Memoria rom

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 29.07.2020 08:31:39
-- Design Name:
-- Module Name: ROM - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------



library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use ieee.numeric_std.all ;

entity ROM is
        Port (
            CLK_ROM: IN STD_LOGIC; -- señal de reloj para mi ROM
            addres: IN STD_LOGIC_VECTOR(3 DOWNTO 0); -- Direccion de memoria de mi
MATRIX 16X8
```

```
SELECTION: IN INTEGER RANGE 0 TO 50; -- Seleccion de CARACTER
data: OUT STD_LOGIC_VECTOR(7 DOWNTO 0)-- datosde ENVIO x ADDRESS
);
```

--- Estas declaraciones son GLOBALES , wahtever architecture que use esta entidad reconocera estas constantes

```
-- Constantes para COMPARACION
CONSTANT  option_a: INTEGER :=1; -- A
CONSTANT  option_b: INTEGER :=2;
CONSTANT  option_c: INTEGER :=3;
CONSTANT  option_d: INTEGER :=4;
CONSTANT  option_e: INTEGER :=5;
CONSTANT  option_f: INTEGER :=6;
CONSTANT  option_g: INTEGER :=7;
CONSTANT  option_h: INTEGER :=8;
CONSTANT  option_i: INTEGER :=9;
CONSTANT  option_j: INTEGER :=10;
CONSTANT  option_k: INTEGER :=11;
CONSTANT  option_l: INTEGER :=12;
CONSTANT  option_m: INTEGER :=13;
CONSTANT  option_n: INTEGER :=14;
CONSTANT  option_o: INTEGER :=15;
CONSTANT  option_p: INTEGER :=16;
CONSTANT  option_q: INTEGER :=17;
CONSTANT  option_r: INTEGER :=18;
CONSTANT  option_s: INTEGER :=19;
CONSTANT  option_t: INTEGER :=20;
CONSTANT  option_u: INTEGER :=21;
CONSTANT  option_v: INTEGER :=22;
CONSTANT  option_w: INTEGER :=23;
CONSTANT  option_x: INTEGER :=24;
```

```
CONSTANT  option_y: INTEGER :=25;

CONSTANT  option_z: INTEGER :=26; -- Z


-- **


end ROM;


architecture Behavioral of ROM is

-- 1RA PARTE

TYPE data_rom IS ARRAY(NATURAL RANGE <>)OF  STD_LOGIC_VECTOR(7 DOWNTO 0); -- OJO se define en orden DESCENDENTE MSB -> LSB servira para las columnas VGA



-- 2DA PART --- SOLO RECONOCIBLES EN ESTA ARQUITECTURA


CONSTANT  letter_a: data_rom :=(


   --   12345678

      "00000000", -- 0

                  "00000000", -- 1

                  "00010000", -- 2    *

                  "00111000", -- 3   ***

                  "01101100", -- 4  ** **

                  "11000110", -- 5 **   **

                  "11000110", -- 6 **   **

                  "11111110", -- 7 *******

                  "11000110", -- 8 **   **

                  "11000110", -- 9 **   **

                  "11000110", -- a **   **

                  "11000110", -- b **   **

                  "00000000", -- c
```

```
                    "00000000", -- d
                    "00000000", -- e
                    "00000000" -- f
);

--
CONSTANT  letter_b: data_rom :=(

      --   12345678
          "00000000", -- 0
          "00000000", -- 1
          "11111100", -- 2 ******
          "01100110", -- 3  **  **
          "01100110", -- 4  **  **
          "01100110", -- 5  **  **
          "01111100", -- 6  *****
          "01100110", -- 7  **  **
          "01100110", -- 8  **  **
          "01100110", -- 9  **  **
          "01100110", -- a  **  **
          "11111100", -- b ******
          "00000000", -- c
          "00000000", -- d
          "00000000", -- e
          "00000000" -- f
);
--
CONSTANT  letter_c: data_rom :=(

      --   12345678
          "00000000", -- 0
```

```
            "00000000", -- 1
            "00111100", -- 2   ****
            "01100110", -- 3  **  **
            "11000010", -- 4 **    *
            "11000000", -- 5 **
            "11000000", -- 6 **
            "11000000", -- 7 **
            "11000000", -- 8 **
            "11000010", -- 9 **    *
            "01100110", -- a  **  **
            "00111100", -- b   ****
            "00000000", -- c
            "00000000", -- d
            "00000000", -- e
            "00000000" -- f

