UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Ingeniería Electrónica
Técnicas Digitales 1

Examen Final
22/12/2021

Apellido, nombre	Legajo	Cantidad de hojas	Nota

Problema 1 (3 puntos)
Implemente en VHDL un circuito con la entidad que se muestra a continuación:

```
entity problema1 is
generic(N : natural := 8);                          -- Cantidad de bits.
port ( clk: in  std_logic;                          -- senial de reloj.
       rst: in  std_logic;                          -- rst asincronico.
       njr: in  std_logic;                          -- entrada de configuracion.
       q  : out std_logic_vector (N-1 downto 0));   -- salida.

);
end problema1;
```

El funcionamiento es el siguiente. Cuando njr es igual a uno el contador cuenta como un contador en anillo de N bits, si está en cero cuenta como un contador Johnson de N bits. Se pide:

a) Implemente en VHDL.
b) Explicite el valor inicial que le da al contador y por qué.

Problema 2 (4 puntos)
Dada la siguiente entidad de la que se pretende que en salida ponga la cantidad de unos que tiene la señal entrada.

```
entity bitCnt is
port ( e: in  std_logic_vector (3 downto 0);
       s: out std_logic_vector (2 downto 0));
end bitCnt;
```

Se pide:

a) De la tabla de verdad para la entidad **`bitCnt`** e implemente con multiplexores de 3 entradas de selección.
b) Utilizando la entidad anterior dada como un bloque de RTL, sumadores, multiplexores y lógica que considere necesaria implemente un diagrama RTL para un contador de unos para una palabra de entrada (**`X`**) de 12 bits generando una salida (**`Y`**) de la cantidad de bits necesaria para contar bien en todos los casos.
c) Implemente en VHDL el punto anterior, utilizando sentencias concurrentes únicamente. Utilice instancias de la entidad **`bitCnt`** dada para contar unos de palabras de 4 bits.

Problema 3 (3 puntos)
La sucesión de Fibonacci está dada por las siguientes condiciones iniciales y la ecuación $f_0 = 0,\ f_1 = 1,\ f_n = f_{n-1} + f_{n-2}$

Se pide:

a) Implementar el diagrama RTL utilizando registros, sumadores, multiplexores y demás bloques y lógica que considere necesario para generar la sucesión de Fibonacci en N bits, con la siguiente interfaz:
   i) Una entrada **`clk`** para la señal de reloj activa en flanco ascendente.
   ii) Una entrada **`rst`** para reset sincrónico.
   iii) Una salida **`q`** de N bits que genera un nuevo valor de la sucesión por cada ciclo de reloj.
   iv) Una salida **`tc`** se debe poner en uno cuando se alcanza el máximo valor que se puede generar de la sucesión para los N bits de la salida, luego de esto el circuito debe reiniciar la cuenta.
b) Implemente en VHDL basado en el RTL del punto anterior.

PROBLEMA 1

a)
```
entity problema1 is
  generic (N: NATURAL := 8);
  port (clu, rst, njr: in std_logic;
        q: out std_logic_vector (N-1 downto 0));
end problema1;

architecture arch_problema1 of problema1 is
signal qaux: std_logic_vector (N-1 downto 0);
pro: process (clu, rst)   -- En lista de sensibilidad
                          -- RST al ser asincrónico
begin
  if rst = '1' then
    qaux <= "00000001";
  elsif njr = '1' then
    for i in 0 to N-1 loop
      if i < N-1 then
        q_aux (i+1) <= q_aux (i);
      elsif i = N-1 then
        q_aux (0) <= q_aux (i);
      end if;
    end loop;
  else
    for i in 0 to N-1 loop
      if i < N-1 then
        q_aux (i+1) <= q_aux (i);
      elsif i = N-1 then
        q_aux (0) <= NOT (q_aux (N-1));
      end if;
    end loop;
  end if;
end process;
  q <= qaux;
end arch_problema1;
```

b) Motivo de estar a "00000001"

Necesito que haya un '1' en el circuito para que vaya siendo desplazado entre los distintos FLIP-FLOPS, ya que de lo contrario entraría en estados inutilizables para ambos tipos de contadores.

