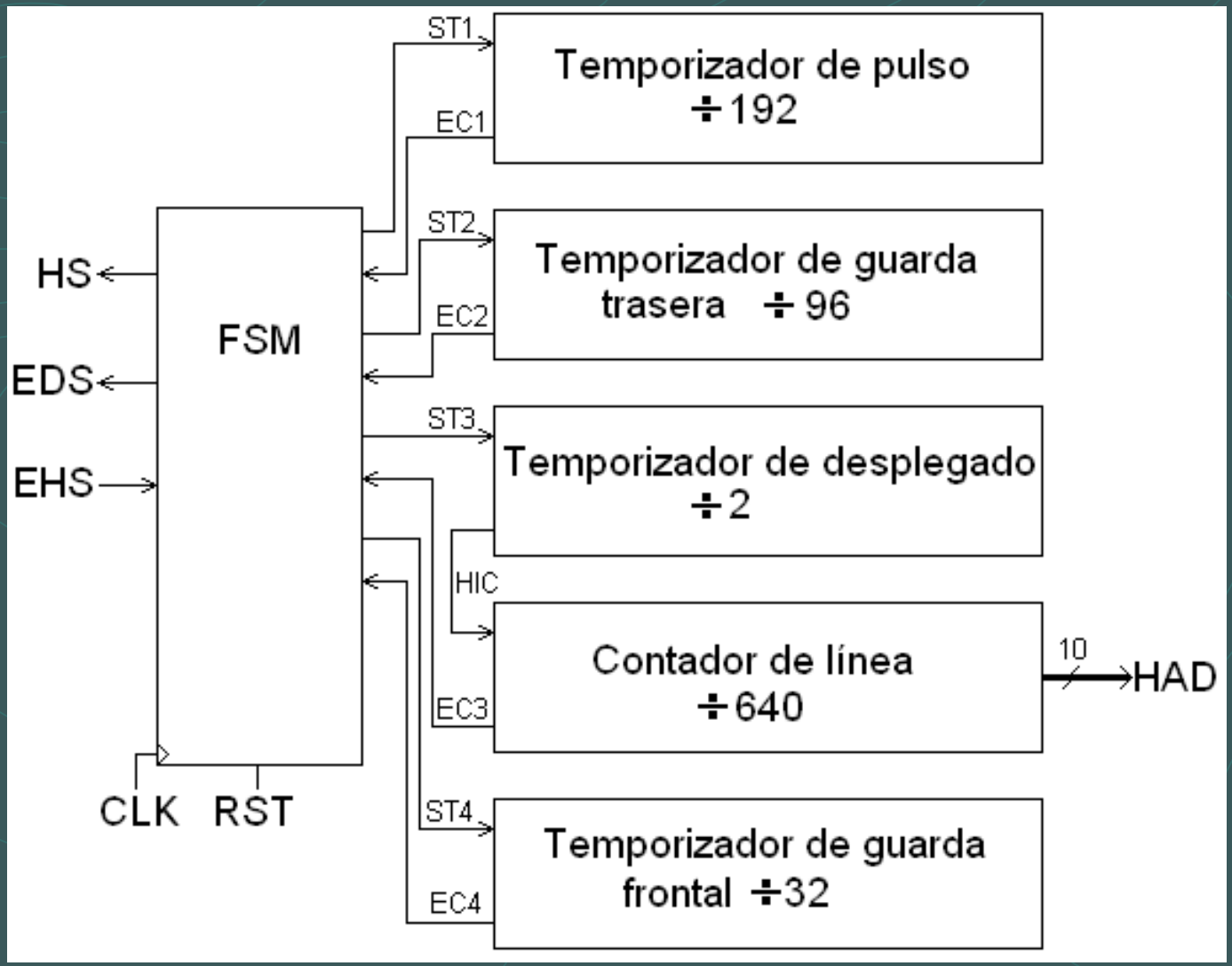
- VS Sincronía vertical
- VAD Dirección del renglón (9 bits, 480 líneas)
- EHS Habilita sincronía horizontal
- HAD Dirección de la columna (10 bits, 640 puntos)
- EDS Habilita color
- KLR Control de color
- RGB Red-Green-Blue

Sincronía vertical





Sincronía horizontal



Temporizadores simples

- Vertical
  - Pulso
  - Guarda trasera
  - Desplegado
  - Guarda frontal
- Horizontal
  - Pulso
  - Guarda trasera
  - Guarda frontal

Temporizador de pulso vertical

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity Pulso_V is
   port(
      RST : in  std_logic; -- Reset maestro
      CLK : in  std_logic; -- Reloj maestro
      ST  : in  std_logic; -- Habilitacion de entrada
      EC  : out std_logic  -- Fin de cuenta
      );
   end Pulso_V;

architecture Cascada of Pulso_V is

-- Estados internos
signal Qp, Qn : std_logic_vector(11 downto 0);

-- Detector de 3199
signal Z      : std_logic;

-- Incrementador
signal I      : std_logic_vector(11 downto 0);
```

```
begin

   Detector: process(Qp)
   begin
      if (Qp="110001111111") then
         Z <= '0';
      else
         Z <= '1';
      end if;
   end process Detector;

   Incrementador: process(Qp)
   begin
      I <= Qp + 1;
   end process Incrementador;

   Anulador: process(Z,I,ST)
   begin
      for j in 0 to 11 loop
         Qn(j) <= ST AND Z AND I(j);
      end loop;
      EC <= NOT(Z);
   end process Anulador;

   Secuencial: process(RST,CLK)
   begin
      if (RST='1') then
         Qp <= (others => '0');
      elsif (CLK'event and CLK='1') then
         Qp <= Qn;
      end if;
   end process Secuencial;

end Cascada;
```

Temporizador de desplegado horizontal

```
library IEEE;
use IEEE.std_logic_1164.all;

entity Desplegado_H is
   port(
      RST : in  std_logic; -- Reset maestro
      CLK : in  std_logic; -- Reloj maestro
      ST  : in  std_logic; -- Habilitacion de entrada
      HIC : out std_logic  -- Habilitacion de linea
      );
   end Desplegado_H;

architecture Cascada of Desplegado_H is

-- Estados internos
signal Qp, Qn : std_logic;

