Universidad Nacional de San Martín

Electrónica digital 1

Trabajo práctico final (códigos)

Voltímetro digital

AUTOR:
Lautaro José Aguzin Parrilli

DOCENTES:

Miguel Angel Sagreras, Nicolas Álvarez.

1er cuatrimestre 2023

VOLTÍMETRO DIGITAL

Aguzin Parrilli, Lautaro José

Escuela de Ciencia y Tecnología, Universidad Nacional de San Martín, 25 de Mayo y Francia,1650 San Martín, Buenos Aires, Argentina
ljaguzinparrilli@estudiantes.unsam.edu.ar

Kit de desarrollo

```
entity volticonffd is
port(
pin_in: in bit;
pin_out: out bit;
clk_in: in bit;
rst: in bit:='0';
h_sync: out bit;
v_sync: out bit;
red:out bit;
green: out bit;
blue: out bit);

end;

architecture test of volticonffd is

component cont10b is
            port(
            clkc: in bit;
            rstc: in bit;
            enac: in bit;
            qoc: out bit_vector(0 to 9));
            end component;
   component volti is
      port(
      clkvolt: in bit;
      rstvolt: in bit;
      unos: in bit;
      hsvolt: out bit;
      vsvolt: out bit;
      redV:out bit;
      greenV: out bit;
      blueV: out bit);
   end component;
   component ffd is
      port (
         clki: in bit;
         rsti: in bit;
         enai: in bit;
         di: in bit;
         qo: out bit);
   end component;
```

```
signal unos_aux: bit;
signal clk25: bit;
signal qcont: bit_vector(0 to 9);

   begin
      dutvolti: volti port
map(clkvolt=>clk25,rstvolt=>rst,unos=>unos_aux,hsvolt=>h_sync,vsvolt=>v_sync,redV=>red,greenV=>green,blueV=>
blue);

      dutffd: ffd port map(clki=>clk25,rsti=>'0',enai=>'1',di=>pin_in,qo=>unos_aux);

            dutcont10b: cont10b port map(clkc=>clk_in,rstc=>'0',enac=>'1',qoc=>qcont);

            pin_out<= not unos_aux;
            clk25<=qcont(8);

end;
```

Flip flop D

```
entity ffd is
            port (
                        clki: in bit;
                        rsti: in bit;
                        enai: in bit;
                        di: in bit;
                        qo: out bit);
end ffd;

architecture ffdarq of ffd is
begin
process(clki,rsti)
begin
if (clki 'event and clki='1') then
            if rsti='1' then
                        qo <= '0';
                        elsif enai = '1' then
                                    qo <= di;
                        end if;
            end if;
end process;
end ffdarq;
```

Voltimetro

```
entity volti is
   port(
      clkvolt: in bit;
      rstvolt: in bit;
      unos: in bit;
      hsvolt: out bit;
      vsvolt: out bit;
      redV:out bit;
      greenV: out bit;
      blueV: out bit
   );
end;
architecture test of volti is

         --DECLARACION DE COMPONENTES-----
         component cont_bin_330 is
         port(
            clk330: in bit;
            rst330: in bit;
            ena330: in bit:='1';
            out1, out2 : out bit);
         end component;
         component cont1s is
            port(
               clk1s: in bit;
               rst1s: in bit;
               ena1s: in bit;
               data_out0: out bit_vector(0 to 3);
               data_out1: out bit_vector(0 to 3);
               data_out4: out bit_vector(0 to 3);
               data_out3: out bit_vector(0 to 3);
               data_out2: out bit_vector(0 to 3));
         end component;
         component reg is
            port(
               clkr: in bit;
               rstr: in bit:= '0';
               enar: in bit;
               bcd0,bcd1,bcd2: in bit_vector(0 to 3);
               a_0,a_1,a_2: out bit_vector(0 to 3));
         end component;
         component mux is
            port(
               a0: in bit_vector(0 to 3); --codigo para el bit mas significativo
               a1: in bit_vector(0 to 3); --codigo para el segundo bit mas significativo
               a2: in bit_vector(0 to 3); --codigo para el bit menos significativo
               a3: in bit_vector(0 to 3) := "1010"; -- codigo para el punto
               a4: in bit_vector(0 to 3) := "1011"; --codigo para la V
               a5: in bit_vector (0 to 3):= "1100"; -- codigo par el espacio
               pixel_x: in bit_vector (0 to 9); --posicion horizontal en la pantalla
               pixel_y: in bit_vector (0 to 9);
            mux_out: out bit_vector (0 to 3));
        end component;
        component rom is
          port(
            pixel_x: in bit_vector(0 to 9);
            pixel_y: in bit_vector(0 to 9);
            char_adress: in bit_vector(0 to 3);
            rom_out: out bit);
        end component;
        component vga is
          port(
            clkvga: in bit;
            data_in_vga: in bit;
            hsyncvga, vsyncvga: out bit;
            r_vga,g_vga,b_vga: out bit;
            pixely_vidon: out bit;
            pixel_x_vga,pixel_y_vga: out bit_vector(0 to 9));
        end component;
        --FIN DECLARACION DE COMPONENTES-------------

    --DECLARACION DE SENIALES----------
signal pixel_x_aux,pixel_y_aux: bit_vector(0 to 9);
signal v_vidon_aux: bit;
signal not_v_vidon_aux: bit;
signal data4,data3: bit_vector(0 to 3);

signal not_vvidon_and_out1: bit;

signal out1_aux, out2_aux, out3_aux: bit;
signal a0_c1s_reg: bit_vector(0 to 3);
signal a1_c1s_reg: bit_vector(0 to 3);
signal a2_c1s_reg: bit_vector(0 to 3);
signal a0_reg_mux: bit_vector(0 to 3);
signal a1_reg_mux: bit_vector(0 to 3);
signal a2_reg_mux: bit_vector(0 to 3);


signal mux_rom :bit_vector(0 to 3);
signal rom_out_aux: bit;
    --FIN DECLARACION DE SENIALES------

begin
    dutcontbin330: cont_bin_330 port
map(clk330=>clkvolt,rst330=>rstvolt,ena330=>'1',out1=>out1_aux,out2=>out2_aux);

    dutcont1s: cont1s port map(clk1s=>clkvolt,rst1s=>out2_aux,ena1s=>unos,data_out0=>a0_c1s_reg,
data_out1=>a1_c1s_reg, data_out2=>a2_c1s_reg,data_out3=>data3,data_out4=>data4);

    dutreg: reg port map(clkr=>clkvolt,rstr=>'0', enar=>not_vvidon_and_out1,
bcd0=>a0_c1s_reg,bcd1=>a1_c1s_reg,bcd2=>a2_c1s_reg, a_0=>a0_reg_mux,a_1=>a1_reg_mux,a_2=>a2_reg_mux);

    not_vvidon_and_out1<= out1_aux and not_v_vidon_aux;
    not_v_vidon_aux<= not v_vidon_aux;
    dutmux: mux port
map(a0=>a0_reg_mux,a1=>a1_reg_mux,a2=>a2_reg_mux,a3=>"1010",a4=>"1011",a5=>"1100",pixel_x=>pixel_x_au
x,pixel_y=>pixel_y_aux,mux_out=>mux_rom);

    dutrom: rom port
map(pixel_x=>pixel_x_aux,pixel_y=>pixel_y_aux,char_adress=>mux_rom,rom_out=>rom_out_aux);

    dutvga: vga port
map(clkvga=>clkvolt,data_in_vga=>rom_out_aux,hsyncvga=>hsvolt,vsyncvga=>vsvolt,r_vga=>redV,g_vga=>greenV,b
_vga=>blueV,pixely_vidon=>v_vidon_aux,pixel_x_vga=>pixel_x_aux,pixel_y_vga=>pixel_y_aux);
end;
```

