LABORATORIO DE ARQUITECTURA DE COMPUTADORES.

I. T. I. SISTEMAS / GESTIÓN

GUÍA DEL ALUMNO

Práctica 2: La Unidad Aritmético - Lógica

Objetivos

- Comprender cómo se realiza un sumador con propagación de acarreo
- Comprender como funciona la Unidad Aritmético - Lógica

Método de trabajo

Lee cuidadosamente las indicaciones de la guía, realizando los pasos que se indican. La guía tiene como propósito que **TÚ** te asegures de que comprendes lo que va sucediendo mientras realizas lo que indica que hagas.

Para todas las sesiones, necesitaremos textos de referencia del lenguaje para realizar programas o para comprender lo que ocurre durante las simulaciones. Por lo tanto se recomienda traer libros, apuntes y cualquier bibliografía de VHDL que sirva para hacer consultas. En el apartado "bibliografía" se proporcionan algunas referencias.

También traer un **disquete** o una **memoria USB** para llevar copia de los ficheros que se generen y se deseen conservar ya que, al tratarse de una máquina compartida, no hay garantías de que más tarde encontremos los trabajos que dejamos hoy. Pero dado que V-System no trabaja bien con los disquetes siempre habrá **que copiar de/al disco duro y trabajar en él.**

Bibliografía

- "VHDL: Lenguaje para síntesis y modelado de circuitos". F.Pardo y J.A.Boluda. Ed. Ra-Ma,1999.
- Lluis Terés y otros "VHDL: Lenguaje estándar de diseño electrónico" Mc Graw Hill, 1998
- Buscar en la WEB: Google encuentra referencias y pueden encontrarse libros en pdf

Primera parte

Introducción

Aunque esta práctica trata de la unidad Aritmético – Lógica en esta primera parte trataremos los sumadores ya que la suma es la operación más importante de todas, puesto que:

–Se emplea para el cálculo de la dirección de la siguiente instrucción

–Se utiliza para el cálculo de las direcciones a los operandos

–Otras operaciones la emplean: resta, multiplicación, división

Sumador Binario

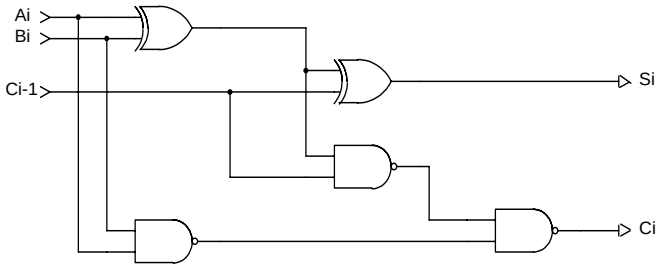
Si desarrollamos la tabla de la verdad de un sumador elemental tendríamos lo siguiente:

A	B	Ci-1		Ci	S	Carry estatus
0	0	0		0	0	Delete
0	0	1		0	1	Delete
0	1	0		0	1	Propagate
0	1	1		1	0	Propagate
1	0	0		0	1	Propagate
1	0	1		1	0	Propagate
1	1	0		1	0	Generate
1	1	1		1	1	Generate

De ahí obtendríamos las siguientes funciones

$$Si = Ai \oplus Bi \oplus Ci\text{-}1$$
$$Ci = Ai\ Bi + Bi\ Ci\text{-}1 + Ai\ Ci\text{-}1$$

Y el circuito del sumador elemental.