-- Detector de 1
signal Z      : std_logic;

-- Incrementador
signal I      : std_logic;
```

```
begin

   Detector: process(Qp)
   begin
      if (Qp='1') then
         Z <= '0';
      else
         Z <= '1';
      end if;
   end process Detector;

   Incrementador: process(Qp)
   begin
      I <= NOT(Qp);
   end process Incrementador;

   Anulador: process(Z,I,ST)
   begin
      Qn  <= ST AND Z AND I;
      HIC <= NOT(Z) AND ST;
   end process Anulador;

   Secuencial: process(RST,CLK)
   begin
      if (RST='1') then
         Qp <= '0';
      elsif (CLK'event and CLK='1') then
         Qp <= Qn;
      end if;
   end process Secuencial;

end Cascada;
```

Contador de línea vertical

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity Contador_V is
   port(
      RST : in  std_logic;                       -- Reset maestro
      CLK : in  std_logic;                       -- Reloj maestro
      VIC : in  std_logic;                       -- Habilitacion de entrada
      VAD : out std_logic_vector(8 downto 0); -- Direccion vertical
      EC  : out std_logic                        -- Fin de cuenta
      );
   end Contador_V;

architecture Cascada of Contador_V is

-- Estados internos
signal Qp, Qn : std_logic_vector(8 downto 0);

-- Detector de 479
signal Z      : std_logic;
```

```
-- Incrementador
signal I      : std_logic_vector(8 downto 0);

-- Anulador
signal A      : std_logic_vector(8 downto 0);

begin

   Detector: process(Qp)
   begin
      if (Qp="111011111") then
         Z <= '0';
      else
         Z <= '1';
      end if;
   end process Detector;

   Incrementador: process(Qp)
   begin
      I   <= Qp + 1;
      VAD <= Qp;
   end process Incrementador;

   Anulador: process(Z,I,VIC)
   begin
      for j in 0 to 8 loop
         A(j) <= Z AND I(j);
      end loop;
      EC <= NOT(Z) AND VIC;
   end process Anulador;
```

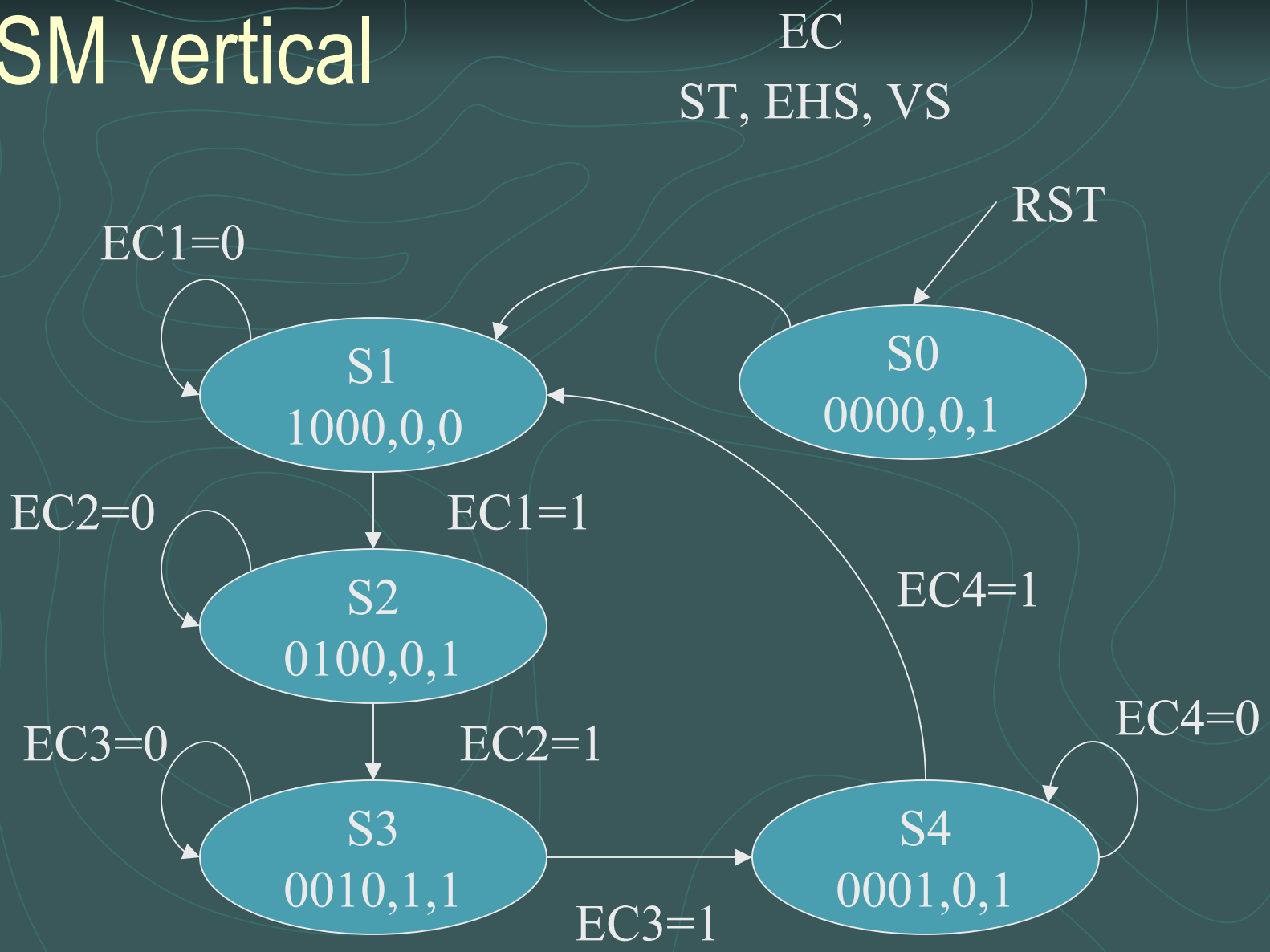
```
   Habilitacion: process(Qp,VIC,A)
   begin
          if (VIC='1') then
             Qn <= A;
          else
             Qn <= Qp;
          end if;
   end process Habilitacion;

   Secuencial: process(RST,CLK)
   begin
      if (RST='1') then
         Qp <= (others => '0');
      elsif (CLK'event and CLK='1') then
         Qp <= Qn;
      end if;
   end process Secuencial;

end Cascada;
```

