1.1.d) $F1(A, B, C) = A\,\overline{B}\,C + \overline{B}\,\overline{C} + \overline{A}\,B\,C + B\,\overline{C}$

$F2(A, B, C) = A\,\overline{B}\,C + B\,\overline{C}$

$F3(A, B, C) = \overline{A}\,B\,C + \overline{B}\,\overline{C}$

```
library ieee;
use ieee.std_logic_1164.all;
entity pagina1_1_d is
port ( A,B,C: in std_logic;
       F1,F2,F3: out std_logic );
end pagina1_1_d;
```

```
architecture uno of pagina1_1_d is

signal NA,NB,NC: std_logic;
signal P1,P2,P3,P4: std_logic;

begin

NA <= not A;
NB <= not B;
NC <= not C;
P1 <= A and NB and C;
P2 <= NB and NC;
P3 <= NA and B and C;
P4 <= B and NC;
F1 <= P1 or P2 or P3 or P4;
F2 <= P1 or P4;
F3 <= P3 or P2;

end uno;
```

2.1.d)

$F1(A, B, C) = \sum(0, 2, 3, 4, 5) + \sum\emptyset(6)$

$F2(A, B, C) = \sum(5, 6) + \sum\emptyset(1, 2)$

$F3(A, B, C) = \sum(3, 4) + \sum\emptyset(0, 1)$

```
library ieee;
use ieee.std_logic_1164.all;
entity pagina2_d is
port ( A,B,C: in std_logic;
       F1,F2,F3: out std_logic );
end pagina2_d;
```

```
architecture uno of pagina2_d is
begin
process (A,B,C)
variable ABC: std_logic_vector(3 downto 1);
variable F123: std_logic_vector(3 downto 1);
begin
ABC := A & B & C;
-- Las salidas son F1 F2 F3
case ABC is
   when "000" => F123 := "10-";   -- m0
   when "001" => F123 := "0--";   -- m1
   when "010" => F123 := "1-0";   -- m2
   when "011" => F123 := "101";   -- m3
   when "100" => F123 := "101";   -- m4
   when "101" => F123 := "110";   -- m5
   when "110" => F123 := "-10";   -- m6
   when "111" => F123 := "000";   -- m7
end case;
F1 <= F123(3);
F2 <= F123(2);
F3 <= F123(1);
end process;
end uno;
```

3.1. Se desea diseñar un circuito lógico para determinar el vencedor de un combate entre dos contendientes X e Y mediante las siguientes especificaciones:

- El combate será a tres toques. El vencedor se declara cuando uno o los dos contendientes llegue a tres toques (se permite la posibilidad de toque simultáneo), o se llegue al llegar al final del tiempo de combate. El número de toques (de 0 a 3) realizado por cada contendiente se almacena en binario en dos variables lógicas para X (x1x0) y dos para Y (y1y0).
- Al finalizar el combate se declara vencedor al contendiente que haya realizado más toques. En caso de empate el combate se dilucida por la decisión de un árbitro (variable lógica A) que declara vencedor a X (A a valor 1) o a Y (A a valor 0). El árbitro no puede declarar el combate empatado.

Realizar una descripción VHDL que permita obtener el ganador del combate. Se deben considerar las siguientes entradas: T que determina si se ha llegado o no al final del tiempo de combate, nX y nY que indica el número de toques realizado por cada contendiente y A que contiene la decisión del árbitro. Se deben utilizar dos salidas Gx y Gy que indican el vencedor, Gx y Gy están a 0 hasta que se determine si ha vencido X (Gx a 1) o Y (Gy a 1) en función del valor de las entradas.

