ED. Aritmética binaria. Marzo 2019. V. 4.

Dadas las cantidades A y B, expresadas en **complemento a dos (Ca2)**:

A = 10101010 $_{Ca2}$

B = 00001111 $_{Ca2}$

Se pide:

1) (20%) Convierta a base decimal (dec) con signo A y B:

A = ... dec

B = ... dec

2) (10%) Calcule C en Ca2, siendo:

C = A + B = ... Ca2

3) (10%) Calcule D en Ca2, siendo:

D = A - B = ... Ca2

4) (10%) En una **suma aritmética binaria con signo**, expresada en **Ca2**, considerando que **Z = X + Y**, atendiendo únicamente al bit de signo de X, Y y Z, describa la(s) **condición(es) de Desbordamiento u Overflow (Ov)**:

.....

.....

5) (25%) Complete la **asignación condicionada concurrente** para la **detección de Overflow Ov** (con **Ov = 1**), para el siguiente código VHDL:

```
Library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

Entity SUMA is
port (A,B: in signed(7 downto 0);
C:   out signed(7 downto 0);
Ov:  out std_logic  );
End SUMA;

Architecture FUNCIONALIDAD of SUMA is
Signal resultado: signed(7 downto 0);
Begin
resultado <= A + B;
C  <= resultado;
Ov  <=  .....  When  .....  Else  .....  ;

End FUNCIONALIDAD;
```

6) (25%) Analice y complete el siguiente código VHDL para poder calcular en E el complemento a dos (Ca2) de una cantidad D: **E = Ca2 (D); siendo E y D de 8 bits.**

```
Library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;


Entity COMPDOS  is
port (
            D:  ……..  signed(……..……..);
            E:  ……. signed(……..……..)
);
End COMPDOS  ;

Architecture FUNCIONALIDAD of COMPDOS  is
Begin
……

End FUNCIONALIDAD;
```

Soluciones:

Dadas las cantidades A y B, expresadas en complemento a dos (Ca2):

A = 10101010 $_{Ca2}$

B = 00001111 $_{Ca2}$

Se pide:

1) (20%) Convierta a base decimal (dec) con signo A y B:

A = **-86** dec

B = **+15** dec

2) (10%) Calcule **C** en Ca2, siendo:

C = A + B = 10111001 Ca2

3) (10%) Calcule **D** en Ca2, siendo:

D = A - B = 10011011 Ca2

4) (10%) En una **suma aritmética binaria con signo**, expresada en **Ca2**, considerando que **Z = X + Y**, atendiendo únicamente al bit de signo de X, Y y Z, describa la(s) **condición(es) de Desbordamiento u Overflow (Ov)**:

Si los dos operandos X e Y tienen el mismo signo y el resultado de la operación Z tiene signo diferente el resultado presenta Overflow (no es válido): Ov = 1.

5) (25%) Complete la **asignación condicionada concurrente** para la **detección de Overflow Ov** (con **Ov = 1**), para el siguiente código VHDL:

```
Library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

Entity SUMA is
port (A,B: in signed(7 downto 0);
C:   out signed(7 downto 0);
Ov:  out std_logic  );
End SUMA;

Architecture FUNCIONALIDAD of SUMA is
Signal resultado: signed(7 downto 0);
Begin
resultado <= A + B;
C  <= resultado;
Ov <= '1' When (A(7)='1' and B(7)='1' and resultado(7)='0') or (A(7)='0' and B(7)='0' and resultado(7)='1') Else '0';
End FUNCIONALIDAD;
```

6) (25%) Analice y complete el siguiente código VHDL para poder calcular en E el complemento a dos (Ca2) de una cantidad D: **E = Ca2 (D); siendo E y D de 8 bits.**