Contador binario de 33000

```
entity cont_bin_330 is
port(
clk330: in bit;
rst330: in bit:='0';
ena330: in bit:='1';
out1, out2 : out bit
);
end;

architecture test of cont_bin_330 is

    ---DECLARACION DE COMPONENTES--------------
    component cont16b
        port(
            clkc: in bit;
            rstc: in bit;
            enac: in bit;
            qoc: out bit_vector(0 to 15));
    end component;

    component ffd
        port (
            clki: in bit;
            rsti: in bit;
            enai: in bit;
            di: in bit;
            qo: out bit);
    end component;
    -------------------------------------------

signal flag330, rst_aux: bit;
signal q_aux: bit_vector(0 to 15);

begin

        --  0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
--33000 = --1 0 0 0 0 0 0 0 1 1 1  0  1   0  0 0
dut: cont16b port map(clkc=>clk330,rstc=>rst_aux,enac=>ena330,qoc=>q_aux);

flag330 <= q_aux(0) and (not q_aux(1)) and (not q_aux(2)) and (not q_aux(3)) and (not q_aux(4)) and (not q_aux(5))
and (not q_aux(6)) and (not q_aux(7)) and q_aux(8) and q_aux(9) and q_aux(10) and (not q_aux(11)) and q_aux(12)
and (not q_aux(13)) and (not q_aux(14)) and (not q_aux(15));
rst_aux <= flag330 or rst330;
out1<=flag330;


dut1: ffd port map(clki=>clk330, rsti=>rst330, enai=> ena330, di=>flag330,qo=>out2);
end;
```