Contador de línea horizontal

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity Contador_H is
   port(
      RST : in  std_logic;                      -- Reset maestro
      CLK : in  std_logic;                      -- Reloj maestro
      HIC : in  std_logic;                      -- Habilitacion de entrada
      HAD : out std_logic_vector(9 downto 0); -- Direccion horizontal
      EC  : out std_logic                       -- Fin de cuenta
      );
   end Contador_H;

architecture Cascada of Contador_H is

-- Estados internos
signal Qp, Qn : std_logic_vector(9 downto 0);

-- Detector de 639
signal Z      : std_logic;
```

```
-- Incrementador
signal I      : std_logic_vector(9 downto 0);

-- Anulador
signal A      : std_logic_vector(9 downto 0);

begin

   Detector: process(Qp)
   begin
      if (Qp="1001111111") then
         Z <= '0';
      else
         Z <= '1';
      end if;
   end process Detector;

   Incrementador: process(Qp)
   begin
      I   <= Qp + 1;
      HAD <= Qp;
   end process Incrementador;

   Anulador: process(Z,I,HIC)
   begin
      for j in 0 to 9 loop
         A(j) <= Z AND I(j);
      end loop;
      EC <= NOT(Z) AND HIC;
   end process Anulador;

   Habilitacion: process(Qp,HIC,A)
   begin
      if (HIC='1') then
         Qn <= A;
      else
         Qn <= Qp;
      end if;
   end process Habilitacion;

   Secuencial: process(RST,CLK)
   begin
      if (RST='1') then
         Qp <= (others => '0');
      elsif (CLK'event and CLK='1') then
         Qp <= Qn;
      end if;
   end process Secuencial;

end Cascada;
```

FSM vertical



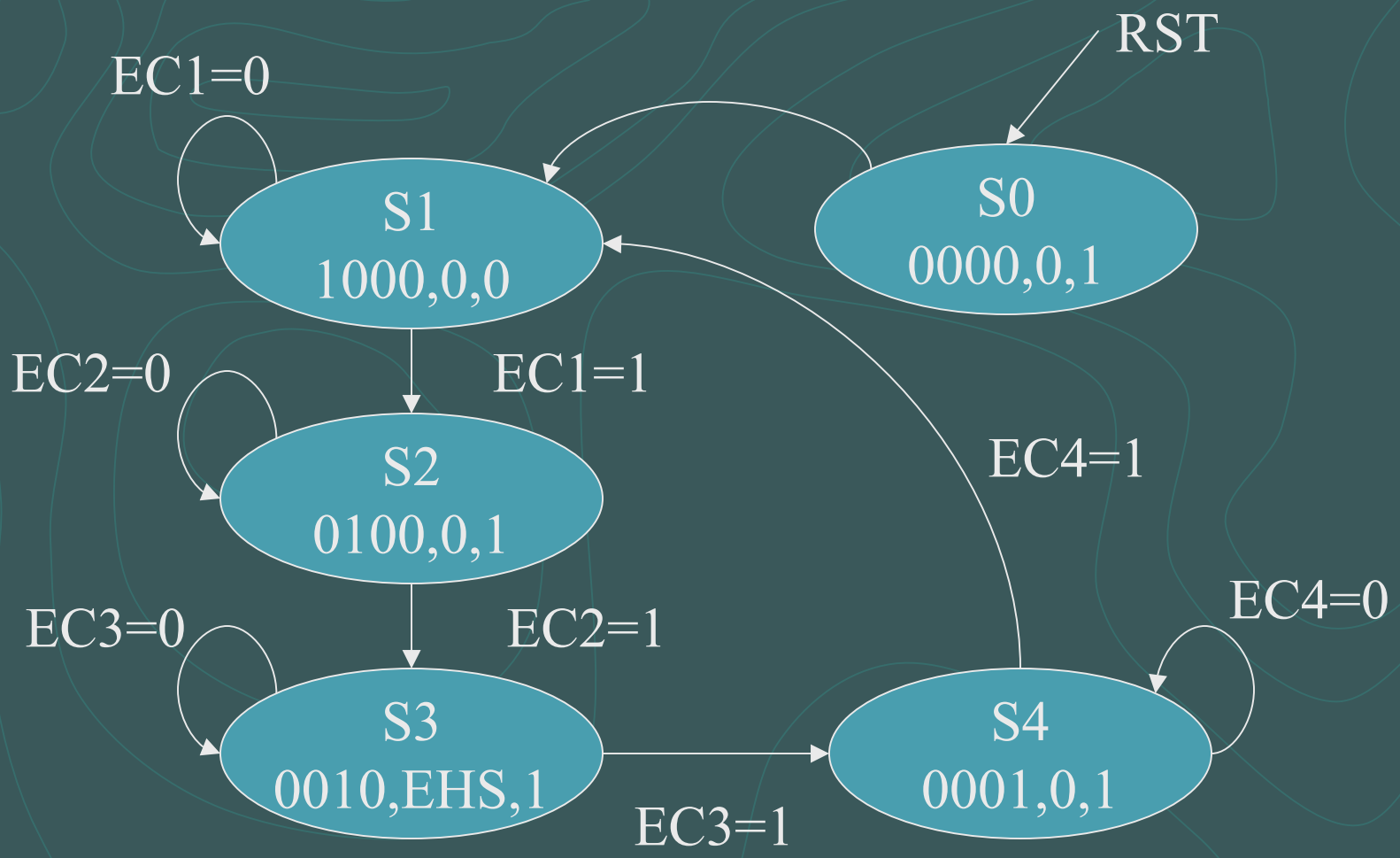
```
library IEEE;
use IEEE.std_logic_1164.all;

entity FSM_Vertical is
   port(
      RST : in  std_logic;                    -- Reset maestro
      CLK : in  std_logic;                    -- Reloj maestro
      EC  : in  std_logic_vector(1 to 4); -- Fin de cuenta
      ST  : out std_logic_vector(1 to 4); -- Inicio de cuenta
      EHS : out std_logic;                    -- Habilitacion horizontal
      VS  : out std_logic                     -- Sincronia vertical
      );
   end FSM_Vertical;

architecture Cascada of FSM_Vertical is

-- Estados internos
signal Qp, Qn : std_logic_vector(2 downto 0);

begin

   Combinacional: process(EC,Qp)
   begin
      case Qp is
         when "000"  =>
            Qn  <= "001";
            ST  <= "0000";
            EHS <= '0';
            VS  <= '1';
```