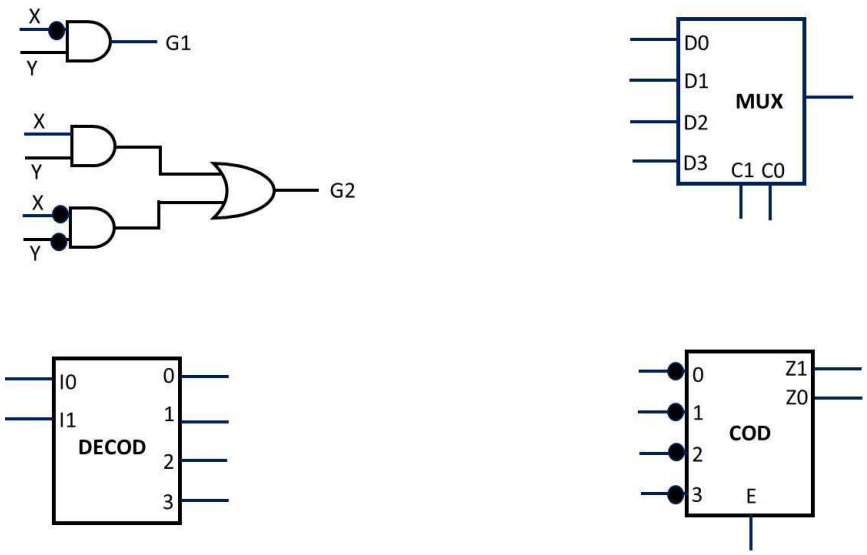
```
Library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

Entity COMPDOS  is
port (
    D: in signed(7 downto 0);
    E:  out signed(7 downto 0)
    );
End COMPDOS  ;

Architecture FUNCIONALIDAD of COMPDOS  is
Begin
    E  <= (not D) + 1;          -- también:  E  <= -D;
End FUNCIONALIDAD;
```

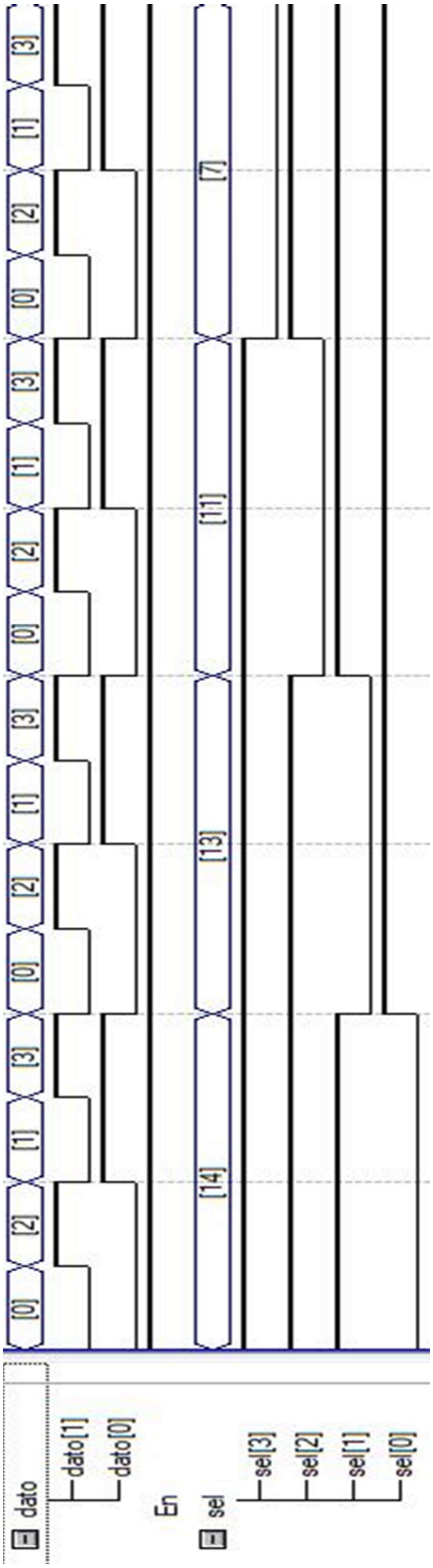
Problema 2 (30 minutos – 3,5 puntos)