Contador de 16 bits

```
entity cont16b is
   port(
      clkc: in bit;
      rstc: in bit:='0';
      enac: in bit:='1';
      qoc: out bit_vector(0 to 15)
      );
   end;

   architecture test of cont16b is

   signal ds,qos: bit_vector(0 to 15);
   -----------------------------
   component ffd
      port (
         clki: in bit;
         rsti: in bit;
         enai: in bit;
         di: in bit;
         qo: out bit);
   end component;
   -----------------------------
   begin
   ffd15: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(15),enai=>enac,qo=>qos(15));
   ds(15) <= enac xor qos(15);

   ffd14: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(14),enai=>enac,qo=>qos(14));
   ds(14)<= (qos(15) and enac) xor qos(14);

   ffd13: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(13),enai=>enac,qo=>qos(13));
   ds(13) <= (qos(14) and (qos(15) and enac)) xor qos(13);

   ffd12: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(12),enai=>enac,qo=>qos(12));
   ds(12)<= (qos(13) and (qos(14) and (qos(15) and enac))) xor qos(12);

   ffd11: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(11),enai=>enac,qo=>qos(11));
   ds(11)<= (qos(12) and (qos(13) and (qos(14) and (qos(15) and enac)))) xor qos(11);

   ffd10: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(10),enai=>enac,qo=>qos(10));
   ds(10) <= (qos(11) and (qos(12) and (qos(13) and (qos(14) and (qos(15) and enac))))) xor qos(10);

   ffd9: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(9),enai=>enac,qo=>qos(9));
      ds(9) <= (qos(10) and (qos(11) and (qos(12) and (qos(13) and (qos(14) and (qos(15) and enac)))))) xor qos(9);

   ffd8: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(8),enai=>enac,qo=>qos(8));
      ds(8)<= (qos(9) and (qos(10) and (qos(11) and (qos(12) and (qos(13) and (qos(14) and (qos(15) and enac)))))))
xor qos(8);

   ffd7: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(7),enai=>enac,qo=>qos(7));
      ds(7)<= (qos(8) and (qos(9) and (qos(10) and (qos(11) and (qos(12) and (qos(13) and (qos(14) and (qos(15) and
enac)))))))) xor qos(7);
```

```
    ffd6: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(6),enai=>enac,qo=>qos(6));
        ds(6)<= (qos(7) and (qos(8) and (qos(9) and (qos(10) and (qos(11) and (qos(12) and (qos(13) and (qos(14) and
(qos(15) and enac))))))))) xor qos(6);

    ffd5: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(5),enai=>enac,qo=>qos(5));
        ds(5)<= (qos(6) and (qos(7) and (qos(8) and (qos(9) and (qos(10) and (qos(11) and (qos(12) and (qos(13) and
(qos(14) and (qos(15) and enac)))))))))) xor qos(5);

    ffd4: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(4),enai=>enac,qo=>qos(4));
        ds(4)<= (qos(5) and (qos(6) and (qos(7) and (qos(8) and (qos(9) and (qos(10) and (qos(11) and (qos(12) and
(qos(13) and (qos(14) and (qos(15) and enac))))))))))) xor qos(4);

    ffd3: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(3),enai=>enac,qo=>qos(3));
        ds(3)<= (qos(4) and (qos(5) and (qos(6) and (qos(7) and (qos(8) and (qos(9) and (qos(10) and (qos(11) and
(qos(12) and (qos(13) and (qos(14) and (qos(15) and enac)))))))))))) xor qos(3);

    ffd2: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(2),enai=>enac,qo=>qos(2));
        ds(2)<=(qos(3) and (qos(4) and (qos(5) and (qos(6) and (qos(7) and (qos(8) and (qos(9) and (qos(10) and (qos(11)
and (qos(12) and (qos(13) and (qos(14) and (qos(15) and enac))))))))))))) xor qos(2);

    ffd1: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(1),enai=>enac,qo=>qos(1));
        ds(1)<= (qos(2) and (qos(3) and (qos(4) and (qos(5) and (qos(6) and (qos(7) and (qos(8) and (qos(9) and (qos(10)
and (qos(11) and (qos(12) and (qos(13) and (qos(14) and (qos(15) and enac)))))))))))))) xor qos(1);

    ffd0: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(0),enai=>enac,qo=>qos(0));
        ds(0)<= (qos(1) and (qos(2) and (qos(3) and (qos(4) and (qos(5) and (qos(6) and (qos(7) and (qos(8) and (qos(9)
and (qos(10) and (qos(11) and (qos(12) and (qos(13) and (qos(14) and (qos(15) and enac))))))))))))))) xor qos(0);

    qoc <= qos;
    end;
```

Contador de unos (bcd de 5 digitos)

```
entity cont1s is
   port(
      clk1s: in bit;
      rst1s: in bit;
      ena1s: in bit;
      data_out0: out bit_vector(0 to 3);
      data_out1: out bit_vector(0 to 3);
      data_out4: out bit_vector(0 to 3);
      data_out3: out bit_vector(0 to 3);
      data_out2: out bit_vector(0 to 3)
   );
end;

architecture test of cont1s is
   --------------------------------
   component bcd is
   port(
      clkb: in bit;
      rstb: in bit;
      enab: in bit;
      f: out bit;
      qob: out bit_vector(0 to 3));
   end component;
   ---------------------------------
   signal flag4: bit;
   signal flag3: bit;
   signal flag2: bit;  -- se?al de clock que sale de las unidades
   signal flag1: bit;
   signal flag0: bit; -- se?al de clock que sale de las decenas
   signal qo0,qo1,qo2,qo3,qo4: bit_vector(0 to 3); --salidas de los bcds
   signal ena0,ena1,ena2,ena3: bit;

   begin
      bcd4 : bcd port map (clkb=>clk1s,rstb=>rst1s, enab=> ena1s,f=>flag4,qob=>qo4); --menos significativo
      bcd3 : bcd port map (clkb=>clk1s,rstb=>rst1s, enab=> ena3,f=>flag3,qob=>qo3);
      bcd2 : bcd port map (clkb=>clk1s,rstb=>rst1s, enab=> ena2,f=>flag2,qob=>qo2);
      bcd1 : bcd port map (clkb=>clk1s,rstb=>rst1s, enab=> ena1,f=>flag1,qob=>qo1);
      bcd0 : bcd port map (clkb=>clk1s,rstb=>rst1s, enab=> ena0,f=>flag0,qob=>qo0); --MAS SIGNIFICATIVO

  ena3<= flag4 and ena1s;
  ena2<= flag3 and ena3;
  ena1<= flag2 and ena2;
  ena0<= flag1 and ena1;

   data_out0 <= qo0;
   data_out1 <= qo1;
   data_out2 <= qo2;
   data_out3 <= qo3;
   data_out4 <= qo4;

end;
```