PROBLEMA 2

LEANDRO RODRÍGUEZ | 22-12-2021

PROBLEMA 2

a)

e_3	e_2	e_1	e_0	S_2	S_1	S_0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	0
0	1	1	1	0	1	1
1	0	0	0	0	0	1
1	0	0	1	0	1	0
1	0	1	0	0	1	0
1	0	1	1	0	1	1
1	1	0	0	0	1	0
1	1	0	1	0	1	1
1	1	1	0	0	1	1
1	1	1	1	1	0	0

Utilizo e_3, e_2 y e_1 como señales de control

S_2:

$e_3 e_2 e_1$
"000" => $S_2 = '0'$
"001" => $S_2 = '0'$
"010" => $S_2 = '0'$
"011" => $S_2 = '0'$
"100" => $S_2 = '1'$
"101" => $S_2 = '1'$
"110" => $S_2 = '1'$
"111" => $S_2 = e_0$

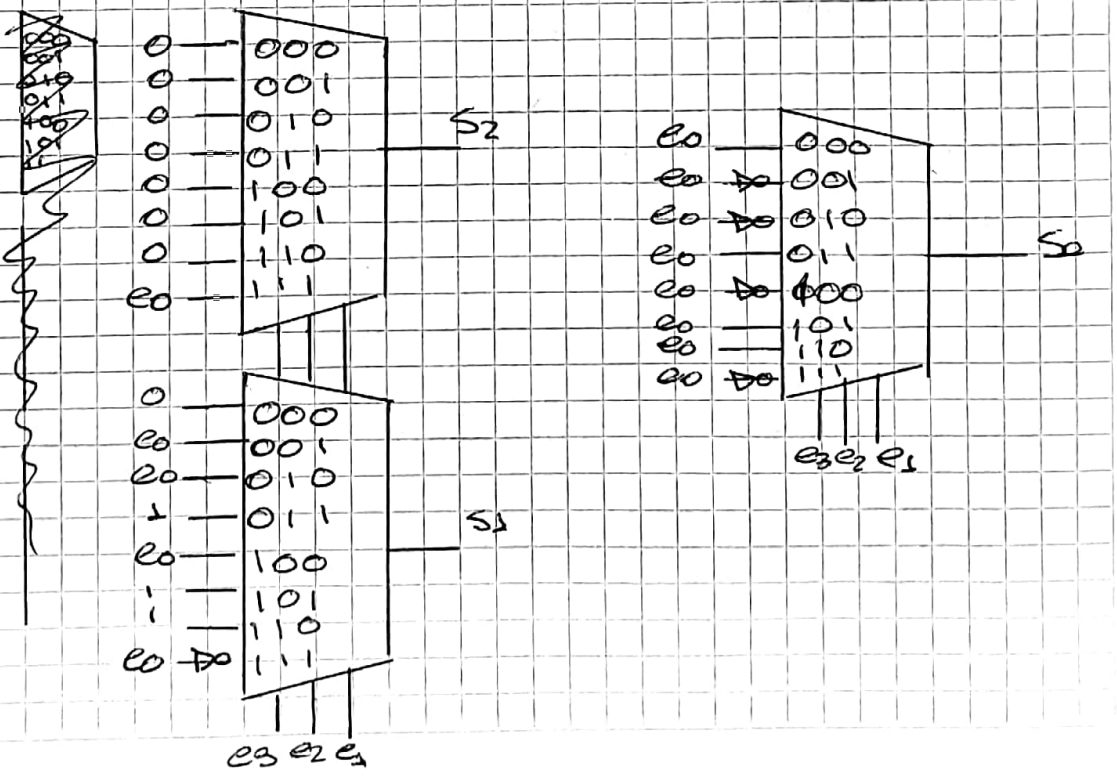
Por simple inspección de la tabla

S_1:

e_3	e_2	e_1	S_1
0	0	0	0
0	0	1	e_0
0	1	0	e_0
0	1	1	1
1	0	0	e_0
1	0	1	1
1	1	0	1
1	1	1	$\overline{e_0}$

