Facultad de Informática
Universidad Complutense de Madrid

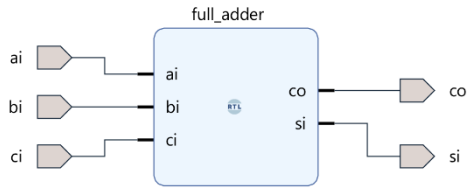
LABORATORIO DE FUNDAMENTOS DE COMPUTADORES 1

PRÁCTICA 2: DISEÑO Y SIMULACIÓN DE UN SUMADOR BINARIO DE 2 BITS MEDIANTE VHDL Y LA HERRAMIENTA MODELSIM

El objetivo de esta práctica es avanzar en el manejo de la herramienta de simulación de circuitos integrados *ModelSim* de *Mentor Graphics* y del lenguaje de descripción de hardware VHDL. Para ello se va a implementar y simular un sumador binario de 2 bits.

DISEÑO CON VHDL

En la siguiente figura se puede ver la caja negra que describe el interfaz del sumador total que vamos a implementar inicialmente:



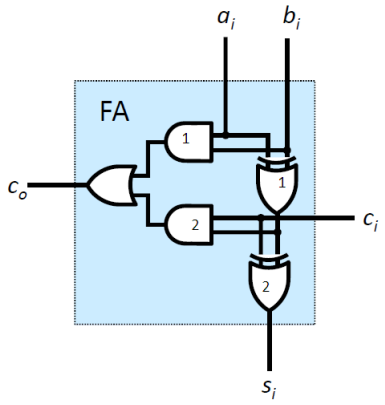
La declaración de entidad para esta caja negra es la siguiente:

```
entity sumador is
        port(ai, bi : in bit; ci : in bit; si : out bit; co : out bit);
end sumador;
```

Como se puede apreciar, el módulo tiene:

- Tres entradas de datos de tipo bit.
- Dos salidas, *co* y *si* de tipo bit.

En esta práctica vamos a realizar una descripción estructural de la arquitectura del sumador total utilizando las puertas lógicas and2, or2 y xor2 incluidas en el archivo *full_adder.vhd* que se proporciona al alumno. En concreto ese archivo implementa la siguiente red de puertas lógicas:



A las conexiones internas de esta red las vamos a nombrar de la siguiente manera:

- sal_and1: la conexión entre and1 y or
- sal_and2: conexión entre and2 y or
- sal_xor1 la conexión entre xor1 y xor2 y and2

Visto lo anterior, la arquitectura estructural del sumador total es la siguiente:

```
entity full_adder is
        port(ai, bi : in bit; ci : in bit; si : out bit; co : out bit);
end full_adder;

--use work.all;
architecture puertas of full_adder is
--declaración de componentes
 component or2
        port(i1, i2 : in bit; o : out bit);
 end component;
 component and2
        port(i1, i2 : in bit; o : out bit);
 end component;
 component xor2
         port(i1, i2 : in bit; o : out bit);
 end component;

 --declaración de señales internas
 signal sal_and1,sal_and2,sal_xor1: bit;
--empieza el cuerpo de la arquitectura
begin
 i_and1 : and2  port map(ai, bi, sal_and1);
 i_xor1 : xor2  port map(ai, bi, sal_xor1);
 i_and2 : and2  port map(sal_and1, ci, sal_and2);
 i_xor2 : xor2  port map(sal_xor1, ci, si);
 i_or   : or2   port map(sal_and1, sal_and2, co);
end puertas;
```

En este fragmento de código se pueden observar los siguientes elementos:

Después de la cabecera de la arquitectura aparece la declaración de los componentes que se van a utilizar en el diseño. La declaración de componentes sirve para establecer el tipo de diseños que se van a utilizar como elemento de construcción de otro diseño. En nuestro

caso se declaran tres componentes: la puerta lógica or2, la and2 y la xor2. Por ejemplo, el fragmento de código:

```
component or2
        port(i1, i2 : in bit; o : out bit);
end component;
```

declara el componente or2, definido por dos entradas y una salida, de tipo bit.

A continuación, se declaran las señales internas que conectan las diferentes puertas lógicas entre sí, en concreto se declaran las señales sal_and1, sal_and2 y sal_xor1.

```
signal sal_and1, sal_and2, sal_xor1 : bit;
```

Ya dentro del cuerpo de la arquitectura aparecen las instancias de las componentes que vamos a utilizar. Como hemos visto, un componente es el tipo de diseño que utilizamos como elemento de construcción, y una instancia es el elemento de construcción propiamente dicho. Si nos fijamos en la figura que contiene la red de puertas lógicas que implementa el sumador total, vemos que éste necesita puertas de tipo and2, or2 y xor2. Estos son los tipos de componentes que tiene que utilizar. En concreto necesita dos puertas and, dos puertas xor y una puerta or. Estas son las instancias de las componentes que se pueden ver en el fragmento de código.

Nota: Los componentes deben estar implementados previamente a su uso. En este laboratorio se le proporcionarán al alumno en el archivo *full_adder.vhd*.

DESARROLLO DE LA PRÁCTICA

1. Crear un proyecto que incluya el fichero full_adder.vhd proporcionado al alumno, compilarlo y simularlo comprobando que el sumador funciona correctamente.
2. Completar el archivo full_adder2_plantilla.vhd proporcionado, de manera que implemente un sumador binario de 2 bits mediante una arquitectura estructural que utilice el *full_adder* del apartado anterior como componente, siguiendo la estructura explicada en las trasparencias.