);
-----D
CONSTANT  letter_d: data_rom :=(

            --   12345678
                "00000000", -- 0
                "00000000", -- 1
                "11111000", -- 2 *****
                "01101100", -- 3  ** **
                "01100110", -- 4  **  **
                "01100110", -- 5  **  **
                "01100110", -- 6  **  **
                "01100110", -- 7  **  **
                "01100110", -- 8  **  **
                "01100110", -- 9  **  **
```

```
                "01101100", -- a  ** **
                "11111000", -- b *****
                "00000000", -- c
                "00000000", -- d
                "00000000", -- e
                "00000000" -- f

);
------E

CONSTANT  letter_e: data_rom :=(

        --    12345678
                        "00000000", -- 0
            "00000000", -- 1
            "11111110", -- 2 *******
            "01100110", -- 3  **  **
            "01100010", -- 4  **   *
            "01101000", -- 5  ** *
            "01111000", -- 6  ****
            "01101000", -- 7  ** *
            "01100000", -- 8  **
            "01100010", -- 9  **   *
            "01100110", -- a  **  **
            "11111110", -- b *******
            "00000000", -- c
            "00000000", -- d
            "00000000", -- e
            "00000000" -- f

);
```

```
---F -----
CONSTANT  letter_f: data_rom :=(

   --   12345678
                    "00000000", -- 0
       "00000000", -- 1
       "11111110", -- 2 *******
       "01100110", -- 3  **  **
       "01100010", -- 4  **   *
       "01101000", -- 5  ** *
       "01111000", -- 6  ****
       "01101000", -- 7  ** *
       "01100000", -- 8  **
       "01100000", -- 9  **
       "01100000", -- a  **
       "11110000", -- b ****
       "00000000", -- c
       "00000000", -- d
       "00000000", -- e
       "00000000" -- f

);

----G
CONSTANT  letter_g: data_rom :=(

   --   12345678
                    "00000000", -- 0
       "00000000", -- 1
       "00111100", -- 2   ****
```

```
        "01100110", -- 3  **  **
        "11000010", -- 4 **    *
        "11000000", -- 5 **
        "11000000", -- 6 **
        "11011110", -- 7 ** ****
        "11000110", -- 8 **   **
        "11000110", -- 9 **   **
        "01100110", -- a  **  **
        "00111010", -- b   *** *
        "00000000", -- c
        "00000000", -- d
        "00000000", -- e
        "00000000" -- f


);

----H
CONSTANT  letter_h: data_rom :=(

--   12345678
    "00000000", -- 0
    "00000000", -- 1
    "11000110", -- 2 **   **
    "11000110", -- 3 **   **
    "11000110", -- 4 **   **
    "11000110", -- 5 **   **
    "11111110", -- 6 *******
    "11000110", -- 7 **   **
    "11000110", -- 8 **   **
    "11000110", -- 9 **   **
```

```
    "11000110", -- a **   **
    "11000110", -- b **   **
    "00000000", -- c
    "00000000", -- d
    "00000000", -- e
    "00000000" -- f

);



----K
CONSTANT  letter_k: data_rom :=(

--    12345678
          "00000000", -- 0
    "00000000", -- 1
    "11100110", -- 2 ***  **
    "01100110", -- 3  **  **
    "01100110", -- 4  **  **
    "01101100", -- 5  ** **
    "01111000", -- 6  ****
    "01111000", -- 7  ****
    "01101100", -- 8  ** **
    "01100110", -- 9  **  **
    "01100110", -- a  **  **
    "11100110", -- b ***  **
    "00000000", -- c
    "00000000", -- d
    "00000000", -- e
    "00000000" -- f
```

```
);

----G
CONSTANT  letter_l: data_rom :=(

--    12345678
            "00000000", -- 0
    "00000000", -- 1
    "11110000", -- 2 ****
    "01100000", -- 3  **
    "01100000", -- 4  **
    "01100000", -- 5  **
    "01100000", -- 6  **
    "01100000", -- 7  **
    "01100000", -- 8  **
    "01100010", -- 9  **   *
    "01100110", -- a  **  **
    "11111110", -- b *******
    "00000000", -- c
    "00000000", -- d
    "00000000", -- e
    "00000000" -- f

);