```
            when "001"  =>
               if (EC(1)='1') then
                  Qn <= "010";
               else
                  Qn <= Qp;
               end if;
               ST  <= "1000";
               EHS <= '0';
               VS  <= '0';
            when "010"  =>
               if (EC(2)='1') then
                  Qn <= "011";
               else
                  Qn <= Qp;
               end if;
               ST  <= "0100";
               EHS <= '0';
               VS  <= '1';
            when "011"  =>
               if (EC(3)='1') then
                  Qn <= "100";
               else
                  Qn <= Qp;
               end if;
               ST  <= "0010";
               EHS <= '1';
               VS  <= '1';
```

```
            when "100"  =>
               if (EC(4)='1') then
                  Qn <= "001";
               else
                  Qn <= Qp;
               end if;
               ST  <= "0001";
               EHS <= '0';
               VS  <= '1';
            when others =>
               Qn  <= "000";
               ST  <= "0000";
               EHS <= '0';
               VS  <= '1';
         end case;
      end process Combinacional;

      Secuencial: process(RST,CLK)
      begin
         if (RST='1') then
            Qp <= (others => '0');
         elsif (CLK'event and CLK='1') then
            Qp <= Qn;
         end if;
      end process Secuencial;

end Cascada;
```

FSM horizontal

EC, EHS
ST, EDS, HS



```
library IEEE;
use IEEE.std_logic_1164.all;

entity FSM_Horizontal is
   port(
      RST : in  std_logic;                    -- Reset maestro
      CLK : in  std_logic;                    -- Reloj maestro
      EHS : in  std_logic;                    -- Habilitacion horizontal
      EC  : in  std_logic_vector(1 to 4); -- Fin de cuenta
      ST  : out std_logic_vector(1 to 4); -- Inicio de cuenta
      EDS : out std_logic;                    -- Habilitacion de pixel
      HS  : out std_logic                     -- Sincronia horizontal
      );
   end FSM_Horizontal;

architecture Cascada of FSM_Horizontal is

-- Estados internos
signal Qp, Qn : std_logic_vector(2 downto 0);

begin

   Combinacional: process(EHS,EC,Qp)
   begin
      case Qp is
         when "000"  =>
            Qn  <= "001";
            ST  <= "0000";
            EDS <= '0';
            HS  <= '1';
```

```
            when "001"  =>
               if (EC(1)='1') then
                  Qn <= "010";
               else
                  Qn <= Qp;
               end if;
               ST  <= "1000";
               EDS <= '0';
               HS  <= '0';
            when "010"  =>
               if (EC(2)='1') then
                  Qn <= "011";
               else
                  Qn <= Qp;
               end if;
               ST  <= "0100";
               EDS <= '0';
               HS  <= '1';
            when "011"  =>
               if (EC(3)='1') then
                  Qn <= "100";
               else
                  Qn <= Qp;
               end if;
               ST  <= "0010";
               EDS <= EHS;
               HS  <= '1';
```

```
            when "100"  =>
               if (EC(4)='1') then
                  Qn <= "001";
               else
                  Qn <= Qp;
               end if;
               ST  <= "0001";
               EDS <= '0';
               HS  <= '1';
            when others =>
               Qn  <= "000";
               ST  <= "0000";
               EDS <= '0';
               HS  <= '1';
         end case;
      end process Combinacional;

      Secuencial: process(RST,CLK)
      begin
         if (RST='1') then
            Qp <= (others => '0');
         elsif (CLK'event and CLK='1') then
            Qp <= Qn;
         end if;
      end process Secuencial;

end Cascada;
```

Sincronizador vertical

```
library IEEE;
use IEEE.std_logic_1164.all;

entity Sincronia_Vertical is
   port(
      RST : in  std_logic;                          -- Reset maestro
      CLK : in  std_logic;                          -- Reloj maestro
      EHS : out std_logic;                          -- Habilitacion horizontal
      VAD : out std_logic_vector(8 downto 0); -- Direccion vertical
      VS  : out std_logic                           -- Sincronia vertical
      );
   end Sincronia_Vertical;

architecture Control of Sincronia_Vertical is

-- Contador de pulso vertical
component Pulso_V
   port(
      RST : in  std_logic; -- Reset maestro
      CLK : in  std_logic; -- Reloj maestro
      ST  : in  std_logic; -- Habilitacion de entrada
      EC  : out std_logic  -- Fin de cuenta
      );
   end component;
```

```
-- Contador de guarda trasera
component Trasero_V
   port(
      RST : in  std_logic; -- Reset maestro
      CLK : in  std_logic; -- Reloj maestro
      ST  : in  std_logic; -- Habilitacion de entrada
      EC  : out std_logic  -- Fin de cuenta
      );
   end component;

-- Contador de desplegado
component Desplegado_V
   port(
      RST : in  std_logic; -- Reset maestro
      CLK : in  std_logic; -- Reloj maestro
      ST  : in  std_logic; -- Habilitacion de entrada
      VIC : out std_logic  -- Habilitacion de linea
      );
   end component;

-- Contador de linea
component Contador_V
   port(
      RST : in  std_logic;                    -- Reset maestro
      CLK : in  std_logic;                    -- Reloj maestro
      VIC : in  std_logic;                    -- Habilitacion de entrada
      VAD : out std_logic_vector(8 downto 0); -- Direccion vertical
      EC  : out std_logic                     -- Fin de cuenta
      );
   end component;
```

```
-- Contador de guarda frontal
component Frontal_V
   port(
      RST : in  std_logic; -- Reset maestro
      CLK : in  std_logic; -- Reloj maestro
      ST  : in  std_logic; -- Habilitacion de entrada
      EC  : out std_logic  -- Fin de cuenta
      );
   end component;