Un circuito digital contiene los bloques que se muestran en la figura:



Se pide:

1. (**10%**) Tabla de verdad del bloque codificador (COD) y del bloque decodificador (DECOD).
2. (**30%**) Identifique los bloques en el código VHDL, señale las líneas de código en las que se describe cada uno de los elementos de la figura. Tenga en cuenta que no hay correspondencia de nombres entre la figura y el código.
3. (**30%**) A partir del código VHDL conecte correctamente los elementos del circuito. Puede unir los elementos por nombre sin necesidad de conectarlos con líneas. Indique en el dibujo los nombres utilizados en el código VHDL.
4. (**30%**) Complete el siguiente cronograma, añadiendo la señal de salida F del código vhdl.


[3]
[1]
[2]
[0]
[3]
[1]
[2]
[0]
[3]
[1]
[2]
[0]
[3]
[1]
[2]
[0]
[7]
[11]
[13]
[14]
dato
dato[1]
dato[0]
En
sel
sel[3]
sel[2]
sel[1]
sel[0]

CÓDIGO VHDL

```
library ieee;
use ieee.std_logic_1164.all;

ENTITY circuito IS
    PORT( dato: IN STD_LOGIC_VECTOR(1 DOWNTO 0);
          sel:IN STD_LOGIC_VECTOR(3 DOWNTO 0);
          En:IN STD_LOGIC;
          F: OUT STD_LOGIC);
END circuito;

ARCHITECTURE func OF circuito IS
SIGNAL sal: STD_LOGIC_VECTOR(1 DOWNTO 0);
SIGNAL sal2: STD_LOGIC_VECTOR(3 DOWNTO 0);

BEGIN

--DECODIFICADOR----------------------------------------------------------------------------

    process(dato)
    begin
        case dato is
            when "00" => sal2<="0001";
            when "01" => sal2<="0010";
            when "10" => sal2<="0100";
            when "11" => sal2<="1000";
            when others => sal2<="0000";
        end case;
    end process;


--MULTIPLEXOR-----------------------------------------------------------------------------

    process(sal,dato,sal2)
    begin
        case sal is
            when "00" => F<= (not(dato(1))and dato(0));    --G1--------------------
            when "01" => F<=((not dato(1))and (not dato(0)))or (dato(1) and
dato(0));    --G2-----------------------------------------------------------------------------
```

```
                    when "10" => F<=sal2(0);
                    when "11" => F<=sal2(3);
                    when others => F<='0';
                end case;
end process;

    --CODIFICADOR----------------------------------------------------------------------------

    process(sel,En)
    begin
                if En='1' then
                    case sel is
                        when "1110" => sal<="00";
                        when "1101" => sal<="01";
                        when "1011" => sal<="10";
                        when "0111" => sal<="11";
                        when others =>sal<="00";
                    end case;
                else
                    sal<="00";
                end if;
    end process;

END func;
```

SOLUCIÓN

1- Tabla de verdad codificador (0,5/1)
(Entradas activas a nivel bajo, con entrada de habilitación)

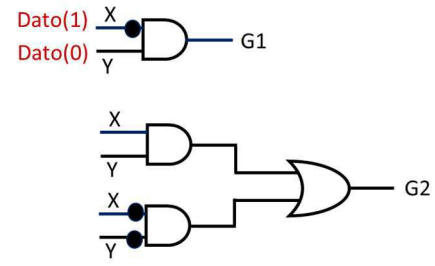
E	3	2	1	0	Z1	Z0
0	X	X	X	X	0	0
1	0	1	1	1	1	1
1	1	0	1	1	1	0
1	1	1	0	1	0	1
1	1	1	1	0	0	0
1	Resto de combinaciones				0	0