BCD

```
entity bcd is
port(
clkb: in bit;
rstb: in bit;
enab: in bit;
f: out bit;
qob: out bit_vector(0 to 3));
end;

architecture test of bcd is

signal ds,qos: bit_vector(0 to 3);
--------------------------------------
component ffd
port (
clki: in bit;
rsti: in bit;
enai: in bit;
di: in bit;
qo: out bit);
end component;
--------------------------------------



begin
ffd3: ffd port map(clki=>clkb,rsti=>rstb,enai=>enab,di=>ds(3),qo=>qos(3));

ffd2: ffd port map(clki=>clkb,rsti=>rstb,enai=>enab,di=>ds(2),qo=>qos(2));

ffd1: ffd port map(clki=>clkb,rsti=>rstb,enai=>enab,di=>ds(1),qo=>qos(1));

ffd0: ffd port map(clki=>clkb, rsti=>rstb,enai=>enab,di=>ds(0),qo=>qos(0));

ds(0)<= (qos(0) and (not qos(2)) and (not qos(3))) or (qos(1) and qos(2) and qos(3));
ds(1) <= (qos(1) and (not qos(2))) or (qos(1) and (not qos(3))) or ((not qos(1)) and qos(2) and  qos(3));
ds(2) <= (qos(2) and (not qos(3))) or ((not qos(0)) and (not qos(2)) and qos(3));
ds(3) <= not qos(3);

qob <= qos;
f <= qos(0) and (not qos(1) and (not qos(2)) and qos(3));
end;
```

Registro completo

```
--registro--
entity reg is
	port(
	clkr: in bit;
	rstr: in bit:= '0';
	enar: in bit;
	bcd0,bcd1,bcd2: in bit_vector(0 to 3);
	a_0,a_1,a_2: out bit_vector(0 to 3));
end;
architecture test of reg is
	---------------------------------------------
	component reg4 is
		port(
			clkr4: in bit;
			rstr4: in bit;
			enar4: in bit;
			dr4: in bit_vector(0 to 3);
			qr4: out bit_vector(0 to 3));
	end component;
	---------------------------------------------
begin
	reg4_0: reg4 port map(clkr4 => clkr, rstr4 => rstr, enar4 => enar, dr4 => bcd0, qr4 => a_0);
	reg4_1: reg4 port map(clkr4 => clkr, rstr4 => rstr, enar4 => enar, dr4 => bcd1, qr4 => a_1);
	reg4_2: reg4 port map(clkr4 => clkr, rstr4 => rstr, enar4 => enar, dr4 => bcd2, qr4 => a_2);
end;
```

Regsistro de 4 bits

```
entity reg4 is
	port(
	clkr4: in bit;
	rstr4: in bit:='0';
	enar4: in bit;
	dr4: in bit_vector(0 to 3);
	qr4: out bit_vector(0 to 3));
end;

architecture test of reg4 is
	component ffd is
	port (
		clki: in bit;
		rsti: in bit;
		enai: in bit;
		di: in bit;
		qo: out bit);
	end component;
begin
ffd0: ffd port map(clki=>clkr4,rsti=>rstr4,enai=>enar4,di=>dr4(0),qo=>qr4(0));
ffd1: ffd port map(clki=>clkr4,rsti=>rstr4,enai=>enar4,di=>dr4(1),qo=>qr4(1));
ffd2: ffd port map(clki=>clkr4,rsti=>rstr4,enai=>enar4,di=>dr4(2),qo=>qr4(2));
ffd3: ffd port map(clki=>clkr4,rsti=>rstr4,enai=>enar4,di=>dr4(3),qo=>qr4(3));

end;
```