Utilizando como componentes el modelo de una puerta XOR y una puerta NAND tendríamos la siguiente implementación VHDL para el sumador elemental:

```
------------------------------ Modelo de un sumador elemental ----------------------------

ENTITY sum_elemental IS
        PORT (a,b,carry_in : IN bit; suma,carry_out : OUT bit);
END sum_elemental;

ARCHITECTURE estructural OF sum_elemental IS
-- Declaración de componentes
        COMPONENT xor2
                PORT (e1,e2 : IN bit; sal : OUT bit);
        END COMPONENT;
        COMPONENT nand2
                PORT (e1,e2 : IN bit; sal : OUT bit);
        END COMPONENT;
-- Declaración de señales
        SIGNAL s1,s2,s3 : bit;
-- Localización de las arquitecturas de los componentes
        FOR puerta1,puerta2 : xor2 USE ENTITY WORK.xor2(comportamiento);
        FOR OTHERS : nand2 USE ENTITY WORK.nand2(comportamiento);
-- Conexiones
BEGIN
        puerta1 : xor2 PORT MAP (a,b,s1);
        puerta2 : xor2 PORT MAP (s1,carry_in,suma);
        puerta3 : nand2 PORT MAP (a,b,s2);
        puerta4 : nand2 PORT MAP (s1,carry_in,s3);
        puerta5 : nand2 PORT MAP (s2,s3,carry_out);
END estructural;
```

```
------------------------------------ Modelo de una puerta XOR ----------------------------------

ENTITY xor2 IS
        GENERIC (retardo: TIME := 2 ns);
        PORT (e1,e2 : IN bit; sal : OUT bit);
END xor2;

ARCHITECTURE comportamiento OF xor2 IS
BEGIN
        PROCESS (e1,e2)
        BEGIN
                sal <= e1 XOR e2 AFTER retardo;
```

```
        END PROCESS;
END comportamiento;
```

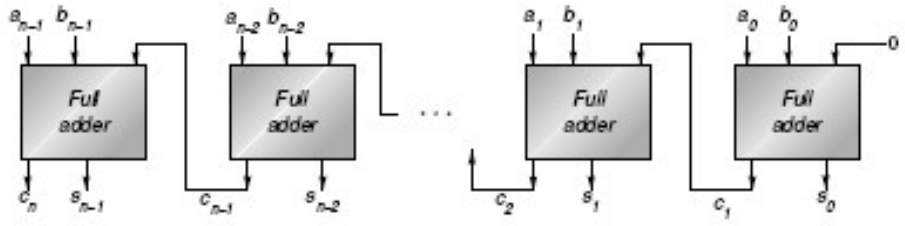
---------- Modelo de una puerta NAND ----------

```
ENTITY nand2 IS
        GENERIC (retardo : TIME := 2 ns);
        PORT (e1,e2 : IN bit; sal : OUT bit);
END nand2;

ARCHITECTURE comportamiento OF nand2 IS
BEGIN
        PROCESS (e1,e2)
        BEGIN
                sal <= e1 NAND e2 AFTER retardo;
        END PROCESS;
END comportamiento;
```

Sumador con propagación de acarreo

Poniendo en cascada n sumadores elementales tenemos un sumador de n bits con propagación de acarreo.



Como entrada de cada uno de los sumadores elementales (excepto del primero) tenemos el acarreo de salida del anterior lo que hace que haya que ir esperando un cierto retardo para cada uno de ellos que hace que tengamos la salida definitiva cuando tengamos C_n.

El tiempo de propagación es linealmente proporcional al número de bits, esto hace que sea un sumador muy lento y por ello se suelen utilizar mejoras de este sumador de propagación que intentan optimizar el t_{carry} por ejemplo definiendo señales intermedias G (generate), P (propagate) y D (delete).

Actividad 1.

En la práctica se pide:

1. Modelar mediante lenguaje VHDL un sumador con propagación de acarreo de 16 bits partiendo del modelo VHDL del sumador binario
   Se recomienda utilizar la función GENERATE.

2. Calcular el retardo para el caso más desfavorable.

3. Realizar un test-bench en el que se compruebe, entre otras opciones, el caso más desfavorable.

Segunda parte

Introducción

La Unidad aritmético-lógica (ALU): es la parte del computador que se encarga de procesar la información obteniendo resultados que dependen de la operación seleccionada.