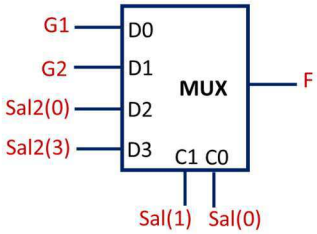
Tabla de verdad decodificador (0,5/1)

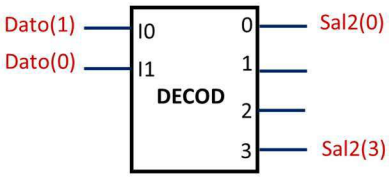
I1	I0	3	2	1	0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

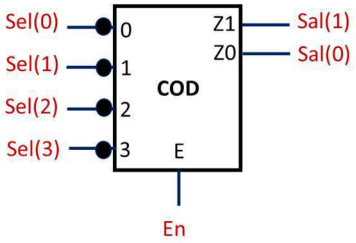
2- Contestado sobre el código:
Codificador: 1/3
Decodificador: 1/3
Multiplexor: 0,6/3
G1 y G2: 0,4/3

3-
Codificador: 1/3
Decodificador: 0,5/3
Multiplexor: 0,5/3
G1 y G2: 1/3

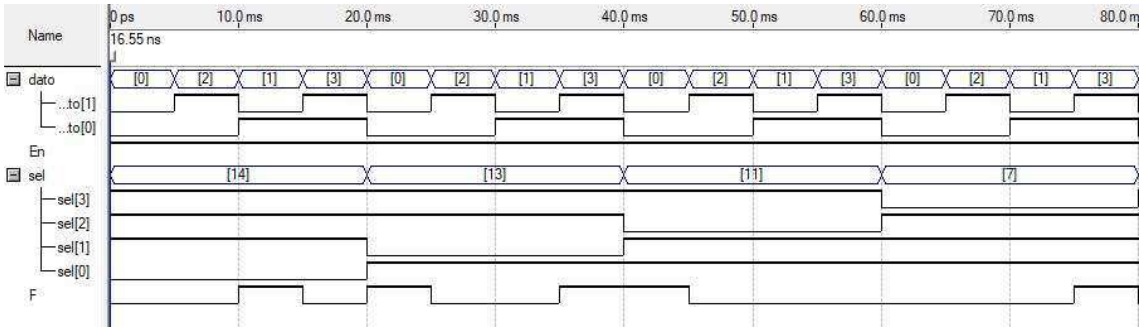

Dato(1)
Dato(0)
X
Y
G1
X
Y
X
Y
G2


G1
G2
Sal2(0)
Sal2(3)
D0
D1
D2
D3
MUX
F
C1 C0
Sal(1)
Sal(0)


Dato(1)
Dato(0)
I0
I1
DECOD
0
1
2
3
Sal2(0)
Sal2(3)


Sel(0)
Sel(1)
Sel(2)
Sel(3)
0
1
2
3
COD
Z1
Z0
E
Sal(1)
Sal(0)
En

4-


Name
0 ps
10.0 ms
20.0 ms
30.0 ms
40.0 ms
50.0 ms
60.0 ms
70.0 ms
80.0 m
16.55 ns
dato
...to[1]
...to[0]
En
sel
sel[3]
sel[2]
sel[1]
sel[0]
F
[14]
[13]
[11]
[7]