S_0:

e_3	e_2	e_1	S_0
0	0	0	e_0
0	0	1	$\overline{e_0}$
0	1	0	e_0
0	1	1	$\overline{e_0}$
1	0	0	e_0
1	0	1	$\overline{e_0}$
1	1	0	e_0
1	1	1	$\overline{e_0}$

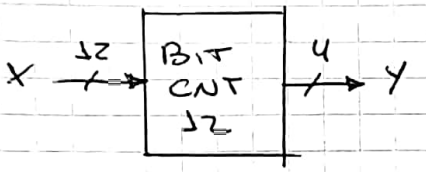


b)

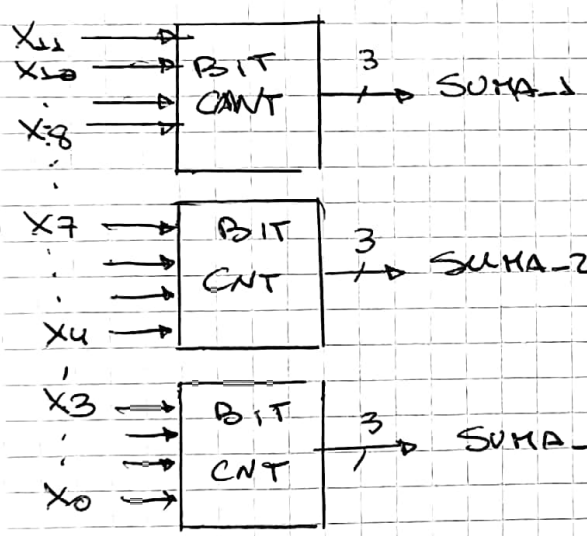
↳ Bloque a utilizar

Si tengo que contar los unos de una palabra de 12 bits, la máxima salida va a ser de 4 bits: 1100

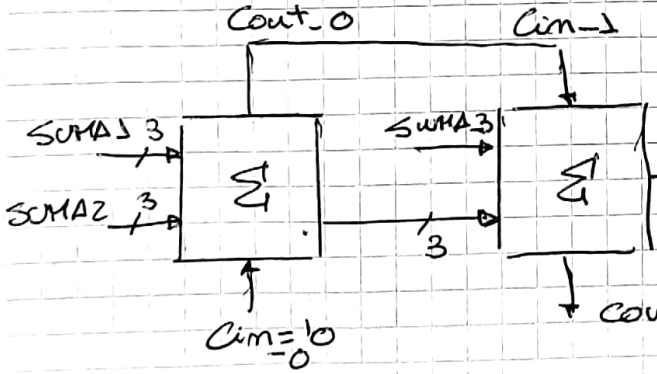
↳ Cant máxima



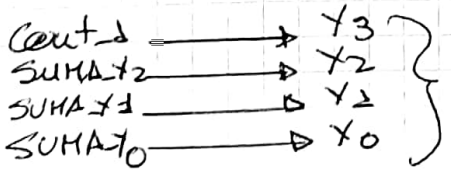
Utilizo 3 bloques del punto a, separando la entrada X en 3.