Esta formada por:

–Operadores:aritméticos, lógicos y de desplazamiento

–Registros para almacenar datos temporales

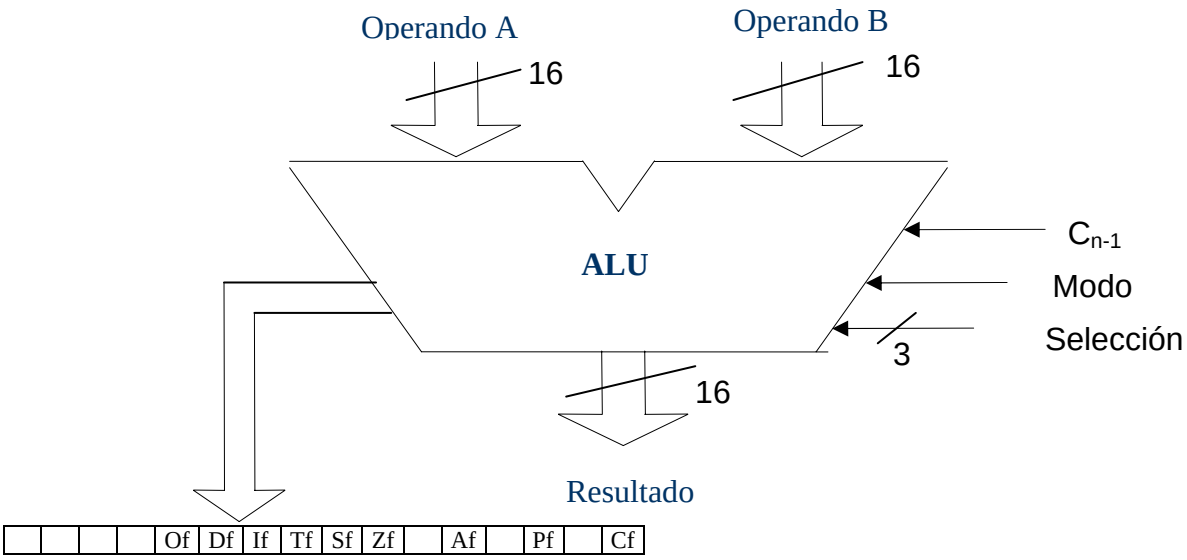
–Registro de estado: conjunto de flags que indican situaciones ocurridas al operar

–Registro contador de programa

–Registro de direcciones de interrupción

Unidad Aritmético-Lógica

Se desea realizar el modelo de la Unidad Aritmético-Lógica de la figura



Los operandos A y B son de 16 bits al igual que la salida resultado.

Las operaciones que se realizan se codifican según las entradas selección y modo tal y como podemos ver en la tabla:

Selección	Modo = 0 Op. Aritméticas	Modo = 1 Op. Lógicas
000	A+B	NOT A
001	A-B	A AND B
010	C2 de A	A OR B
011	Incrementar A	A XOR B
100	Decrementar A	Desplazamiento lógico a Izq.
101	A	Desplazamiento lógico a dcha.
110	B	Desplazamiento aritmético a Izq
111	A* B (sin signo)	Desplazamiento aritmético a dcha

Una de las salidas de la ALU servirá para modificar convenientemente el registro de estado.

Los flags que son modificados por la ALU son:

OF	Overflow	Indica que se ha producido Desbordamiento
SF	Signo	Indica el signo del resultado
ZF	Cero	Indica que el resultado es cero
AF	Auxiliar	Ajuste en operaciones BCD
PF	Paridad	Indica que en el resultado hay un nº par de unos
CF	Acarreo	Indica que se ha producido acarreo de salida

El resto de flags: DF (Dirección) IF(Interrupción) y TF (trap) son modificados por la unidad de control

Actividad 2.

1. Modelar en VHDL una ALU con las especificaciones dadas. Para simplificar el modelo se puede trabajar con un estilo comportamental para las distintas operaciones y supondremos que cualquiera de las operaciones tarda en realizarse 30ns. Por ser más sencillo trabajar a veces con números enteros, se recomienda utilizar las funciones del paquete ARQ_PACK que permite pasar de binario a entero y de entero a binario.

2. Realizar un test-bench en el que se compruebe el correcto funcionamiento del diseño.