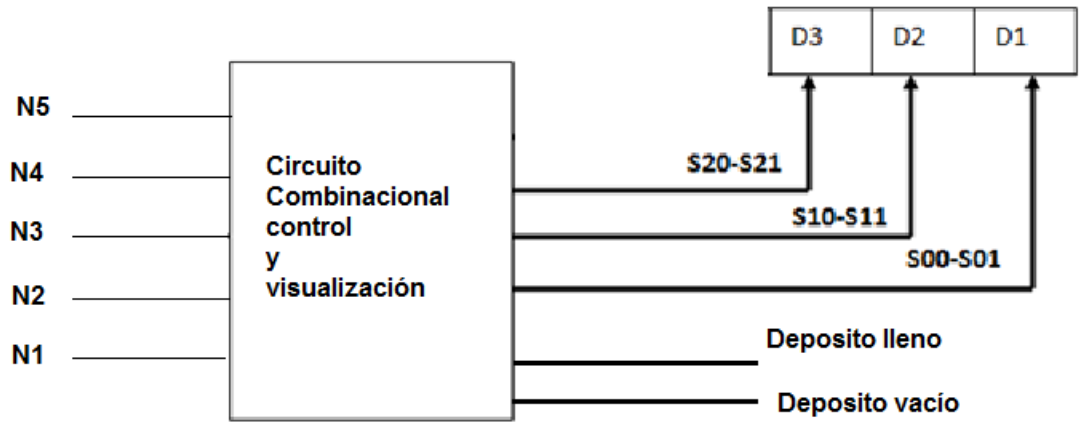
Problema 3 (30 minutos – 4 puntos) SOLUCIÓN

Se desea diseñar un circuito digital que muestra el nivel de agua de un tanque en números romanos. El circuito leerá la salida de 5 sensores (N5, N4, N3, N2, N1) que se activarán con un uno lógico cuando el agua supere su posición y en caso contrario suministrarán un cero lógico. Para mostrar el nivel de agua detectado por los sensores se utilizan 3 displays que muestran los números romanos de 1 a 5 (I, II, III, IV y V).
Valores imposibles de los sensores del tanque no mostrarán ningún valor en los displays.
El circuito además activará dos señales de alarma bajo determinadas circunstancias:

- Señal Lleno, cuando el nivel de agua sea máximo. Se activa a nivel alto.
- Señal Vacío, cuando el nivel de agua sea inferior al sensor 1. Se activa a nivel alto.

Cada uno de los 3 displays se controla con dos bits (SX1,SX2, siendo X el número del display utilizado) tal y como se muestra a continuación:

	SX1	SX0
OFF	0	0
I	0	1
V	1	1



Según lo indicado anteriormente se pide:

1. Complete la siguiente tabla de verdad del sistema completo (**10%**)

Nivel	N5	N4	N3	N2	N1	S21	S20	S11	S10	S01	S00	D_lleno	D_vacío
vacio	0	0	0	0	0	0	0	0	0	0	0	0	1
I	0	0	0	0	1	0	0	0	0	0	1	0	0
II	0	0	0	1	1	0	0	0	1	0	1	0	0
III	0	0	1	1	1	0	1	0	1	0	1	0	0
IV	0	1	1	1	1	0	0	0	1	1	1	0	0
V	1	1	1	1	1	0	0	0	0	1	1	1	0

2. Complete la entidad que resulta del sistema digital. (**10%**)

```
ENTITY nivel_R IS
        PORT (
                N5,N4,N3,N2,N1:    IN OUT STD_LOGIC;
                dep_lleno, dep_vacio: OUT STD_LOGIC;
                S2: OUT STD_LOGIC_VECTOR(1  DOWNTO   0); --control del display 2
                S1: OUT STD_LOGIC_VECTOR( 1 DOWNTO   0);--control del display 1
                S0: OUT STD_LOGIC_VECTOR( 1 DOWNTO   0 ));--control del display 0
END nivel_R;
```