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity pagina3_1 is
port (T,A: in std_logic;
      nX,nY: in std_logic_vector(2 downto 1);
      GX,GY: out std_logic := '0');
end pagina3_1;
```

```vhdl
architecture uno of pagina3_1 is

begin
process (nX,nY,A,T)
begin
if ( T = '1' or nX = 3 or nY = 3 ) then  -- Fin de combate
   if ( nX > nY ) then                   -- Gana X por toques
      GX <= '1'; GY <= '0';
   elsif ( nX < nY ) then                -- Gana Y por toques
      GX <= '0'; GY <= '1';
   elsif ( A = '1' ) then                -- Gana X por el arbitro
      GX <= '1'; GY <= '0';
   else                                  -- Gana Y por el arbitro
      GX <= '0'; GY <= '1';
   end if;
else                        -- Combate sin finalizar: no hay ganador
   GX <= '0'; GY <= '0';
end if;
end process;

end uno;
```

4.1. Realizar la descripción VHDL de quiere realizar un circuito de 4 entradas (I4-I1) que muestre como resultado 4 salidas (O4-O1), tal que la salida muestra la entrada pero eliminando todos los unos menos el más significativo. Por ejemplo, si I = “0110” => O = “0100”; si I = “0001” => O = “0001”, etc. Si todos los bits de la entrada son 0, los de la salida también: I = “0000” => O = “0000”.

```
library ieee;
use ieee.std_logic_1164.all;
entity pagina4_1 is
port (I: in std_logic_vector(4 downto 1);
      O: out std_logic_vector(4 downto 1) );
end pagina4_1;
```

```
architecture uno of pagina4_1 is
begin
process (I)
begin
if ( I(4) = '1' ) then
   O <= "1000";
elsif ( I(3) = '1' ) then
   O <= "0100";
elsif ( I(2) = '1' ) then
   O <= "0010";
elsif ( I(1) = '1' ) then
   O <= "0001";
else
   O <= "0000";
end if;
end process;
end uno;
```

4.2. En una competición hay cuatro jueces que puntúan con valores enteros entre 0 y 3. Se quiere determinar cuando el promedio de la puntuación es mayor que 1.5. Realizar el código VHDL para la descripción de un circuito digital que resuelva este problema.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity pagina4_2 is
port (J1,J2,J3,J4: in std_logic_vector(2 downto 1);
      MM: out std_logic);
end pagina4_2;
```

```
architecture uno of pagina4_2 is
begin
process(J1,J2,J3,J4)
variable suma: std_logic_vector(4 downto 1);
begin
suma := ("00" & J1) + ("00" & J2) + ("00" & J3) + ("00" & J4);
if ( suma > "0110" ) then
   MM <= '1';
else
   MM <= '0';
end if;
end process;
end uno;
```

4.3. Realizar la descripción VHDL de un circuito que realice la suma dos operandos A y B de 5 bits en notación con bits de signo: 4 bits de módulo y un bit de signo (a 0 positivo, a 1 negativo). El resultado Z debe ser de 6 bits: 5 bits de módulo y un bit de signo. Una forma de hacer esto es comprobando si los signos de A y B son iguales, en ese caso el signo de Z es el signo de los operandos y su módulo la suma de los dos módulos. Si los signos son distintos, el signo de Z será el signo del operando de mayor módulo, y el módulo de Z la resta del mayor módulo menos el menor módulo.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity pagina4_3 is
port ( A, B: in std_logic_vector(5 downto 1);
       Z: out std_logic_vector(6 downto 1) );
end pagina4_3;
```

4.3. Realizar la descripción VHDL de un circuito que realice la suma dos operandos A y B de 5 bits en notación con bits de signo: 4 bits de módulo y un bit de signo (a 0 positivo, a 1 negativo). El resultado Z debe ser de 6 bits: 5 bits de módulo y un bit de signo. Una forma de hacer esto es comprobando si los signos de A y B son iguales, en ese caso el signo de Z es el signo de los operandos y su módulo la suma de los dos módulos. Si los signos son distintos, el signo de Z será el signo del operando de mayor módulo, y el módulo de Z la resta del mayor módulo menos el menor módulo.