-- Maquina de estados de control
component FSM_Vertical
   port(
      RST : in  std_logic;                    -- Reset maestro
      CLK : in  std_logic;                    -- Reloj maestro
      EC  : in  std_logic_vector(1 to 4); -- Fin de cuenta
      ST  : out std_logic_vector(1 to 4); -- Inicio de cuenta
      EHS : out std_logic;                    -- Habilitacion horizontal
      VS  : out std_logic                     -- Sincronia vertical
      );
   end component;

-- Habilitaciones internas
signal ST, EC : std_logic_vector(1 to 4);
signal VIC    : std_logic;
```

```
begin

   -- Contador de pulso vertical
   Bloque_01: Pulso_V      port map(RST,CLK,ST(1),EC(1));

   -- Contador de guarda trasera
   Bloque_02: Trasero_V    port map(RST,CLK,ST(2),EC(2));

   -- Contador de desplegado
   Bloque_03: Desplegado_V port map(RST,CLK,ST(3),VIC);

   -- Contador de linea
   Bloque_04: Contador_V   port map(RST,CLK,VIC,VAD,EC(3));

   -- Contador de guarda frontal
   Bloque_05: Frontal_V    port map(RST,CLK,ST(4),EC(4));

   -- Maquina de estados de control
   Bloque_06: FSM_Vertical port map(RST,CLK,EC,ST,EHS,VS);

end Control;
```

Sincronizador horizontal

```
library IEEE;
use IEEE.std_logic_1164.all;

entity Sincronia_Horizontal is
   port(
      RST : in  std_logic;                       -- Reset maestro
      CLK : in  std_logic;                       -- Reloj maestro
      EHS : in  std_logic;                       -- Habilitacion horizontal
      EDS : out std_logic;                       -- Habilitacion pixel
      HAD : out std_logic_vector(9 downto 0); -- Direccion horizontal
      HS  : out std_logic                        -- Sincronia horizontal
      );
   end Sincronia_Horizontal;

architecture Control of Sincronia_Horizontal is

-- Contador de pulso horizontal
component Pulso_H
   port(
      RST : in  std_logic; -- Reset maestro
      CLK : in  std_logic; -- Reloj maestro
      ST  : in  std_logic; -- Habilitacion de entrada
      EC  : out std_logic  -- Fin de cuenta
      );
   end component;
```

```
-- Contador de guarda trasera
component Trasero_H
   port(
      RST : in  std_logic; -- Reset maestro
      CLK : in  std_logic; -- Reloj maestro
      ST  : in  std_logic; -- Habilitacion de entrada
      EC  : out std_logic  -- Fin de cuenta
      );
   end component;

-- Contador de desplegado
component Desplegado_H
   port(
      RST : in  std_logic; -- Reset maestro
      CLK : in  std_logic; -- Reloj maestro
      ST  : in  std_logic; -- Habilitacion de entrada
      HIC : out std_logic  -- Habilitacion de linea
      );
   end component;

-- Contador de linea
component Contador_H
   port(
      RST : in  std_logic;                    -- Reset maestro
      CLK : in  std_logic;                    -- Reloj maestro
      HIC : in  std_logic;                    -- Habilitacion de entrada
      HAD : out std_logic_vector(9 downto 0); -- Direccion horizontal
      EC  : out std_logic                     -- Fin de cuenta
      );
   end component;
```

```
-- Contador de guarda frontal
component Frontal_H
   port(
      RST : in  std_logic; -- Reset maestro
      CLK : in  std_logic; -- Reloj maestro
      ST  : in  std_logic; -- Habilitacion de entrada
      EC  : out std_logic  -- Fin de cuenta
      );
   end component;

-- Maquina de estados de control
component FSM_Horizontal
   port(
      RST : in  std_logic;                    -- Reset maestro
      CLK : in  std_logic;                    -- Reloj maestro
      EHS : in  std_logic;                    -- Habilitacion horizontal
      EC  : in  std_logic_vector(1 to 4); -- Fin de cuenta
      ST  : out std_logic_vector(1 to 4); -- Inicio de cuenta
      EDS : out std_logic;                    -- Habilitacion de pixel
      HS  : out std_logic                     -- Sincronia horizontal
      );
   end component;

-- Habilitaciones internas
signal ST, EC : std_logic_vector(1 to 4);
signal HIC    : std_logic;
```

```
begin

   -- Contador de pulso horizontal
   Bloque_01: Pulso_H        port map(RST,CLK,ST(1),EC(1));

   -- Contador de guarda trasera
   Bloque_02: Trasero_H      port map(RST,CLK,ST(2),EC(2));

   -- Contador de desplegado
   Bloque_03: Desplegado_H   port map(RST,CLK,ST(3),HIC);

   -- Contador de linea
   Bloque_04: Contador_H     port map(RST,CLK,HIC,HAD,EC(3));

   -- Contador de guarda frontal
   Bloque_05: Frontal_H      port map(RST,CLK,ST(4),EC(4));

   -- Maquina de estados de control
   Bloque_06: FSM_horizontal port map(RST,CLK,EHS,EC,ST,EDS,HS);

end Control;
```

Sincronizador VGA

```
library IEEE;
use IEEE.std_logic_1164.all;

entity Sincronizador_VGA is
   port(
      RST : in  std_logic;                       -- Reset maestro
      CLK : in  std_logic;                       -- Reloj maestro
      VS  : out std_logic;                       -- Sincronia vertical
      HS  : out std_logic;                       -- Sincronia horizontal
      VAD : out std_logic_vector(8 downto 0); -- Direccion vertical
      HAD : out std_logic_vector(9 downto 0); -- Direccion horizontal
      EDS : out std_logic                        -- Habilitacion pixel
      );
   end Sincronizador_VGA;

architecture Control_Maestro of Sincronizador_VGA is

-- Sincronia vertical
component Sincronia_Vertical
   port(
      RST : in  std_logic;                       -- Reset maestro
      CLK : in  std_logic;                       -- Reloj maestro
      EHS : out std_logic;                       -- Habilitacion horizontal
      VAD : out std_logic_vector(8 downto 0); -- Direccion vertical
      VS  : out std_logic                        -- Sincronia vertical
      );
   end component;
```

