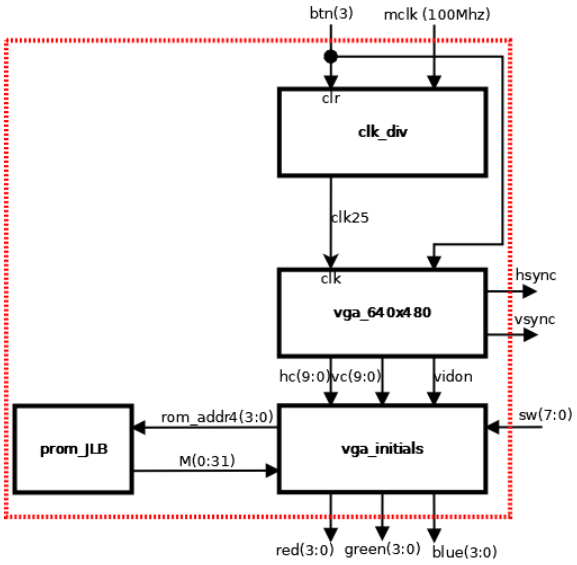
Diseño de Sistemas digitales

Actividad VI

lunes, noviembre 27, 2023

Diseñar e implementar un controlador de pantalla, que cumpla con el estándar VGA. Para ello, este deberá ajustarse al modelo mostrado en la figura.



Para describir una memoria PROM con sus iniciales en VHDL y acceder a ella:

```
entity prom_initials is
        port (addr: in std_logic_vector (3 downto 0);
              M: out std logic vector (0 to 31));
end prom_initials;

architecture Behavioral of prom_initials is
    type rom_array is array (NATURAL range<>) of
std_logic_vector (0 to 31);
    constant rom: rom_array := (
    "01111111100100000000011111111000", --0
    "00000100000100000000010000000100", --1
    "00000100000100000000010000000010", --2
    "00000100000100000000010000000010", --3
    "00000100000100000000010000000010", --4
    "00000100000100000000010000000100", --5
    "00000100000100000000010000001000", --6
    "00000100000100000000011111110000", --7
    "00000100000100000000010000001000", --8
    "00000100000100000000010000000100", --9
    "00000100000100000000010000000010", --A
    "01000100000100000000010000000010", --B
    "01000100000100000000010000000010", --C
    "01000100000100000000010000000010", --D
    "01000100000100000000010000000010", --E
    "00111000000111111111001111111100"  --F
    );
    --Un "0" = pixel apagado
    --Un "1" = el pixel correspondiente aparecerá en la pantalla
begin
    process (addr)
    variable j: integer;
    begin
        j:= conv_integer(addr);
        M <= rom(j); --En M tenemos el dato le de la direcci
    end process;
end Behavioral;
```

Para describir el controlador con los tiempos para una resolución de 640x480:

```
entity vga_640x480 is
    Port ( clk, clr : in STD_LOGIC;
           hsync : out STD_LOGIC;
           vsync : out STD_LOGIC;
           hc : out STD_LOGIC_VECTOR(9 downto 0);
           vc : out STD_LOGIC_VECTOR(9 downto 0);
           vidon : out STD_LOGIC);
end vga_640x480;

architecture Behavioral of vga_640x480 is
    --horizontal timig--
    constant hbp: std_logic_vector(9 downto 0) := "0010010000"; --HBP = SP+BP = 96+48 = 144
    constant hfp: std_logic_vector(9 downto 0) := "1100010000"; --HFP = HBP+HV = 144+640 = 784
    constant hpixels: std_logic_vector(9 downto 0) := "1100100000"; --quantity of pixels on horizontal
line = SP+BP+HV+FP = 96+48+640+16 = 800

    --vertical timig--
    constant vbp: std_logic_vector(9 downto 0) := "0000100011"; --VBP = SP+BP = 2+33 = 35
    constant vfp: std_logic_vector(9 downto 0) := "1000000011"; --VFP = VBP+VV = 35+480 = 515
    constant vlines: std_logic_vector(9 downto 0) := "1000001101"; --quantity of vertical lines on dis-
play = SP+BP+VV+FP = 2+33+480+10= 521
```

```
    signal hcs, vcs: std_logic_vector(9 downto 0); --horizontal & vertical counters
    signal vsenable: std_logic; --vertical counter enable
begin
    --horizontal counter syncronization signal
    process (clk,clr)
    begin
        if (clr = '1') then
            hcs <= "0000000000";
        elsif (rising_edge(clk)) then
            if (hcs = hpixels - 1) then --counter has reached end of count ?
                hcs <= "0000000000";    --reset
                vsenable <= '1';    --set flag to go vertical counter
            else
                hcs <= hcs + 1; --increment horizontal counter
                vsenable <= '0';    --clear vertical counter flag
            end if;
        end if;
    end process;
    hsync <= '0' when (hcs < 96) else '1'; --SP=0 when hc<128 pixels

     --vertical counter syncronization signal
    process (clk,clr,vsenable)
    begin
        if (clr = '1') then
            vcs <= "0000000000";
        elsif ((rising_edge(clk)) and (vsenable = '1')) then
            if (vcs = vlines - 1) then --counter has reached the end of count ?
                vcs <= "0000000000";    --reset
            else
                vcs <= vcs + 1; --increment vertical counter
            end if;
        end if;
    end process;
    vsync <= '0' when (vcs < 2) else '1'; --SP=0 when vc<2 lines
    vidon <= '1' when (((hcs < hfp) and (hcs >= hbp)) and ((vcs < vfp) and (vcs >= vbp))) else '0'; --
set video on when visible area

    --horizontal and vertical counters update
    hc <= hcs;
    vc <= vcs;

end Behavioral;
```