```
architecture uno of pagina4_3 is
begin
process (A,B)
begin
-- Mismo signo
if ( A(5) = B(5) ) then
   Z(6) <= A(5);
   Z(5 downto 1) <= ('0' & A(4 downto 1)) + ('0' & B(4 downto 1));
-- Distinto signo, mismo modulo
elsif ( A(4 downto 1) = B(4 downto 1) ) then
   Z(6) <= '0';
   Z(5 downto 1) <= "00000";
-- Distinto signo, modulo de A mayor que el de B
elsif ( A(4 downto 1) > B(4 downto 1) ) then
   Z(6) <= A(5);
   Z(5 downto 1) <= ('0' & A(4 downto 1)) - ('0' & B(4 downto 1));
else  -- Distinto signo, modulo de B mayor que el de A
   Z(6) <= B(5);
   Z(5 downto 1) <= ('0' & B(4 downto 1)) - ('0' & A(4 downto 1));
end if;
end process;
end uno;
```

5.1. Realizar el código VHDL que permita convertir una entrada X de 5 bits que representa un número binario sin signo en dos salidas que representen el número en notación decimal: D de 2 bits que representa las decenas del número y U de 4 bits que representa las unidades del número. Hay que usar sentencias estándar VHDL (case, if-else, etc) sin recurrir a funciones de librerías, aunque sí se pueden usar operadores.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity pagina5_1 is
port ( X: in std_logic_vector(5 downto 1);
       D: out std_logic_vector(2 downto 1);
       U: out std_logic_vector(4 downto 1) );
end pagina5_1;
```

5.1. Realizar el código VHDL que permita convertir una entrada X de 5 bits que representa un número binario sin signo en dos salidas que representen el número en notación decimal: D de 2 bits que representa las decenas del número y U de 4 bits que representa las unidades del número. Hay que usar sentencias estándar VHDL (case, if-else, etc) sin recurrir a funciones de librerías, aunque sí se pueden usar operadores.

```
architecture uno of pagina5_1 is
begin
process(X)
variable R: std_logic_vector(5 downto 1);
begin
-- X mayor o igual que 30: D = 3; U = X -30
if ( X >= "11110" ) then      -- o X >= 30
   D <= "11";
   R := X - "11110"; -- o R := X - 30;
-- X menor que 30 y mayor o igual que 20: D = 2; U = X - 20
elsif ( X >= "10100" ) then  -- o X >= 20
   D <= "10";
   R := X - "10100"; -- o R := X - 20;
-- X menor que 20 y mayor o igual que 10: D = 1; U = X - 10;
elsif ( X >= "01010" ) then  -- o X >= 10
   D <= "01";
   R := X - "01010"; -- o R := X - 10;
else    -- X menor que 10 : D = 0; U = X;
   D <= "00";
   R := X;
end if;
U <= R(4 downto 1);  -- Toma los 4 bits menos significativos
end process;
end uno;
```

5.2. Desarrollar el código VHDL de un circuito que comprueba si en una entrada A de N bits (por defecto N es 4) hay más unos, más ceros o igual número de unos y ceros.

```
library ieee;
use ieee.std_logic_1164.all;
entity pagina5_2 is
generic( N: integer := 4);
port ( A: in std_logic_vector(N downto 1);
       MU,MC,I: out std_logic );
end pagina5_2;
```

```
architecture uno of pagina5_2 is
begin
process(A)
variable numc,numu: integer range 0 to N; -- Numeros de ceros y unos
begin
numc := 0; numu := 0;    -- Inicializacion de los numeros
for i in 1 to N loop     -- Encuentra el numero de ceros y de unos
   if ( A(i) = '0' ) then numc := numc + 1;
   else numu := numu + 1;
   end if;
end loop;

if ( numc = numu ) then -- Numeros de ceros y unos iguales
   I <= '1'; MC <= '0'; MU <= '0';
elsif ( numc > numu ) then  -- Numero de ceros mayor que el de unos
   I <= '0'; MC <= '1'; MU <= '0';
else  -- Numero de ceros menor que el de unos
   I <= '0'; MC <= '0'; MU <= '1';
end if;
end process;
end uno;
```