----O
CONSTANT  letter_o: data_rom :=(

--    12345678
            "00000000", -- 0
    "00000000", -- 1
```

```
    "01111100", -- 2  *****
    "11000110", -- 3 **   **
    "11000110", -- 4 **   **
    "11000110", -- 5 **   **
    "11000110", -- 6 **   **
    "11000110", -- 7 **   **
    "11000110", -- 8 **   **
    "11000110", -- 9 **   **
    "11000110", -- a **   **
    "01111100", -- b  *****
    "00000000", -- c
    "00000000", -- d
    "00000000", -- e
    "00000000" -- f

);

----P
CONSTANT  letter_p: data_rom :=(

--   12345678
          "00000000", -- 0
    "00000000", -- 1
    "11111100", -- 2 ******
    "01100110", -- 3  **  **
    "01100110", -- 4  **  **
    "01100110", -- 5  **  **
    "01111100", -- 6  *****
    "01100000", -- 7  **
    "01100000", -- 8  **
    "01100000", -- 9  **
```

```
    "01100000", -- a  **
    "11110000", -- b ****
    "00000000", -- c
    "00000000", -- d
    "00000000", -- e
    "00000000" -- f

);

----G
CONSTANT  letter_s: data_rom :=(

--   12345678
          "00000000", -- 0
    "00000000", -- 1
    "01111100", -- 2  *****
    "11000110", -- 3 **   **
    "11000110", -- 4 **   **
    "01100000", -- 5  **
    "00111000", -- 6   ***
    "00001100", -- 7     **
    "00000110", -- 8      **
    "11000110", -- 9 **   **
    "11000110", -- a **   **
    "01111100", -- b  *****
    "00000000", -- c
    "00000000", -- d
    "00000000", -- e
    "00000000" -- f

);
```

```
----T
CONSTANT  letter_t: data_rom :=(

--    12345678
            "00000000", -- 0
     "00000000", -- 1
     "11111111", -- 2 ********
     "11011011", -- 3 ** ** **
     "10011001", -- 4 *  **  *
     "00011000", -- 5    **
     "00011000", -- 6    **
     "00011000", -- 7    **
     "00011000", -- 8    **
     "00011000", -- 9    **
     "00011000", -- a    **
     "00111100", -- b   ****
     "00000000", -- c
     "00000000", -- d
     "00000000", -- e
     "00000000" -- f

);

----X
CONSTANT  letter_x: data_rom :=(

--    12345678
            "00000000", -- 0
     "00000000", -- 1
     "11000011", -- 2 **    **
```

```
    "11000011", -- 3 **    **
    "01100110", -- 4  **  **
    "00111100", -- 5   ****
    "00011000", -- 6    **
    "00011000", -- 7    **
    "00111100", -- 8   ****
    "01100110", -- 9  **  **
    "11000011", -- a **    **
    "11000011", -- b **    **
    "00000000", -- c
    "00000000", -- d
    "00000000", -- e
    "00000000" -- f

);

----Y
CONSTANT  letter_y: data_rom :=(

--   12345678
          "00000000", -- 0
    "00000000", -- 1
    "11000011", -- 2 **    **
    "11000011", -- 3 **    **
    "11000011", -- 4 **    **
    "01100110", -- 5  **  **
    "00111100", -- 6   ****
    "00011000", -- 7    **
    "00011000", -- 8    **
    "00011000", -- 9    **
    "00011000", -- a    **
```

```
    "00111100", -- b   ****
    "00000000", -- c
    "00000000", -- d
    "00000000", -- e
    "00000000" -- f

);
------nulll
CONSTANT  NULO: data_rom :=(

--    12345678
          "00000000", -- 0
    "00000000", -- 1
    "00000000", -- 2
    "00000000", -- 3
    "00000000", -- 4
    "00000000", -- 5
    "00000000", -- 6
    "00000000", -- 7
    "00000000", -- 8
    "00000000", -- 9
    "00000000", -- a
    "00000000", -- b
    "00000000", -- c
    "00000000", -- d
    "00000000", -- e
    "00000000" -- f