```
-- Sincronia horizontal
component Sincronia_Horizontal
   port(
      RST : in  std_logic;                       -- Reset maestro
      CLK : in  std_logic;                       -- Reloj maestro
      EHS : in  std_logic;                       -- Habilitacion horizontal
      EDS : out std_logic;                       -- Habilitacion pixel
      HAD : out std_logic_vector(9 downto 0); -- Direccion horizontal
      HS  : out std_logic                        -- Sincronia horizontal
      );
   end component;

-- Habilitacion interna
signal EHS : std_logic;

begin

   -- Sincronia Vertical
   Control_01: Sincronia_Vertical   port map(RST,CLK,EHS,VAD,VS);

   -- Sincronia horizontal
   Control_02: Sincronia_horizontal port map(RST,CLK,EHS,EDS,HAD,HS);

end Control_Maestro;
```

Ejemplo 1
Desplegado de color

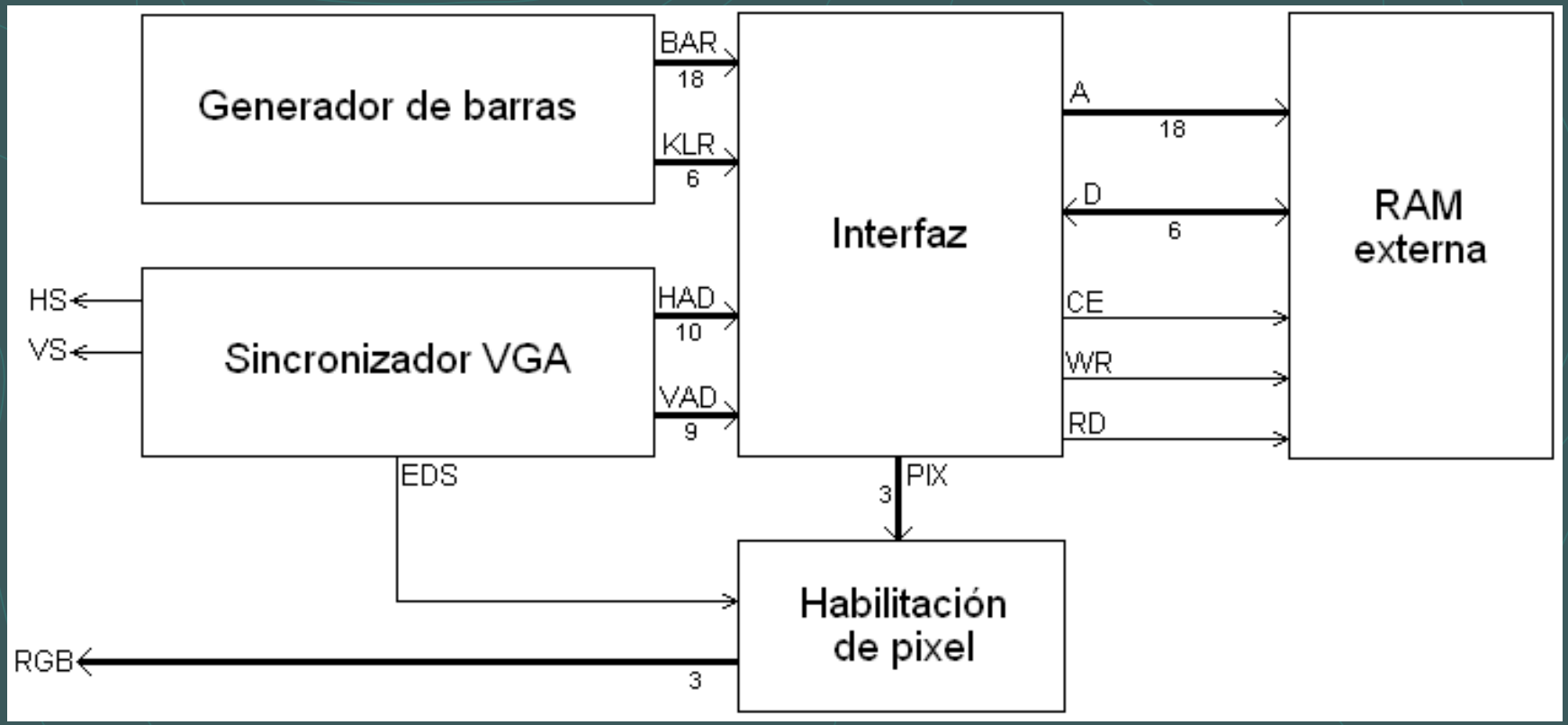
```
library IEEE;
use IEEE.std_logic_1164.all;

entity Ejemplo_VGA_1 is
   port(
      RST : in  std_logic;                     -- Reset maestro
      CLK : in  std_logic;                     -- Reloj maestro
      KLR : in  std_logic_vector(2 downto 0); -- Color VGA
      VS  : out std_logic;                     -- Sincronia vertical
      HS  : out std_logic;                     -- Sincronia horizontal
      R   : out std_logic;                     -- Red
      G   : out std_logic;                     -- Green
      B   : out std_logic                      -- Blue
      );
   end Ejemplo_VGA_1;

architecture Prueba of Ejemplo_VGA_1 is
```

```
-- Sincronizador VGA
component Sincronizador_VGA
   port(
      RST : in  std_logic;                          -- Reset maestro
      CLK : in  std_logic;                          -- Reloj maestro
      VS  : out std_logic;                          -- Sincronia vertical
      HS  : out std_logic;                          -- Sincronia horizontal
      VAD : out std_logic_vector(8 downto 0); -- Direccion vertical
      HAD : out std_logic_vector(9 downto 0); -- Direccion horizontal
      EDS : out std_logic                           -- Habilitacion pixel
      );
   end component;

-- Habilitacion de pixel
signal EDS: std_logic;

-- Direccion de pixel (no utilizadas en este ejemplo)
signal VAD: std_logic_vector(8 downto 0);
signal HAD: std_logic_vector(9 downto 0);

begin

   -- Sincronizador VGA
   DUT_01: Sincronizador_VGA port map(RST,CLK,VS,HS,VAD,HAD,EDS);