5.3. Desarrollar el código VHDL (sin desarrollar la tabla de verdad) de un circuito lógico que permite realizar en su salida Z de 1 bit las operaciones de comparación mayor (A > B), igual (A = B) o menor (A < B) sobre dos entradas de datos A y B de N bits (por defecto N es 4). La operación se selecciona mediante dos entradas de control S1 y S0 con valores 00 (igual), 01 (mayor), 10 (menor), respectivamente, teniendo en cuenta que, cuando S1 y S0 son 1 simultáneamente, la salida Z se debe fijar a 0.

```
library ieee;
use ieee.std_logic_1164.all;
entity pagina5_3 is
generic( N: integer := 4);
port ( A,B: in std_logic_vector(N downto 1);
       S1,S0: in std_logic;
       Z: out std_logic );
end pagina5_3;
```

5.3. Desarrollar el código VHDL (sin desarrollar la tabla de verdad) de un circuito lógico que permite realizar en su salida Z de 1 bit las operaciones de comparación mayor (A > B), igual (A = B) o menor (A < B) sobre dos entradas de datos A y B de N bits (por defecto N es 4). La operación se selecciona mediante dos entradas de control S1 y S0 con valores 00 (igual), 01 (mayor), 10 (menor), respectivamente, teniendo en cuenta que, cuando S1 y S0 son 1 simultáneamente, la salida Z se debe fijar a 0.

```
architecture uno of pagina5_3 is
begin
process(A,B,S1,S0)
variable S: std_logic_vector(2 downto 1);
begin
S := S1 & S0;
case S is
   when "00" => if ( A = B ) then Z <= '1';
                else Z <= '0';
                end if;
   when "01" => if ( A > B ) then Z <= '1';
                else Z <= '0';
                end if;
   when "10" => if ( A < B ) then Z <= '1';
                else Z <= '0';
                end if;
   when others => Z <= '0';
end case;
end process;
end uno;
```

6.1. Un sistema activa una luz artificial L en función de las medidas de cuatro sensores de luz natural que activan o desactivan unas señales lógicas A, B, C y D según el nivel de luz sea menor o mayor de una valor umbral dado, y una señal de control horario H que permite calibrar de dos maneras distintas la influencia de los sensores. En función de los sensores se obtiene un nivel final de luz dado por

$$N = PA \cdot A + PB \cdot B + PC \cdot C + PD \cdot D,$$

donde PA, PB, PC y PD es el peso de cada sensor.

En función de la señal H los pesos PA, PB, PC y PD son respectivamente 3, 5, 7, 2 cuando se usa el primer control horario (H = 0), y son 6, 4, 1, 4 cuando se usa el segundo (H = 1).

L se activa cuando N toma los valores 2, 4, 5, 7, 11 y 17, y se desactiva cuando N toma los valores 3, 6, 8, 9, 12, 13 y 15. Se desea obtener un circuito que implemente L = F(H, A, B, C, D).

Realizar una descripción VHDL según el enunciado del problema.

```
library ieee;
use ieee.std_logic_1164.all;
entity pagina6_1 is
port ( A,B,C,D,H: in std_logic;
       L: out std_logic );
end pagina6_1;
```

```
architecture uno of pagina6_1 is
begin
process(A,B,C,D,H)
variable PA,PB,PC,PD: integer range 1 to 7;
variable N: integer range 0 to 17;
begin
if ( H = '0' ) then   -- Calculo de pesos segun H
   PA := 3; PB := 5; PC := 7; PD := 2;
else
   PA := 6; PB := 4; PC := 1; PD := 4;
end if;
N := 0;  -- Calculo de N
if ( A = '1' ) then N := N + PA; end if;
if ( B = '1' ) then N := N + PB; end if;
if ( C = '1' ) then N := N + PC; end if;
if ( D = '1' ) then N := N + PD; end if;
-- Calculo de L
if ( N = 2 or N = 4 or N = 5 or N = 7 or N = 11 or N = 17 ) then
   L <= '1';
elsif ( N = 3 or N = 6 or N = 8 or N = 9 or N = 12 or N = 13
        or N = 15 ) then
   L <= '0';
else
   L <= '-';
end if;
end process;
end uno;
```