Multiplexor

```
entity mux is
port(
a0: in bit_vector(0 to 3); --codigo para el bit mas significativo
a1: in bit_vector(0 to 3); --codigo para el segundo bit mas significativo
a2: in bit_vector(0 to 3); --codigo para el bit menos significativo
a3: in bit_vector(0 to 3) := "1010"; -- codigo para el punto
a4: in bit_vector(0 to 3) := "1011"; --codigo para la V
a5: in bit_vector (0 to 3):= "1100"; -- codigo par el espacio
pixel_x: in bit_vector (0 to 9); --posicion horizontal en la pantalla
pixel_y: in bit_vector (0 to 9);
mux_out: out bit_vector (0 to 3)
);
end;

architecture test of mux is

signal sel: bit_vector(0 to 2);
signal sely: bit_vector(0 to 2);
--sel: a0=000 a1=010 a2=011 a3=001 a4=100 a5=101
signal salida_and_a0: bit_vector(0 to 3);
signal salida_and_a1: bit_vector(0 to 3);
signal salida_and_a2: bit_vector(0 to 3);
signal salida_and_a3: bit_vector(0 to 3);
signal salida_and_a4: bit_vector(0 to 3);
signal salida_and_a5: bit_vector(0 to 3);
signal mux_out_aux: bit_vector(0 to 3);

begin
sel <= pixel_x(0 to 2);
sely <= pixel_y(0 to 2);

--SELECCION A0--
salida_and_a0(0) <= a0(0) and (not sel(0)) and (not sel(1)) and (not sel(2));
salida_and_a0(1) <= a0(1) and (not sel(0)) and (not sel(1)) and (not sel(2));
salida_and_a0(2) <= a0(2) and (not sel(0)) and (not sel(1)) and (not sel(2));
salida_and_a0(3) <= a0(3) and (not sel(0)) and (not sel(1)) and (not sel(2));

--seleccion A1--
salida_and_a1(0) <= a1(0) and (not sel(0)) and sel(1) and (not sel(2));
salida_and_a1(1) <= a1(1) and (not sel(0)) and sel(1) and (not sel(2));
salida_and_a1(2) <= a1(2) and (not sel(0)) and sel(1) and (not sel(2));
salida_and_a1(3) <= a1(3) and (not sel(0)) and sel(1) and (not sel(2));

--SELECCION A2--
salida_and_a2(0) <= a2(0) and (not sel(0)) and sel(1) and sel(2);
salida_and_a2(1) <= a2(1) and (not sel(0)) and sel(1) and sel(2);
salida_and_a2(2) <= a2(2) and (not sel(0)) and sel(1) and sel(2);
salida_and_a2(3) <= a2(3) and (not sel(0)) and sel(1) and sel(2);

--SELECCION A3--
```

```
salida_and_a3(0) <= a3(0) and (not sel(0)) and (not sel(1)) and sel(2);
salida_and_a3(1) <= a3(1) and (not sel(0)) and (not sel(1)) and sel(2);
salida_and_a3(2) <= a3(2) and (not sel(0)) and (not sel(1)) and sel(2);
salida_and_a3(3) <= a3(3) and (not sel(0)) and (not sel(1)) and sel(2);

--SELECCION A4--
salida_and_a4(0) <= a4(0) and sel(0) and (not sel(1)) and (not sel(2));
salida_and_a4(1) <= a4(1) and sel(0) and (not sel(1)) and (not sel(2));
salida_and_a4(2) <= a4(2) and sel(0) and (not sel(1)) and (not sel(2));
salida_and_a4(3) <= a4(3) and sel(0) and (not sel(1)) and (not sel(2));

--SELECCION A5--
salida_and_a5(0) <= a5(0) and sel(0) and (not sel(1)) and sel(2);
salida_and_a5(1) <= a5(1) and sel(0) and (not sel(1)) and sel(2);
salida_and_a5(2) <= a5(2) and sel(0) and (not sel(1)) and sel(2);
salida_and_a5(3) <= a5(3) and sel(0) and (not sel(1)) and sel(2);


mux_out(0) <= (salida_and_a0(0) or salida_and_a1(0) or salida_and_a2(0) or salida_and_a3(0) or salida_and_a4(0)
or salida_and_a5(0));
mux_out(1) <= (salida_and_a0(1) or salida_and_a1(1) or salida_and_a2(1) or salida_and_a3(1) or salida_and_a4(1)
or salida_and_a5(1));
mux_out(2) <= (salida_and_a0(2) or salida_and_a1(2) or salida_and_a2(2) or salida_and_a3(2) or salida_and_a4(2)
or salida_and_a5(2));
mux_out(3) <= (salida_and_a0(3) or salida_and_a1(3) or salida_and_a2(3) or salida_and_a3(3) or salida_and_a4(3)
or salida_and_a5(3));


end;
```

ROM

```
entity rom is
port(
pixel_x: in bit_vector(0 to 9);
pixel_y: in bit_vector(0 to 9);
char_adress: in bit_vector(0 to 3);
rom_out: out bit);

end;

architecture test of rom is

type matriz is array (0 to 103) of bit_vector(0 to 7);

----------------------------------------------------------------------------------------------------------
constant memoria: matriz:=
        ("01111110","01000010","01000010","01000010","01000010","01000010","01000010","01111110",    --0
        0000
        "00001000","00001000","00001000","00001000","00001000","00001000","00001000","00001000",    --1
        0001
        "01111110","01000010","00000110","00001000","00010000","00100000","01000000","01111110",    --2
        0010
        "01111110","00000010","00000010","00111110","00000010","00000010","00000010","01111110",    --3
        0011
        "01000010","01000010","01000010","01000010","01111110","00000010","00000010","00000010",    --4
        0100
        "01111110","01000000","01000000","01111110","00000010","00000010","00000010","01111110",    --5
0101
        "00011110","00100000","01000000","01111110","01000010","01000010","01000010","01111110",    --6
        0110
        "01111110","00000010","00000100","00000100","00001000","00001000","00010000","00010000",    --7
        0111
        "01111110","01000010","01000010","01111110","01000010","01000010","01000010","01111110",    --8
        1000
        "01111110","01000010","01000010","01111110","00000010","00000010","00000010","01111110",    --9
        1001
        "00000000","00000000","00000000","00000000","00000000","00000000","00011000","00011000",    --.
        1010
        "01000010","01000010","01000010","00100100","00100100","00100100","00011000","00011000",    --V
         1011
        "00000000","00000000","00000000","00000000","00000000","00000000","00000000","00000000");     --
espacio 1100
----------------------------------------------------------------------------------------------------------
signal char_adress_aux: bit_vector(0 to 6);

signal char_adress_aux_aux: bit_vector(6 downto 0);
signal pixel_x_aux: bit_vector(2 downto 0);

-----FUNCION PARA CONERTIR A ENTERO-----

function conv_int(word:bit_vector) return integer is
```

```
variable resultado: integer:= 0 ;

begin
for i in word'range loop
        if (word(i)= '1') then
                resultado:= resultado + 2**i;
        end if;
end loop;
return resultado;
end conv_int;
-----------------------------------------

begin
        char_adress_aux <= char_adress & pixel_y(3 to 5);

        -----------------------------
        pixel_x_aux(2)<= pixel_x(3);
        pixel_x_aux(1)<=pixel_x(4);
        pixel_x_aux(0)<=pixel_x(5);

        char_adress_aux_aux(6)<=char_adress_aux(0);
        char_adress_aux_aux(5)<=char_adress_aux(1);
        char_adress_aux_aux(4)<=char_adress_aux(2);
        char_adress_aux_aux(3)<=char_adress_aux(3);
        char_adress_aux_aux(2)<=char_adress_aux(4);
        char_adress_aux_aux(1)<=char_adress_aux(5);
        char_adress_aux_aux(0)<=char_adress_aux(6);
        -----------------------------



        rom_out<= memoria(conv_int(char_adress_aux_aux))(conv_int(pixel_x_aux));
end;
```