Para describir el módulo que envía los datos al conector y mueve el texto a través de los switches:

```
entity vga_initials is
        port( vidon: in std_logic;
              hc: in std_logic_vector (9 downto 0);
              vc: in std_logic_vector (9 downto 0);
              M: in std_logic_vector (0 to 31);
              sw: in std_logic_Vector (7 downto 0);
              rom_addr4: out std_logic_vector (3 downto 0);
              red: out std_logic_vector (3 downto 0);
              green: out std_logic_vector (3 downto 0);
              blue: out std_logic_vector (3 downto 0));
end vga_initials;

architecture Behavioral of vga_initials is
    constant hbp: std_logic_vector (9 downto 0) := "0010010000";
    --hbp =SP+BP=128+16=144
    constant vbp: std_logic_vector (9 downto 0) := "0000011111";
    --vbp = SP+BP=2+29=31

    --iniciales*************************
    constant w: integer := 32;
    --ancho
    constant h: integer := 16;
    --alto
    signal C1, R1: std_logic_vector (10 downto 0);
    --Columna,Renglon de posicion superior-izq.
```

Actividad VI

lunes, noviembre 27, 2023

```
    --*****************************************

    signal rom_addr, rom_pix: std_logic_vector (10 downto 0);
    --rengl?n-columna del bit en la pantalla
    signal spriteon: std_logic;
    --Determina si est? en la regi?n 16x32
    signal r, g, b: std_logic;
    --
begin
    --Columna y rengl?n con los switches
    C1 <= "00" & sw(3 downto 0) & "00001";
    R1 <= "00" & sw(7 downto 4) & "00001";
    rom_addr <= vc - vbp - R1;
    rom_pix <= hc - hbp - C1;
    rom_addr4 <= rom_addr (3 downto 0);

    --spriteon = "1" si est? en el area de la imagen
    spriteon <= '1' when (((hc >= C1 + hbp) and (hc < C1 + hbp + w ))
        and (( vc >= R1 + vbp) and (vc < R1 + vbp + h))) else '0';

    process (spriteon, vidon, rom_pix, M, r, g, b)
    variable j: integer;
    begin
        red <= "0000";
        green <= "0000";
        blue <= "0000";
        if (spriteon = '1' and vidon = '1') then
            j := conv_integer(rom_pix);
            r <= M(j);
            g <= M(j);
            b <= M(j);
            red <= r & r & r & r;
            green <= g & g & g & g;
            blue <= b & b & b & b;
        end if;
    end process;
end Behavioral;
```

¿A dónde enviar el archivo?

Su archivo "**.bit**" (empaquetado) debe ser depositado en su cuenta de **classroom**, para que registre su entrega. En la siguiente sesión, deberán probar su sistema en la tarjeta de desarrollo.