);
```

```
begin


PROCESS(CLK_ROM,addres,SELECTION) -- LISTA DE SENSIBLIDAD GENARLLY:IN

BEGIN

   IF(CLK_ROM='1' AND CLK_ROM'EVENT)THEN -- ESPERA A FLANCO DE SUBIDA PARA QUE
ENVIE SOLO EL VECTOR-ADDRESS

       CASE (selection) IS -- solo para selccionar la LETRA

          WHEN option_a => data<=letter_a(TO_INTEGER(UNSIGNED(addres))); -- eSTE
PROCEEDINGS enviar el vectotrs que corrresponde a un ADDRESS

          WHEN option_b => data<=letter_b(TO_INTEGER(UNSIGNED(addres)));

          WHEN option_c => data<=letter_c(TO_INTEGER(UNSIGNED(addres)));

          WHEN option_d => data<=letter_d(TO_INTEGER(UNSIGNED(addres)));

          WHEN option_e => data<=letter_e(TO_INTEGER(UNSIGNED(addres)));

          WHEN option_f => data<=letter_f(TO_INTEGER(UNSIGNED(addres)));

          WHEN option_g => data<=letter_g(TO_INTEGER(UNSIGNED(addres)));

          WHEN option_h => data<=letter_h(TO_INTEGER(UNSIGNED(addres)));

          WHEN option_k => data<=letter_k(TO_INTEGER(UNSIGNED(addres)));

          WHEN option_l => data<=letter_l(TO_INTEGER(UNSIGNED(addres)));

          WHEN option_o => data<=letter_o(TO_INTEGER(UNSIGNED(addres)));

          WHEN option_p => data<=letter_p(TO_INTEGER(UNSIGNED(addres)));

          WHEN option_s => data<=letter_s(TO_INTEGER(UNSIGNED(addres)));

          WHEN option_t => data<=letter_t(TO_INTEGER(UNSIGNED(addres)));

          WHEN option_x => data<=letter_x(TO_INTEGER(UNSIGNED(addres)));

          WHEN option_y => data<=letter_y(TO_INTEGER(UNSIGNED(addres)));

          WHEN OTHERS   => data<=NULO(TO_INTEGER(UNSIGNED(addres)));  ------CARACTER
NULO, CUANDO EL USUARIO NON HACE NINGUNA SEÑA----------:C whyyyy ! OTHERSSSSS!!

       END CASE;


   END IF;


END PROCESS;
```

end Behavioral;

modulo de pintado PINTAVGA

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 30.07.2020 11:13:19
-- Design Name:
-- Module Name: PINTA_VGA - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL; -- INCLUIMOS PARA QUE SE PUEDA USAR OPERACIONES ARITMETICAS EN STD_LOGIC TIPO DE DATOS Y OPERADORES MATEMATICOS
use ieee.numeric_std.all ; -- concvbersiond edatos TO_INTEGER


-- Uncomment the following library declaration if using
```

```
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity PINTA_VGA is
        Port (
            COL_P: IN STD_LOGIC_VECTOR(9 DOWNTO 0);  --ESQUINA SUPERIOR IZQUIERDA
            ROW_P: IN STD_LOGIC_VECTOR(9 DOWNTO 0); -- PARA  DIRECCION DE
MEMORIA DE ROM
            VISIBLE_P: IN STD_LOGIC; -- PORCH
            --
            data_img: IN STD_LOGIC_VECTOR(7 DOWNTO 0);   -- VECTOR DEVUELCVE LA
DIRTERCCION DE MEMORIA
            data_i2c: IN INTEGER RANGE 0 TO 50;       -- corazon de la bestia
            SELECTION: OUT INTEGER RANGE 0 TO 50;  --- PARA ELEGIR EL CARACTER DE LA
ROM
            address: OUT STD_LOGIC_VECTOR(3 DOWNTO 0); -- relacion ROW->ADDRESS
            -- COLOR
            R_P:OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
            G_P:OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
            B_P:OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
        );
end PINTA_VGA;

architecture Behavioral of PINTA_VGA is

-- PINTADO: FONDO -> AZUL
    --     LETRA -> ROJO
```

```
begin
PROCESS(data_i2c,data_img,COL_P,ROW_P,VISIBLE_P) -- LISTA DE SENSIBILIDAD

VARIABLE var_bit: STD_LOGIC:='0'; -- PARA ALMACENRA EL bit BIFURCADO
VARIABLE var_img: STD_LOGIC_VECTOR(7 DOWNTO 0); -- PARA ALMACENRA el vector

BEGIN

--- PART1: verificacion de DATA
    IF(data_i2c /= 0) THEN
        IF(ROW_P>=0 AND ROW_P<=15 )THEN -- UBICACAION DE MI FILA
            SELECTION <=data_i2c; -- union parea elegir la LETRA
            address<= ROW_P(3 DOWNTO 0);  -- LOS MENOS SIGNIFICATIVOS DE
            -- almacenamiento del dto
            var_img:=data_img; -- 8 bits , vectores de la Imagen
-- PART2: BIFURCACION
            IF(COL_P>=0  AND COL_P<=7 )THEN -- PARAMETRIZAMOIS LAS COLUMNAS , ESUINA
SUPERIOR IZQUIERDA
                var_bit:=var_img(7- TO_INTEGER(UNSIGNED(COL_P))); -- DAATOS DESDEel : MSB ->
LSB
            ELSE
                var_bit:='0'; -- FUERA DEL RANGO DE COL_´P solor azul
            END IF;
        ELSE
            var_bit:='0';
        END IF;

    ELSE -- asegureamos que se pinta sea AZUL
        var_bit:='0';
```

```
    END IF;

-- PART3: PINTADO
    IF(VISIBLE_P ='1')THEN
        IF(var_bit='1')THEN
            R_P<="1111";
            G_P<="0000";
            B_P<="0000";
        ELSE
            R_P<="0000";
            G_P<="0000";
            B_P<="1111";
        END IF;

    ELSE -- zona no visible
        R_P<="0000";
        G_P<="0000";
        B_P<="0000";
    END IF;

END PROCESS;



end Behavioral;
```