VGA

```
entity vga is
   port(
      clkvga: in bit;
      data_in_vga: in bit;
      hsyncvga, vsyncvga: out bit;
      r_vga,g_vga,b_vga: out bit;
      pixely_vidon: out bit;
      pixel_x_vga,pixel_y_vga: out bit_vector(0 to 9)
   );
end;

architecture test of vga is
   ------------------------------------
   component gensyncs is
      port(
               clksync: in bit;
               pixel_y_syncs, pixel_x_syncs: out bit_vector(0 to 9);
               h_sync,v_sync: out bit;
               pixy_vidon: out bit;
               vid_on: out bit:='1'
            );
   end component;
   ------------------------------------
   component genpixels is
      port(
         r_i: in bit;
         g_i: in bit;
         b_i: in bit:='1';
         vidon_genp: in bit;
         pixel_y: in bit_vector(0 to 9);
         r_o,g_o,b_o: out bit
      );
   end component;
   ------------------------------------
   signal vidon_aux: bit;
   signal pixel_y_aux: bit_vector(0 to 9);
begin
   dutgs: gensyncs port map(
      clksync=>clkvga,
      pixel_y_syncs=>pixel_y_aux,
      pixel_x_syncs=>pixel_x_vga,
      h_sync=>hsyncvga,
      v_sync=>vsyncvga,
      pixy_vidon=>pixely_vidon,
      vid_on => vidon_aux);
     pixel_y_vga<=pixel_y_aux;

   dutgp: genpixels port map(
      r_i=>data_in_vga,
      g_i=>data_in_vga,
      b_i=>'1',
      vidon_genp=>vidon_aux,
      pixel_y=>pixel_y_aux,
      r_o=>r_vga,
      g_o=>g_vga,
      b_o=>b_vga);
end;
```

Generador de pixeles

```
entity genpixels is
port(
r_i: in bit;
g_i: in bit;
b_i: in bit:='1';
vidon_genp: in bit;
pixel_y: in bit_vector(0 to 9);
r_o,g_o,b_o: out bit
);
end;

architecture test of genpixels is

signal sely: bit_vector(0 to 2);
begin
r_o<=r_i and vidon_genp and (not pixel_y(1)) and (not pixel_y(2));
g_o<=g_i and vidon_genp and (not pixel_y(1)) and (not pixel_y(2));
b_o<=b_i and vidon_genp;

end;
```

Generador de pulsos

```
entity gensyncs is
	port(
	clksync: in bit;
	pixel_y_syncs, pixel_x_syncs: out bit_vector(0 to 9);
	h_sync,v_sync: out bit;
	pixy_vidon: out bit;
	vid_on: out bit:='1'

	);
end;
architecture test of gensyncs is

	--DECLARACION DE COMPONENTES-----------------
	component contV
		port(
			clkcontv: in bit;
			rstcontv: in bit;
			encontv: in bit;
			pixel_y_contv: out bit_vector (0 to 9));
	end component;
	--------------------------------------------
	component contH
		port(
			clkconth: in bit;
			rstconth: in bit;
			enconth: in bit;
			pixel_x_conth: out bit_vector (0 to 9);
			flag_rst: out bit);
	end component;
	--------------------------------------------
	component vidon
		port(
			clkh_vidon: in bit;
			clkv_vidon: in bit;
			pixel_y_vidon: in bit_vector (0 to 9);
			pixel_x_vidon: in bit_vector (0 to 9);
			salida_vidon: out bit:='1';
			v_vidon: out bit);
	end component;
	--------------------------------------------
	component vsync
		port(
			clkvsync: in bit;
			bits_contador_vsync: bit_vector(0 to 9);
			salidavsync: out bit);
	end component;
	--------------------------------------------
	component hsync
		port(
			clkhsync: in bit;
			bits_contador_hsync: bit_vector(0 to 9);
```

```
                        salidahsync: out bit);
        end component;
        --------------------------------------------
        ---DECLARACION DE SEÑALES--------------------
        signal pixel_y_aux, pixel_x_aux: bit_vector (0 to 9);
        signal flag_rst_aux: bit;

begin

cv: contV port map(clkcontv=>clksync, rstcontv=>'0', encontv=>flag_rst_aux, pixel_y_contv=>pixel_y_aux);

ch: contH port map(clkconth=>clksync, rstconth=>'0', enconth=>'1', pixel_x_conth=>pixel_x_aux,
flag_rst=>flag_rst_aux);

video_on: vidon port map(clkh_vidon=>clksync, clkv_vidon=>clksync, pixel_y_vidon=>pixel_y_aux, pixel_x_vidon=>
pixel_x_aux, salida_vidon=> vid_on,v_vidon=>pixy_vidon);

syncver: vsync port map(clkvsync=>clksync, bits_contador_vsync=>pixel_y_aux, salidavsync=>v_sync);

synchor: hsync port map(clkhsync=>clksync, bits_contador_hsync=>pixel_x_aux, salidahsync=>h_sync);

pixel_x_syncs <= pixel_x_aux;
pixel_y_syncs <= pixel_y_aux;
end;
```