3. Definir la arquitectura del sistema expuesto haciendo uso de sentencias secuenciales. Indica de forma clara la lista de sensibilidad necesaria. (**40%**)

```
--problema tipo examen
--indicador de nivel de líquidos N5__N1
--ver 1.0 5/13/19
--UC3M JMC


library  ieee;

use ieee.std_logic_1164.all;


entity pro_exa is

    port (
        N5,N4,N3,N2,N1: in std_logic;
        dep_lleno,dep_vacio: out std_logic;
        S2: out std_logic_vector( 1 downto 0 );
        S1: out std_logic_vector( 1 downto 0 );
        S0: out  std_logic_vector( 1 downto 0 )
         );

end pro_exa;
```

```
architecture funcional of pro_exa is

--declarar señales concatenadas
signal entrada: std_logic_vector ( 4 downto 0);
signal salida_temp: std_logic_vector (5 downto 0);

begin

entrada <= N5&N4&N3&N2&N1; --el orden es importante ya que N5 es el DMP

    process(entrada)
    begin
        case entrada is
        when "00000"    => salida_temp<="000000";       --apagado todos
        when "00001"    => salida_temp<="000001";   -- I
        when "00011"    => salida_temp<="000101";   -- II
        when "00111"    => salida_temp<="010101";   -- III
        when "01111"    => salida_temp<="000111";   -- IV
        when "11111"    => salida_temp<="000011";   -- V   dep_lleno<='1';

        when others => salida_temp<="ZZZZZZ";
        end case;
```

```
49          if (entrada="00000") then dep_vacio<='1';
50
51          else dep_vacio<='0';
52          end if;
53
54          if (entrada="11111") then dep_lleno<='1';
55
56          else dep_lleno<='0';
57          end if;
58
59      end process;
60      --actualizamos el valor de la salida
61      S2 <= salida_temp(5 downto 4);
62      S1 <= salida_temp(3 downto 2);
63      S0 <= salida_temp(1 downto 0);
64
65  end funcional;
66
```

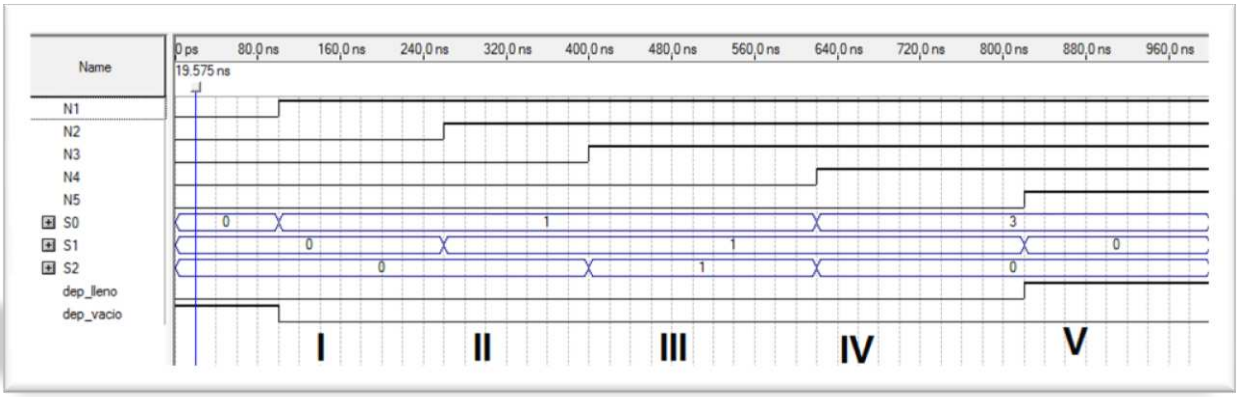
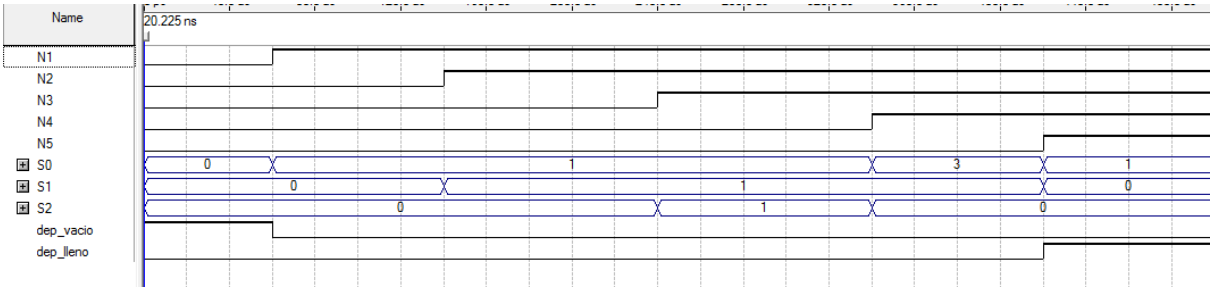