SINCRONIZADOR(TOP MODULE)

MODULO HYSYNC (SUBMODULE)

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 30.07.2020 11:13:19
-- Design Name:
-- Module Name: PINTA_VGA - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------


library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL; -- INCLUIMOS PARA QUE SE PUEDA USAR OPERACIONES ARITMETICAS EN STD_LOGIC TIPO DE DATOS Y OPERADORES MATEMATICOS
use ieee.numeric_std.all ; -- concvbersiond edatos TO_INTEGER


-- Uncomment the following library declaration if using
```

```
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx leaf cells in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity PINTA_VGA is
        Port (
            COL_P: IN STD_LOGIC_VECTOR(9 DOWNTO 0);  --ESQUINA SUPERIOR IZQUIERDA
            ROW_P: IN STD_LOGIC_VECTOR(9 DOWNTO 0); -- PARA  DIRECCION DE
MEMORIA DE ROM
            VISIBLE_P: IN STD_LOGIC; -- PORCH
            --
            data_img: IN STD_LOGIC_VECTOR(7 DOWNTO 0);   -- VECTOR DEVUELCVE LA
DIRTERCCION DE MEMORIA
            data_i2c: IN INTEGER RANGE 0 TO 50;       -- corazon de la bestia
            SELECTION: OUT INTEGER RANGE 0 TO 50;  --- PARA ELEGIR EL CARACTER DE LA
ROM
            address: OUT STD_LOGIC_VECTOR(3 DOWNTO 0); -- relacion ROW->ADDRESS
            -- COLOR
            R_P:OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
            G_P:OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
            B_P:OUT STD_LOGIC_VECTOR(3 DOWNTO 0)
        );
end PINTA_VGA;

architecture Behavioral of PINTA_VGA is

-- PINTADO: FONDO -> AZUL
    --     LETRA -> ROJO
```

```
begin
PROCESS(data_i2c,data_img,COL_P,ROW_P,VISIBLE_P) -- LISTA DE SENSIBILIDAD

VARIABLE var_bit: STD_LOGIC:='0'; -- PARA ALMACENRA EL bit BIFURCADO
VARIABLE var_img: STD_LOGIC_VECTOR(7 DOWNTO 0); -- PARA ALMACENRA el vector

BEGIN

--- PART1: verificacion de DATA
    IF(data_i2c /= 0) THEN
        IF(ROW_P>=0 AND ROW_P<=15 )THEN -- UBICACAION DE MI FILA
            SELECTION <=data_i2c; -- union parea elegir la LETRA
            address<= ROW_P(3 DOWNTO 0);  -- LOS MENOS SIGNIFICATIVOS DE
            -- almacenamiento del dto
            var_img:=data_img; -- 8 bits , vectores de la Imagen
-- PART2: BIFURCACION
            IF(COL_P>=0  AND COL_P<=7 )THEN -- PARAMETRIZAMOIS LAS COLUMNAS , ESUINA
SUPERIOR IZQUIERDA
                var_bit:=var_img(7- TO_INTEGER(UNSIGNED(COL_P))); -- DAATOS DESDEel : MSB ->
LSB
            ELSE
                var_bit:='0'; -- FUERA DEL RANGO DE COL_´P solor azul
            END IF;
        ELSE
            var_bit:='0';
        END IF;