Estos 3 señales de suma las uso para 2 sumadores de 3 bits



Para obtener Y, concateno Cout_1 con SUMA_Y



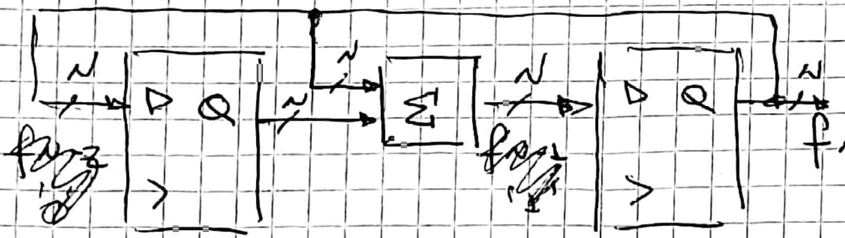
PROBLEMA 2

```
use IEEE.Library IEEE;

d) entity bitCnt is
port (e: in std_logic_vector (3 downto 0);
      s: out std_logic_vector (2 downto 0));
end bitCnt;

architecture arch_bitCnt of bitCnt is
  signal ctrl : std_logic_vector (2 downto 0);
begin

  ctrl <= e(3) & e(2) & e(1);

  with ctrl select
    s2 <= e(0) when "111",
          '0' when others;

  with ctrl select
    s1 <= '0' when "000",
          e(0) when "001",
          e(0) when "010",
          '1' when "011",
          e(0) when "100",
          '1' when "101",
          '1' when "110",
          not(e(0)) when "111",
          '0' when others;

  with ctrl select
    s0 <= e0 when ("000" or "011" or "101" or "110"),
          not(e(0)) when ("001" or "010" or "100" or "111"),
          '0' when others;

end arch_bitCnt
```

```
library IEEE;
use IEEE.STD_LOGIC_ALL;   use IEEE.NUMERIC_STD.ALL;

entity bitCnt12 is
port (x: in std_logic_vector (11 downto 0);
      Y: out std_logic_vector (3 downto 0));
end bitCnt12;

architecture arch_bitCnt12 of bitCnt12 is
signal SUMA1, SUMA2, SUMA3 : std_logic_vector (3 downto 0);
begin
SUMA1(3) <= '0';
SUMA2(3) <= '0';
SUMA3(3) <= '0';

inc0 : entity work.bitCnt
       port map (x(11 dt 8) => e,
                 SUMA1(2 dt 0) => s);

inc1 : entity work.bitCnt
       port map (x(7 dt 4) => e,
                 SUMA2(2 dt 0) => s);

inc2 : entity work.bitCnt
       port map (x(3 dt 0) => e,
                 SUMA3(2 dt 0) => s);

Y <= STD_LOGIC_VECTOR (unsigned(SUMA1) + unsigned(SUMA2)
                       + unsigned(SUMA3));

end arch_bitCnt12;
```