   -- Salida de pixel
   R <= KLR(2) AND EDS;
   G <= KLR(1) AND EDS;
   B <= KLR(0) AND EDS;

end Prueba;
```

Desplegado VGA 2
Imagen almacenada en RAM

- Utilizando el sistema de sincronía para desplegado VGA se desea generar una imagen simple en la memoria RAM externa y desde esta localidad desplegarla en el monitor VGA
- Una máquina de estados genera la imagen simple que consiste en barras de colores

Diagrama de bloques



Generador de barras

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity Genera_Barras is
   port(
      RST : in  std_logic;                        -- Reset maestro
      CLK : in  std_logic;                        -- Reloj maestro
      STG : in  std_logic_vector(1  downto 0); -- Control del generador
      EOG : out std_logic;                        -- Fin de generacion
      BAR : out std_logic_vector(17 downto 0); -- Direccion de la barra
      KLR : out std_logic_vector(5  downto 0)  -- Colores de la barra
      );
   end Genera_Barras;

architecture Contador of Genera_Barras is

-- Estados internos
signal Qp, Qn : std_logic_vector(17 downto 0);
```

```
begin

   Combinacional: process(STG,Qp)
   begin
      case STG is
         when "11"   => Qn <= (others => '0');
         when "10"   => Qn <= Qp + 1;
         when others => Qn <= Qp;
      end case;
      if (Qp="1111111111111111") then
         EOG <= '1';
      else
         EOG <= '0';
      end if;
      BAR <= Qp;
      KLR <= Qp(6 downto 4) & Qp(6 downto 4);
   end process Combinacional;

   Secuencial: process(RST,CLK)
   begin
      if (RST='1') then
         Qp <= (others => '0');
      elsif (CLK'event and CLK='1') then
         Qp <= Qn;
      end if;
   end process Secuencial;

end Contador;
```

Habilitación de pixel

```
library IEEE;
use IEEE.std_logic_1164.all;

entity Control_Pixel is
   port(
      EDS : in  std_logic;                     -- Habilitacion
      PIX : in  std_logic_vector(2 downto 0); -- Color
      R   : out std_logic;                     -- Red
      G   : out std_logic;                     -- Green
      B   : out std_logic                      -- Blue
      );
   end Control_Pixel;

architecture Simple of Control_Pixel is

begin

   R <= EDS AND PIX(2);
   G <= EDS AND PIX(1);
   B <= EDS AND PIX(0);

end Simple;
```

Interfaz

```
library IEEE;
use IEEE.std_logic_1164.all;

entity Interfaz is
   port(
      RST : in    std_logic;                       -- Reset maestro
      CLK : in    std_logic;                       -- Reloj maestro
      VAD : in    std_logic_vector(8  downto 0); -- Direccion vertical
      HAD : in    std_logic_vector(9  downto 0); -- Direccion horizontal
      PIX : out   std_logic_vector(2  downto 0); -- Pixel activo
      STG : out   std_logic_vector(1  downto 0); -- Control del generador
      EOG : in    std_logic;                       -- Fin de generacion
      BAR : in    std_logic_vector(17 downto 0); -- Direccion de la barra
      KLR : in    std_logic_vector(5  downto 0); -- Colores de la barra
      D   : inout std_logic_vector(7  downto 0); -- Datos RAM
      A   : out   std_logic_vector(17 downto 0); -- Direcciones RAM
      CE1 : out   std_logic;                       -- Chip enable 1
      CE2 : out   std_logic;                       -- Chip enable 2
      UB1 : out   std_logic;                       -- Upper byte enable 1
      UB2 : out   std_logic;                       -- Upper byte enable 2
      LB1 : out   std_logic;                       -- Lower byte enable 1
      LB2 : out   std_logic;                       -- Lower byte enable 2
      WR  : out   std_logic;                       -- Write enable
      OE  : out   std_logic                        -- Output enable
      );
   end Interfaz;
```

```
architecture Control of Interfaz is

-- Estados internos
signal Qp, Qn : std_logic_vector(2  downto 0);

-- Mux de direcciones
signal S      : std_logic;

begin

   FSM: process(Qp,EOG)
   begin
      CE2 <= '1';
      UB2 <= '1';
      UB1 <= '1';
      LB2 <= '1';
      LB1 <= '0';
      case Qp is
         when "000"  =>
            CE1 <= '1';
            WR  <= '1';
            OE  <= '1';
            S   <= '0';
            STG <= "11";
            Qn  <= "001";
         when "001"  =>
            CE1 <= '0';
            WR  <= '1';
            OE  <= '1';
            S   <= '0';
            STG <= "00";
            Qn  <= "010";
```

```
when "010"  =>
   CE1 <= '0';
   WR  <= '0';
   OE  <= '1';
   S   <= '0';
   STG <= "00";
   Qn  <= "011";
when "011"  =>
   CE1 <= '0';
   WR  <= '1';
   OE  <= '1';
   S   <= '0';
   STG <= "00";
   if (EOG='0') then
      Qn  <= "100";
   else
      Qn  <= "101";
   end if;
when "100"  =>
   CE1 <= '0';
   WR  <= '1';
   OE  <= '1';
   S   <= '0';
   STG <= "10";
   Qn  <= "001";
when "101"  =>
   CE1 <= '0';
   WR  <= '1';
   OE  <= '1';
   S   <= '1';
   STG <= "11";
   Qn  <= "110";
```

```
         when "110"  =>
            CE1 <= '0';
            WR  <= '1';
            OE  <= '0';
            S   <= '1';
            STG <= "11";
            Qn  <= "110";
         when others =>
            CE1 <= '1';
            WR  <= '1';
            OE  <= '1';
            S   <= '0';
            STG <= "11";
            Qn  <= "000";
      end case;
   end process FSM;

   Direccion: process(VAD,HAD,BAR,S)
   begin
      if (S='0') then
         A <= BAR;
      else
         A <= VAD & HAD(9 downto 1);
      end if;
   end process Direccion;
```

```
   Dato: process(D,KLR,S,HAD)
   begin
      if (S='0') then
         D <= "00" & KLR;
      else
         D <= (others => 'Z');
      end if;
      if (HAD(0)='0') then
         PIX <= D(5 downto 3);
      else
         PIX <= D(2 downto 0);
      end if;
   end process Dato;

   Reloj: process(RST,CLK)
   begin
      if (RST='1') then
         Qp <= (others => '0');
      elsif (CLK'event and CLK='1') then
         Qp <= Qn;
      end if;
   end process Reloj;

end Control;
```