Ilustración 1. Simulación Secuencial

4. Definir la arquitectura del sistema expuesto haciendo uso de sentencias concurrentes. (**40%)**

```
1  --problema tipo examen con
2  --secuencias condicionales
3  --indicador de nivel de liquidos N5__N1
4  --ver 1.0 5/13/19
5  --UC3M JMC
6
7  library  ieee;
8  use ieee.std_logic_1164.all;
9
10
11 entity pro_exa_conc is
12
13     port (
14         N5,N4,N3,N2,N1: in std_logic;
15         dep_lleno,dep_vacio: out std_logic;
16         S2: out std_logic_vector( 1 downto 0 );
17         S1: out std_logic_vector( 1 downto 0 );
18         S0: out  std_logic_vector( 1 downto 0 )
19
20          );
21
22 end pro_exa_conc;
23
24 architecture funcional of pro_exa_conc is
25
26 --declarar señales concatenadas
27 signal entrada: std_logic_vector ( 4 downto 0);
28 signal salida: std_logic_vector (5 downto 0);
29
30
31 begin
32
33 entrada <= N5&N4&N3&N2&N1; --el orden es importante ya que N5 es el DMP
34
35 S2(1)<= '0';
36 S2(0)<= '1' when entrada="00111" else '0';
37
38 S1(1)<= '0';
39 S1(0)<='1' when entrada="00011" or entrada="00111" or entrada="01111" else '0';
40
41 S0(1)<='1' when entrada="01111" else '0';
42
43 S0(0)<='1' when entrada="00001"or entrada="00011" or entrada="00111" or entrada="01111" or entrada="11111" else '0';
44
45 dep_lleno <= '1' when entrada="11111" else '0';
46
47 dep_vacio <= '1' when entrada="00000" else '0';
48
49 end funcional;
```



Posiblemente aparezca otra solución, por ejemplo con WITH XXXXX SELECT;

```
1  --problema tipo examen con
2  --secuencias condicionales
3  --indicador de nivel de líquidos N5__N1
4  --usamos la sentencia with..... sel
5  --ver 1.0 5/13/19
6  --UC3M JMC
7
8  library  ieee;
9  use ieee.std_logic_1164.all;
10
11
12 entity pro_exa_conc is
13
14     port (
15         N5,N4,N3,N2,N1: in std_logic;
16         dep_lleno,dep_vacio: out std_logic;
17         --S2: out std_logic_vector( 1 downto 0 );
18         --S1: out std_logic_vector( 1 downto 0 );
19         --S0: out  std_logic_vector( 1 downto 0 )
20          salida: OUT STD_LOGIC_VECTOR(5 downto 0) --para usar salida directamente
21          );
22
23 end pro_exa_conc;
```

```
25 architecture funcional of pro_exa_conc is
26
27  --declarar señales concatenadas
28  signal entrada: std_logic_vector ( 4 downto 0);
29
30
31 begin
32
33  entrada <= N5&N4&N3&N2&N1; --el orden es importante ya que N5 es el DMP
34
35  with entrada select
36
37      salida <=   "000001" when "00001",
38                  "000101" when "00011",
39                  "010101" when "00111",
40                  "000111" when "01111",
41                  "000011" when "11111",
42                  "ZZZZZZ" when OTHERS;
43
44  dep_vacio <= '1' when entrada="00000" else '0';
45  dep_lleno <= '1' when entrada="11111" else '0';
46  end funcional;
```