    ELSE -- asegureamos que se pinta sea AZUL
        var_bit:='0';
```

```
    END IF;

-- PART3: PINTADO
    IF(VISIBLE_P ='1')THEN
        IF(var_bit='1')THEN
            R_P<="1111";
            G_P<="0000";
            B_P<="0000";
        ELSE
            R_P<="0000";
            G_P<="0000";
            B_P<="1111";
        END IF;

    ELSE -- zona no visible
        R_P<="0000";
        G_P<="0000";
        B_P<="0000";
    END IF;

END PROCESS;




end Behavioral;
```

MODULO VSYNC (SUBMODULE)

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
```

```
-- Create Date: 16.07.2020 09:11:25
-- Design Name:
-- Module Name: VSYNC - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------



library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL; -- INCLUIMOS PARA QUE SE PUEDA USAR OPERACIONES ARITMETICAS EN STD_LOGIC TIPO DE DATOS



entity VSYNC is
    Port (
        CLK_LINE: IN STD_LOGIC; -- FLANCO bajo -- CLK = 32u seconds
        CLR2: IN STD_LOGIC; -- nivel bajo , asincrono

        ROW: OUT STD_LOGIC_VECTOR(9 DOWNTO 0); --Contador de cada pixel - COLUMNAS
        --new_line: OUT STD_LOGIC; -- OJO : este sera como el reloj para VSYNC
        VSYNCR: OUT STD_LOGIC;-- solo indicara Los porches y la sincronizacion
```

```
            visible_F: OUT STD_LOGIC -- Solo lo visible 0 - 639 -- ZONA VISIBLE
        );
end VSYNC;

architecture Behavioral of VSYNC is

begin

PROCESS(CLR2,CLK_LINE)
---oJO VALOR INICIAL
VARIABLE count: STD_LOGIC_VECTOR(9 DOWNTO 0):="0000000000"; --Valor de para FILAS-  el BARRIDO en columnas
VARIABLE vision: STD_LOGIC:='1'; -- VISBIBLE
VARIABLE sincro: STD_LOGIC:='1'; -- HSYNC
VARIABLE linea: STD_LOGIC:='0';--new_line
VARIABLE aux: STD_LOGIC_VECTOR(9 DOWNTO 0):="0000000000";

BEGIN


   IF(CLR2='0')THEN
      count:="0000000000"; -- RETORNO A CERO
      --new_line<= '0'; -- DEFAULT
      VSYNCR<='1'; -- NO ESTA EN LA ZONA de sincronizacio HORIZONTAL
      VISIBLE_F<='1'; -- ZONA INICIAL , ZONA VISIBLE 1
      aux:="0000000001";

-- parte de BARRIDO DE filas

   ELSIF(CLK_LINE='0' AND CLK_LINE'EVENT) THEN -- rELOJ con flanco de bajada
```

```
        IF(aux = 1 AND count= 0)THEN
           count:="0000000000"; -- mantebnsmos el cero
           aux:="0000000000";
        ELSE
           count:=count+1;
        END IF;



        IF(count = 520)THEN -- RESOLUCION DE 800x520
            count:="0000000000";
        END IF;
    END IF;
-- parte VISIBLE
    IF(count <=479 )THEN -- SOLO POARTE VISIBLE!!!!!!!!!
        vision:='1';
    ELSE
        vision:='0';
    END IF;

-- parte PORCHE Y HSYNC
    IF(count>=480 AND count<=488 )THEN
        sincro:='0';
    ELSE
        sincro:='1';
    END IF;
-- NEW LINE
    --IF(count<=399)THEN
        --linea:='0';
    --ELSIF(count>=399 and count<=799)THEN
       --- linea:='1';
    --END IF;
```

```
-- ASIGNAMENTS


ROW<=count; -- HACE LA MAGIA

--new_line<=linea;

VSYNCR<=sincro;

visible_F<=vision;


END PROCESS;


end Behavioral;
```