Contador Vertical

```
entity contV is
port(
	clkcontv: in bit;
	rstcontv: in bit;
	encontv: in bit;
	pixel_y_contv: out bit_vector (0 to 9));
end;
architecture test of contV is
	---------------------------------------------
	component cont10b
		port(
			clkc: in bit;
			rstc: in bit;
			enac: in bit;
			qoc: out bit_vector(0 to 9));
	end component;
	---------------------------------------------
	component compxigual
		port(
			a: in bit_vector (9 downto 0);
			b: in bit_vector (9 downto 0);
			s: out bit);
	end component;
	---------------------------------------------
	signal bits_contador: bit_vector(0 to 9);
	signal rst_aux, rstcont: bit;
begin
	contver: cont10b port map(clkc=>clkcontv, rstc=>rstcont, enac=>encontv,qoc=>bits_contador);

	comprstv: compxigual port map(a=>bits_contador,b=>"1000001101", s=> rst_aux); --525

	rstcont<= rst_aux or rstcontv;

	pixel_y_contv <= bits_contador;

end;
```

Contador horizontal

```
entity contH is
port(
	clkconth: in bit;
	rstconth: in bit;
	enconth: in bit;
	pixel_x_conth: out bit_vector (0 to 9);
	flag_rst: out bit
);
end;

architecture test of contH is
	---------------------------------------------
	component cont10b
```

```
		port(
			clkc: in bit;
			rstc: in bit;
			enac: in bit;
			qoc: out bit_vector(0 to 9));
	end component;
	-------------------------------------------
	component compxigual
		port(
			a: in bit_vector (9 downto 0);
			b: in bit_vector (9 downto 0);
			s: out bit);
	end component;
	--------------------------------------------

	signal bits_contador: bit_vector(0 to 9);
	signal flag_rst_aux, rstcont: bit;
begin
	conthor: cont10b port map(clkc=>clkconth, rstc=>rstcont, enac=>enconth,qoc=>bits_contador);

	comprsth: compxigual port map(a=>bits_contador,b=>"1100011111", s=> flag_rst_aux); --799

	rstcont <= rstconth or flag_rst_aux;

	pixel_x_conth <= bits_contador;
	flag_rst<=flag_rst_aux;
end;
```

Comparador por igual

```
entity compxigual is
	port(
	a: in bit_vector (9 downto 0);
	b: in bit_vector (9 downto 0);
	s: out bit
	);
end;

architecture test of compxigual is

signal salidas_xnor: bit_vector (9 downto 0);


begin

salidas_xnor <= a xnor b;
s <= salidas_xnor(0) and salidas_xnor(1) and salidas_xnor(2) and salidas_xnor(3) and salidas_xnor(4) and
salidas_xnor(5) and salidas_xnor(6) and salidas_xnor(7) and salidas_xnor(8) and salidas_xnor(9);


end;
```

Contador de 10 bits

```
entity cont10b is
port(
	clkc: in bit;
	rstc: in bit;
	enac: in bit;
	qoc: out bit_vector(0 to 9));
end;
architecture test of cont10b is
signal ds,qos: bit_vector(0 to 9);
component ffd
	port (
		clki: in bit;
		rsti: in bit;
		enai: in bit;
		di: in bit;
		qo: out bit);
end component;
begin
ffd9: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(9),enai=>enac,qo=>qos(9));
   ds(9) <= enac xor qos(9);

ffd8: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(8),enai=>enac,qo=>qos(8));
   ds(8)<= (qos(9) and enac) xor qos(8);

ffd7: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(7),enai=>enac,qo=>qos(7));
   ds(7)<= (qos(8) and (qos(9) and enac) ) xor qos(7);

ffd6: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(6),enai=>enac,qo=>qos(6));
   ds(6)<= (qos(7) and (qos(8) and (qos(9) and enac))) xor qos(6);

ffd5: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(5),enai=>enac,qo=>qos(5));
   ds(5)<= (qos(6) and (qos(7) and (qos(8) and (qos(9) and enac)))) xor qos(5);

ffd4: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(4),enai=>enac,qo=>qos(4));
   ds(4)<= (qos(5) and (qos(6) and (qos(7) and (qos(8) and (qos(9) and enac))))) xor qos(4);

ffd3: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(3),enai=>enac,qo=>qos(3));
   ds(3)<= (qos(4) and (qos(5) and (qos(6) and (qos(7) and (qos(8) and (qos(9) and enac)))))) xor qos(3);

ffd2: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(2),enai=>enac,qo=>qos(2));
   ds(2)<= (qos(3) and (qos(4) and (qos(5) and (qos(6) and (qos(7) and (qos(8) and (qos(9) and enac))))))) xor qos(2);

ffd1: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(1),enai=>enac,qo=>qos(1));
   ds(1)<= (qos(2) and (qos(3) and (qos(4) and (qos(5) and (qos(6) and (qos(7) and (qos(8) and (qos(9) and enac))))))))
xor qos(1);

ffd0: ffd port map(rsti=>rstc,clki=>clkc,di=>ds(0),enai=>enac,qo=>qos(0));
   ds(0)<= (qos(1) and (qos(2) and (qos(3) and (qos(4) and (qos(5) and (qos(6) and (qos(7) and (qos(8) and (qos(9) and
enac))))))))) xor qos(0);
qoc <= qos;
end;
```

