Arquitectura de Computadoras I
Práctico Nº 0: Introducción VHDL (Parte I)

1) Determine si el código VHDL (cod. 1) describe el comportamiento de la función ***s = OR (a(2), a(1), a(0)).*** Modifique el contenido del proceso *PrOR* para que describa la función requerida en el caso que el código no deseado.

```
entity  nombre_entidad is
    port (    a: in std_logic_vector(2 downto 0);
              s: out std_logic);
end nombre_entidad;

architecture nombre_arquitectura of nombre_entidad is
signal i: std_logic_vector(3 downto 0);
begin

        PrOR:process (a)
        begin
            i(0) <= '0';
            for j in 1 to 3 loop
                    i(j) <= i(j-1) or a(j-1);
            end loop;
            s <= i(3);
        end process;

end nombre_arquitectura;
```

cod. 1

2) Realice la descripción de un multiplexor de 2 entradas de un bit. Utilice:

i) Procesos explícitos y sentencias secuenciales.
ii) Sentencias concurrentes
iii) Instanciación de componentes. Use componente ***or(a,b)*** y ***and(a,b)***

3) Realice la descripción de un demultiplexor de dos salidas de un bit. Utilice:

i) Procesos explícitos y sentencias secuenciales.
ii) Sentencias concurrentes
iii) Instanciación de componentes. Use componente ***or(a,b)***, ***and(a,b)*** y ***not(a)***

4) Realice la descripción VHDL de un multiplexor de 4 entradas de un bit.

5) Realice ejercicios 2) 3) y 4) trabajando con operandos de 4 bits.

6) Realice un sumador completo (*FA*) de un bit.

7) Diseñe un circuito que realice la comparación entre dos operandos *a* y *b* de dos bits. el circuito posee dos salidas *g* y *e* de un bit. La salida *g* es 1 si *a>b* de lo contrario es 0 y *e* es 1 solo en el caso que *a = b*.

8) Instanciando el módulo realizado en 6), realice un sumador completo de 8 bits.

9) Determine si existen errores para cada una de las arquitecturas de la entidad *entity1*

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity  entity1 is
        port (    a: in std_logic_vector(3 downto 0);
                  b: in std_logic_vector(3 downto 0);
                  c: in std_logic;
                  d: out std_logic_vector(3 downto 0));
end entity1;


architecture arquitecture1 of entity1 is
begin
        d <= (a and b) when (c = '1' ) else d;
end;

architecture arquitecture2 of entity1 is
        signal x: std_logic_vector(3 downto 0);
begin
        x <= (a and b) when (c = '1');
        x <= (a xor b) when (c = '0');
        d <= x;
end;

architecture arquitecture3 of entity1 is
        signal x: std_logic_vector(3 downto 0);
begin
        process (a, b)
        begin
                x <= (a and b);
                x <= (a xor b);
        end process;
        d <= x;
end;

architecture arquitecture4 of entity1 is
begin
        d(1 downto 0) <= a(1 downto 0) or b(1 downto 0);
        d(3 downto 2) <= a(1 downto 0) + b(1 downto 0);
end;
```