Descripción de una AND de dos entradas

Nombre carpeta = Nombre del proyecto

Antes existía una limitación de 3 caracteres para la extensión. Se usaba .vhd

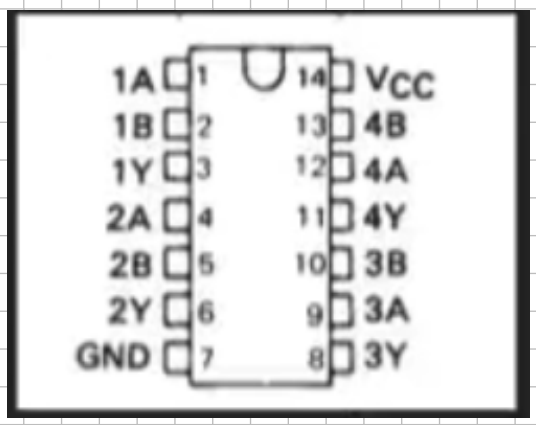
Documentar el archivo con comentarios

-- Fecha – Autor – Nombre Archivo
-- Explicación del archivo

Similar a lo que se hace en C, incluimos una biblioteca

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

Tenemos dos secciones, entity y architecture



Describimos las entradas y salidas en la Entidad

Para esto usamos la palabra reservada Port.

```
entity AND2ecu is
    Port ( a_i : in std_logic;
           b_i : in std_logic;
           y_o : in std_logic );
end entity AND2ecu;
```

se identifican las entradas por terminar en _i de las salidas _o

El ; separa elementos de una lista

Con la sección arquitectura se define la función que define a la salida.

```
architecture FlujoDeDatos of AND2ecu is begin
    Y_o <= a_i and b_i;
    -- Sentencia de la descripción
end architecture FlujoDeDatos;
```

Parte 2. Compilación y Simulación

Los warnings se deben a que no se eligió una placa de desarrollo.

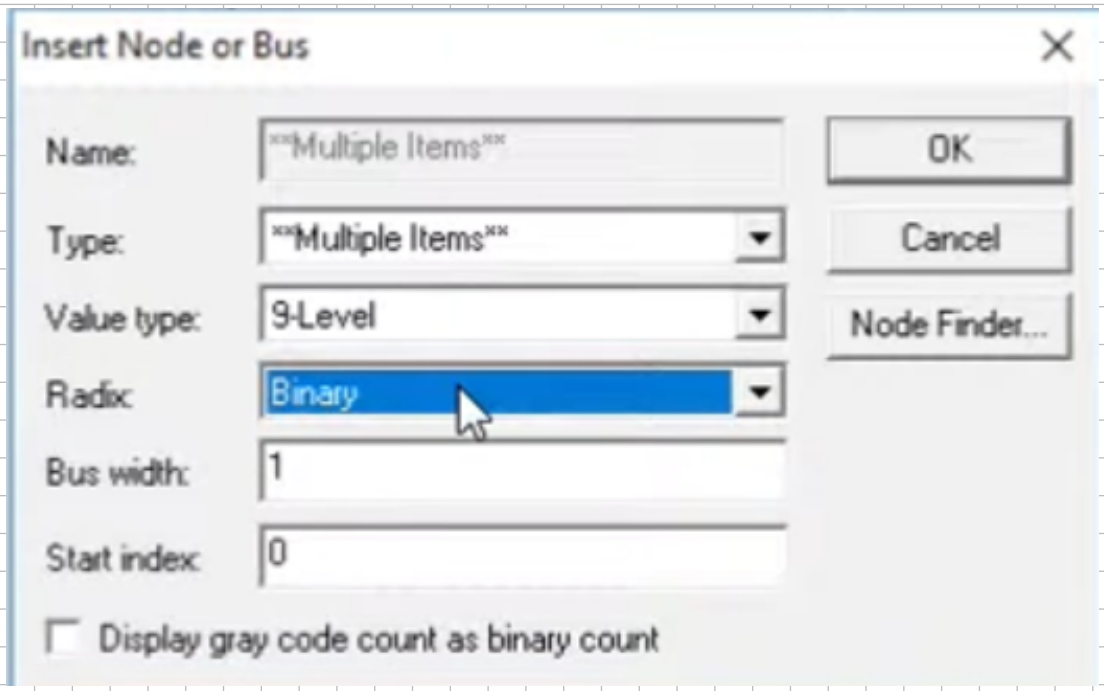
- Simulación mediante testbench

  los testbench son programas mientras que lo otro es descripción de hardware.

Quartus Web Edition es Free v.9.1 permite simular sin escribir el testbench

→ Vector Waveform File

Insertar los nombres de las entradas y salidas usando "Insert Node or Bus", listar los puertos y elegirlos al panel derecho



Descripcion usando when else

```
library ieee;
Use ieee.std_logic_1164.all;

entity OR2_when_else is
    port
    (
        -- input ports
        <name> : is <type>;

        -- output ports
        <name> : is <type>;

    );
end OR2_when_else;

architecture FlujoDeDatos of OR2_when_else is
begin
        Y_o <= '1' when b_i = '1' or a_i = '1' else '0'; ??
end architecture FlujoDeDato;
```

Son sentencias condicionadas que no son mutuamente excluyentes.

El uno lógico va entre comillas simples

Prioridad a las condiciones dadas según el orden en que se escriben.

Video 4. Descripción por tabla de verdad

OR

A	B	Z
0	0	0
0	1	1
1	0	1
1	1	1

```
architecture TablaDeVerdad of OR2_withselect is
    Signal entradas: std_logic_vector(1 downto 0)
begin

    entradas <= b_i & a_i;
    -- defino la tabla
    With entradas select
        Y_o <= '1' when "11",
               '0' when "01",
               '0' when "10",
               '0' when "00",
               '0' when others;
end architecture TablaDeVerdad
```

entre el is y begin
Se llama parte declarativa

entre el begin y en
Se llama cuerpo

Signal es como un cable y une procesos.

STD_logic_vector forma parte del package
STD_logic_1164.

STD_logic_vector(3 downto 0)

Vector dimensión 4

Para llamar al primer elemento
hay que referenciar a la señal
asociada con el vector y entre
paréntesis indicar su posición

entradas(0) es el primer elemento

El simbolo & concatena elementos

Y_o <=
↳ Proceso Implícito

la tabla deberia contener todas las
combinaciones posibles, por ejemplo
con '0' debil, '1' debil, etc
Entonces podemos usar when others

Los casos son mutuamente excluyentes entonces no existen las prioridades

Adecuado para tablas de verdad

para no escribir tantos when podemos usar la barra vertical |

```
y_o <= '1' when "11",
       'o' when "01"|"10"|"00",
       'o' when others;
```

Otra descripción por tabla de verdad

```
use ieee.numeric_std.all;

arch
    signal entradas: std_logic_vector(1 downto 0);
    constant COLUMNA: stdlogic_vector(0 to 3):="0001";
begin
    entradas <= b_i & a_i;
    y_o <= COLUMNA (to_integer(unsigned(entradas)));
end arch
```

Se castea un entero mediante el paquete numeric.