Ejemplo de desplegado VGA
Generador de barras de colores

```
library IEEE;
use IEEE.std_logic_1164.all;

entity Ejemplo_VGA_2 is
   port(
      RST : in    std_logic;                      -- Reset maestro
      CLK : in    std_logic;                      -- Reloj maestro
      D   : inout std_logic_vector(7  downto 0); -- Datos RAM
      A   : out   std_logic_vector(17 downto 0); -- Direcciones RAM
      CE1 : out   std_logic;                      -- Chip enable 1
      CE2 : out   std_logic;                      -- Chip enable 2
      UB1 : out   std_logic;                      -- Upper byte enable 1
      UB2 : out   std_logic;                      -- Upper byte enable 2
      LB1 : out   std_logic;                      -- Lower byte enable 1
      LB2 : out   std_logic;                      -- Lower byte enable 2
      WR  : out   std_logic;                      -- Write enable
      OE  : out   std_logic;                      -- Output enable
      VS  : out   std_logic;                      -- Sincronia vertical
      HS  : out   std_logic;                      -- Sincronia horizontal
      R   : out   std_logic;                      -- Red
      G   : out   std_logic;                      -- Green
      B   : out   std_logic                       -- Blue
      );
   end Ejemplo_VGA_2;
```

```
architecture Prueba of Ejemplo_VGA_2 is

-- Sincronizador VGA
component Sincronizador_VGA
   port(
      RST : in  std_logic;                     -- Reset maestro
      CLK : in  std_logic;                     -- Reloj maestro
      VS  : out std_logic;                     -- Sincronia vertical
      HS  : out std_logic;                     -- Sincronia horizontal
      VAD : out std_logic_vector(8 downto 0); -- Direccion vertical
      HAD : out std_logic_vector(9 downto 0); -- Direccion horizontal
      EDS : out std_logic                      -- Habilitacion pixel
      );
   end component;

-- Habilitacion de pixel
component Control_Pixel
   port(
      EDS : in  std_logic;                     -- Habilitacion
      PIX : in  std_logic_vector(2 downto 0); -- Color
      R   : out std_logic;                     -- Red
      G   : out std_logic;                     -- Green
      B   : out std_logic                      -- Blue
      );
   end component;

-- Generador de barras
component Genera_Barras
   port(
      RST : in  std_logic;                      -- Reset maestro
      CLK : in  std_logic;                      -- Reloj maestro
      STG : in  std_logic_vector(1  downto 0); -- Control del generador
      EOG : out std_logic;                      -- Fin de generacion
      BAR : out std_logic_vector(17 downto 0); -- Direccion de la barra
      KLR : out std_logic_vector(5  downto 0)  -- Colores de la barra
      );
   end component;
```

```
-- Interfaz
component Interfaz
   port(
      RST : in    std_logic;                      -- Reset maestro
      CLK : in    std_logic;                      -- Reloj maestro
      VAD : in    std_logic_vector(8  downto 0); -- Direccion vertical
      HAD : in    std_logic_vector(9  downto 0); -- Direccion horizontal
      PIX : out   std_logic_vector(2  downto 0); -- Pixel activo
      STG : out   std_logic_vector(1  downto 0); -- Control del generador
      EOG : in    std_logic;                      -- Fin de generacion
      BAR : in    std_logic_vector(17 downto 0); -- Direccion de la barra
      KLR : in    std_logic_vector(5  downto 0); -- Colores de la barra
      D   : inout std_logic_vector(7  downto 0); -- Datos RAM
      A   : out   std_logic_vector(17 downto 0); -- Direcciones RAM
      CE1 : out   std_logic;                      -- Chip enable 1
      CE2 : out   std_logic;                      -- Chip enable 2
      UB1 : out   std_logic;                      -- Upper byte enable 1
      UB2 : out   std_logic;                      -- Upper byte enable 2
      LB1 : out   std_logic;                      -- Lower byte enable 1
      LB2 : out   std_logic;                      -- Lower byte enable 2
      WR  : out   std_logic;                      -- Write enable
      OE  : out   std_logic                       -- Output enable
      );
   end component;

-- Sincronizador VGA
signal VAD: std_logic_vector(8 downto 0);
signal HAD: std_logic_vector(9 downto 0);
signal EDS: std_logic;
```

```
-- Habilitacion de pixel
signal PIX: std_logic_vector(2 downto 0);

-- Generador de barras
signal STG: std_logic_vector(1  downto 0);
signal EOG: std_logic;
signal BAR: std_logic_vector(17 downto 0);
signal KLR: std_logic_vector(5  downto 0);


begin

   -- Sincronizador VGA
   DUT_01: Sincronizador_VGA port map(RST,CLK,VS,HS,VAD,HAD,EDS);

   -- Habilitacion de pixel
   DUT_02: Control_Pixel     port map(EDS,PIX,R,G,B);

   -- Generador de barras
   DUT_03: Genera_Barras     port map(RST,CLK,STG,EOG,BAR,KLR);

   -- Interfaz
   DUT_04: Interfaz          port map(RST,CLK,VAD,HAD,PIX,STG,EOG,BAR,KLR,
                                      D,A,CE1,CE2,UB1,UB2,LB1,LB2,WR,OE);

end Prueba;
```

Conclusiones

- Los modelos de interfaz consisten en máquinas de estados finitos que generan patrones de señales de acuerdo con el dispositivo periférico
- Estos sistemas de interfaz pueden ser modelados como un circuito independiente de otros elementos que constituyen la aplicación general

Notas finales

- Los ejemplos mostrados en la presente sección son completamente funcionales y pueden ser exportados a cualquier otra plataforma de desarrollo en futuras aplicaciones
- El desplegado VGA es una de las herramientas más útiles en el desarrollo de sistemas digitales en cómputo reconfigurable con aplicaciones en procesamiento de imágenes

- Si se desea otro tipo de desplegado en pantalla de alta resolución, el modelo de interfaz VGA se puede utilizar como primitiva de diseño