Vidon

```
entity vidon is
port(
        clkh_vidon: in bit;
        clkv_vidon: in bit;
        pixel_y_vidon: in bit_vector (0 to 9);
        pixel_x_vidon: in bit_vector (0 to 9);
        salida_vidon: out bit:='1';
        v_vidon: out bit
);
end;

architecture test of vidon is
        ----------------------------------------------
        component compxigual
                port(
                        a: in bit_vector (9 downto 0);
                        b: in bit_vector (9 downto 0);
                        s: out bit
                        );
        end component;
        ----------------------------------------------
        component ffd
                port (
                        clki: in bit;
                        rsti: in bit;
                        enai: in bit;
                        di: in bit;
                        qo: out bit
                        );
        end component;
        ----------------------------------------------

signal flag_v_vidon_up, flag_v_vidon_down, flag_h_vidon_up, flag_h_vidon_down: bit;
signal salida_hvidon, salida_vvidon: bit;
signal flag_h_vidon_up1,flag_h_vidon_up2: bit;
signal flag_v_vidon_up1,flag_v_vidon_up2: bit;

begin
        flag_h_vidon_up <= flag_h_vidon_up1 or flag_h_vidon_up2; --hvidon se activa cuando esta en 0 o en 799
        flag_v_vidon_up <= flag_v_vidon_up1 or flag_v_vidon_up2; --hvidon se activa cuando esta en 0 o en 525
        -----------COMPORTAMIENTO DE HVIDON---------------------
        hvidonup1: compxigual port map (a=>pixel_x_vidon, b=>"0000000000", s=>flag_h_vidon_up1);
        hvidonup2: compxigual port map (a=>pixel_x_vidon, b=>"1100011111", s=>flag_h_vidon_up2); --799
        hvidondown: compxigual port map (a=>pixel_x_vidon, b=>"1010000000", s=>flag_h_vidon_down); --640

        ffdhvidon: ffd port map (clki=>clkh_vidon,rsti=>flag_h_vidon_down, enai=> flag_h_vidon_up, di=>'1',
qo=>salida_hvidon);
        -------------------------------------------------------

        ----------COMPORTAMIENTO DE VVIDON----------------------
        vvidonup1: compxigual port map (a=>pixel_y_vidon, b=>"0000000000", s=>flag_v_vidon_up1);
```

```
	vvidonup2: compxigual port map (a=>pixel_y_vidon, b=>"1000001101", s=>flag_v_vidon_up2); --525
	vvidondown: compxigual port map (a=>pixel_y_vidon, b=>"0111100001", s=>flag_v_vidon_down); --481

	ffdvvidon: ffd port map (clki=>clkv_vidon,rsti=>flag_v_vidon_down, enai=> flag_v_vidon_up, di=>'1',
qo=>salida_vvidon);


	salida_vidon <= salida_hvidon and salida_vvidon;
	v_vidon<=salida_vvidon;
end;
```

Sincronismo Vertical

```
entity vsync is
port(
	clkvsync: in bit;
	bits_contador_vsync: in bit_vector(0 to 9);
	salidavsync: out bit);
end;

architecture test of vsync is
	--DECLARACION DE COMPONENETES--
	component compxigual
		port(
			a: in bit_vector (0 to 9);
			b: in bit_vector (0 to 9);
			s: out bit);
		end component;
	------------------------------
	component ffd
		port (
			clki: in bit;
			rsti: in bit;
			enai: in bit;
			di: in bit;
			qo: out bit);
		end component;
	------------------------------
	--DECLARACION DE SEÑALES------
signal salida_syncup, salida_syncdown: bit;

begin

	vsyncup: compxigual port map(a=>bits_contador_vsync, b=>"0111101010",s=>salida_syncup);

	vsyncdown: compxigual port map(a=>bits_contador_vsync, b=>"0111101100",s=>salida_syncdown);

	ffdvsync: ffd port map(clki=>clkvsync,rsti=>salida_syncdown,enai=>salida_syncup,di=>'1',qo=>salidavsync);
end;
```

Sincronismo Horizontal

```
entity hsync is
port(
		clkhsync: in bit;
		bits_contador_hsync: in  bit_vector(0 to 9);
		salidahsync: out bit);
end;

architecture test of hsync is
		--DECLARACION DE COMPONENETES--
		component compxigual
				port(
						a: in bit_vector (0 to 9);
						b: in bit_vector (0 to 9);
						s: out bit);
				end component;
		------------------------------
		component ffd
				port (
						clki: in bit;
						rsti: in bit;
						enai: in bit;
						di: in bit;
						qo: out bit);
				end component;
		------------------------------
		--DECLARACION DE SEÑALES------
signal salida_syncup, salida_syncdown: bit;

begin

		hsyncup: compxigual port map(a=>bits_contador_hsync, b=>"1010001111",s=>salida_syncup);

		hsyncdown: compxigual port map(a=>bits_contador_hsync, b=>"1011100111",s=>salida_syncdown);

		ffdhsync: ffd port map(clki=>clkhsync,rsti=>salida_syncdown,enai=>salida_syncup,di=>'1',qo=>salidahsync);
end;
```