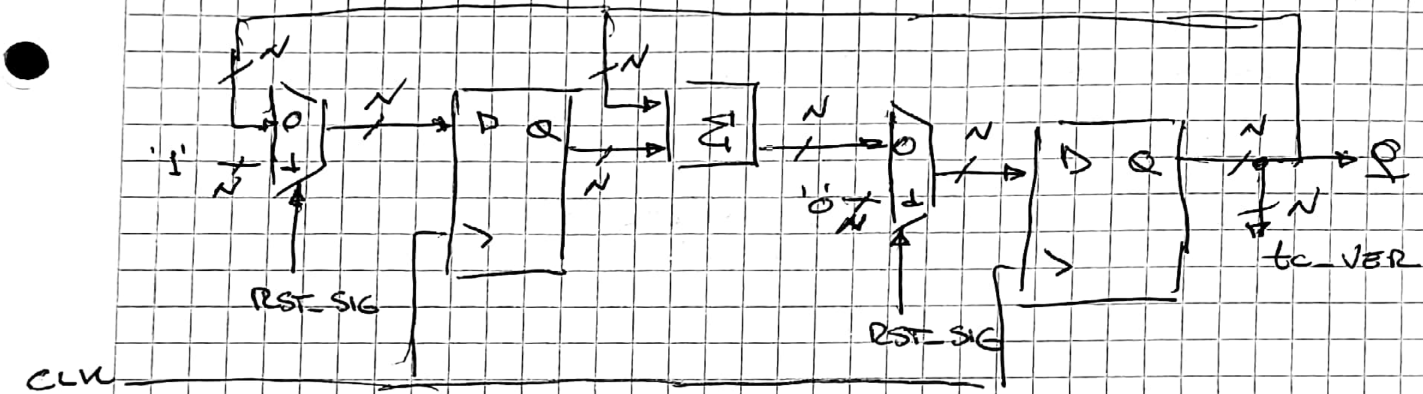
PROBLEMA 3

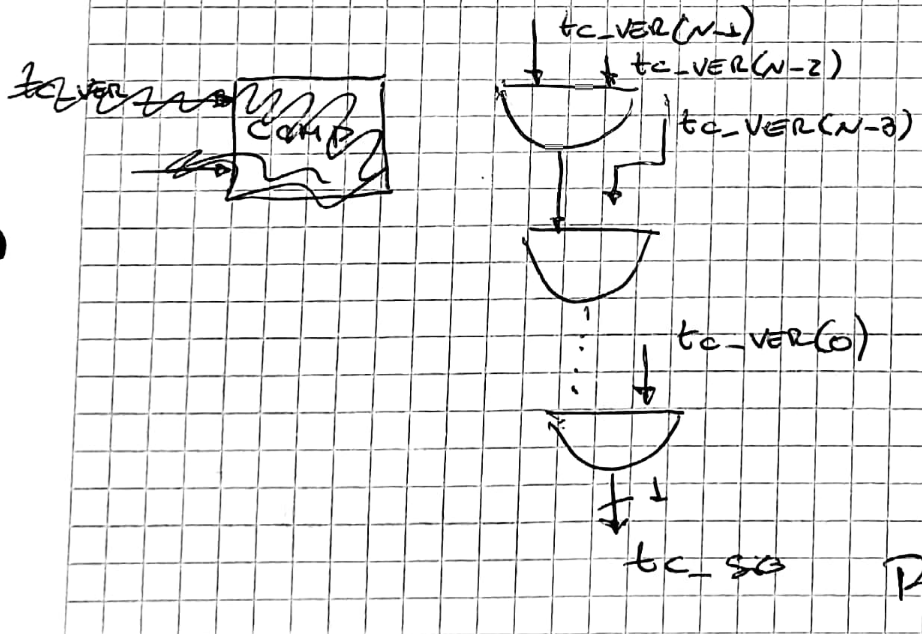
a)



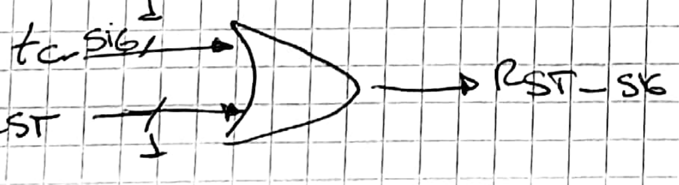
Circuito ya enunciado

Tengo que agregar condición de reset, tanto inicial como por comparación





Verifico si llegó al máximo (Podría también usar bloque comparador)



Reinicio por tc o por RST

tc_SIG — tc

b) entity fib is
port (clk, rst : in std_logic;
f : out std_logic_vector (N-1 downto 0);
tc : out std_logic);

end fib

architecture arch_fib of fib is
signal fnow1, fnow2, fnext1, fnext2 : std_logic_vector (N-1 downto 0);
signal rst_sig : std_logic;
begin

pro : process (clk)
begin
if rising_edge (clk) then
if rst_sig = '1' then
fnow1 <= (0 => '1', others => '0');
fnow2 <= (others => '0');

else
fnow1 <= fnext1;
fnow2 <= fnext2;

end if;
end if;

end process;

f_next1 <= fnow2;
f_next2 <= s_l_v (unsigned(fnow2) + unsigned(fnow1));
rst_sig <= '1' when (RST = '1' OR tc_sig = '1');

f <= fnow2;

-- Falta obtener tc_sig