MODULO PIX_FORM (SUBMODULE)

```
----------------------------------------------------------------------------------

-- Company:

-- Engineer: DENILSON V.G

--

-- Create Date: 15.07.2020 18:58:43

-- Design Name:

-- Module Name: PIX_FORM - Behavioral

-- Project Name:

-- Target Devices:

-- Tool Versions:

-- Description:

--

-- Dependencies:

--

-- Revision:

-- Revision 0.01 - File Created

-- Additional Comments:

--
```

```
----------------------------------------------------------------------------------


--- DESCRIPCION: DIVISOR DE FRECUENCIAS PARA FORMAR PIXEL


library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

USE IEEE.NUMERIC_STD.ALL; -- INCLUIMOS PARA QUE SE PUEDA USAR OPERACIONES
ARITMETICAS EN STD_LOGIC TIPO DE DATOS



entity PIX_FORM is
        Port (
           CLR: IN STD_LOGIC;
           CLK: IN STD_LOGIC;
           pix_form: OUT STD_LOGIC
        );
end PIX_FORM;

architecture Behavioral of PIX_FORM is



begin
PROCESS(CLR,CLK)--- LISTA DE SENSIBILIDAD

VARIABLE pix: STD_LOGIC:='0'; -- Valores uniciales
VARIABLE count: INTEGER:=0;

BEGIN --- T=10ns F=100MHz divisor de frecuencias
   IF( CLR = '0' )THEN -- ENYTRADA ASINCRONA
```

```
        pix:='0'; -- SALIDA bajo

    ELSIF(CLK='1' AND CLK'EVENT)THEN
        count:=count+1; -- cuenta los ESTADOS
        pix:='0'; -- estado bajo

        IF(count=4)THEN -- count= 4 ESTDOS  3 estado cambia la salida(IMMEDIATO)
            pix:='0'; -- Cmbio de estado
            count:=0;
        END IF;

        IF(count>=2)THEN
            pix:='1';
        END IF;

    END IF;
pix_form<=pix; -- ASIGNACION DE LA SALIDA

END PROCESS;
end Behavioral;
```

MODULO COUNT_PIX (SUBMODULE)

```
----------------------------------------------------------------------------------
-- Company:
-- Engineer:
--
-- Create Date: 16.07.2020 10:28:15
-- Design Name:
-- Module Name: COUNT_PIX - Behavioral
-- Project Name:
-- Target Devices:
-- Tool Versions:
```

```
-- Description:
--
-- Dependencies:
--
-- Revision:
-- Revision 0.01 - File Created
-- Additional Comments:
--
----------------------------------------------------------------------------------



library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
USE IEEE.STD_LOGIC_UNSIGNED.ALL; -- INCLUIMOS PARA QUE SE PUEDA USAR OPERACIONES ARITMETICAS EN STD_LOGIC TIPO DE DATOS



entity COUNT_PIX is
        Port (
           CLK_COUNT: IN STD_LOGIC; -- flanco bajada SEÑAL DE RELOJ DE 40ns = T
           CLR_C: IN STD_LOGIC; -- Active LOW
           pix_count: OUT STD_LOGIC_VECTOR(19 DOWNTO 0)-- CONTADOR DE PIXELES
        );
end COUNT_PIX;

architecture Behavioral of COUNT_PIX is

begin
PROCESS(CLK_COUNT,CLR_C)

VARIABLE count: STD_LOGIC_VECTOR(19 DOWNTO 0):="00000000000000000000"; -- CONTADOR DE PIXELES
```

```
VARIABLE aux_count: STD_LOGIC_VECTOR(19 DOWNTO 0); -- CONTADOR DE PIXELES



BEGIN
   IF(CLR_C = '0') THEN
      count:="00000000000000000000"; -- contador de pixel
      aux_count:="00000000000000000001";

   ELSIF(CLK_COUNT='0' AND CLK_COUNT'EVENT)THEN

    IF(count=0 AND aux_count=1 )THEN
      count:="00000000000000000000";
      aux_count:="00000000000000000000";
    ELSE
      count:= count+1;
    END IF;

   IF(count=307200)THEN
      count:="00000000000000000000";
   END IF;

   END IF;
pix_count<=count;

END PROCESS;



end Behavioral;
```