Práctica 3

Módulos aritméticos I

1 Introducción

El diseño de ALUs que sean capaces de realizar cálculos a alta velocidad es fundamental dentro del diseño de la CPU. Con este objetivo, se plantea en esta práctica el diseño de un sumador que supere las prestaciones de la aproximación clásica del sumador de n bits construido a partir de n sumadores completos donde el retraso en la propagación de la señal será δ*n. En la figura 1 se muestra el peor caso en la propagación de la señal con un sumador de 4 bits construido con sumadores completos.

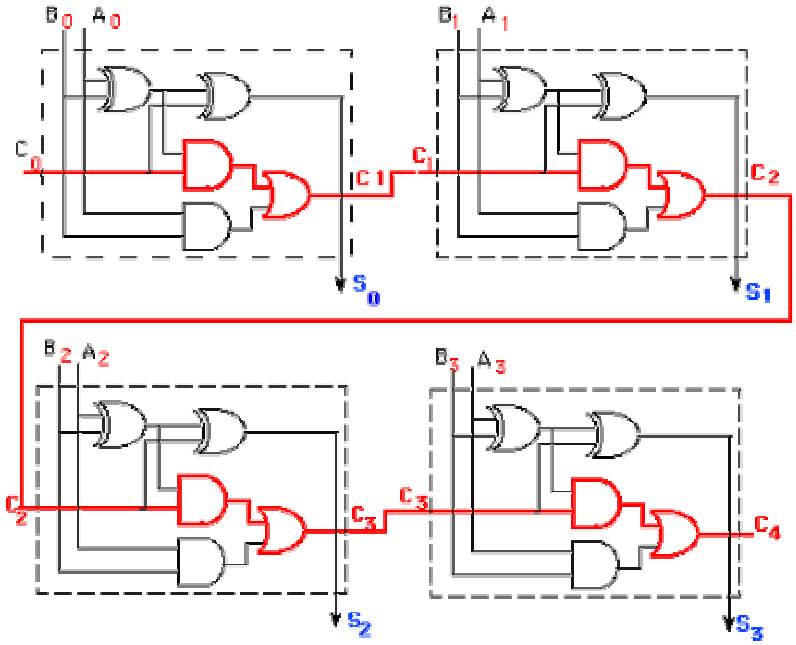

Figura 1: *Propagación de la señal con un sumador de 4 bits construido con sumadores completos.*

Una solución a este problema consiste en calcular de manera adelantada el acarreo de cada módulo de suma. Para simplificar estos cálculos se definen los términos Generación y Propagación de acarreo respectivamente como:

$$Gi=Ai \cdot Bi$$
$$Pi=Ai \oplus Bi$$

Se asume un retraso de 1 puerta para la función AND y de 2 puertas para la XOR. De esta manera los acarreos de cada bit en el caso del sumador de 4 bits pueden ser reescritos como:

$$C1=G0+P0 \cdot C0$$
$$C2=G1+P1 \cdot C1=G1+P1 \cdot G0+P1 \cdot P0 \cdot C0$$
$$C3=G2+P2 \cdot G1+P2 \cdot P1 \cdot G0+P2 \cdot P1 \cdot P0 \cdot C0$$
$$C4=G3+P3 \cdot G2+P3 \cdot P2 \cdot G1+P3 \cdot P2 \cdot P1 \cdot G0+P3 \cdot P2 \cdot P1 \cdot P0 \cdot C0$$

Cabe hacer notar como en el peor caso el acarreo se calculará con un retraso de 4 puertas (asumiendo el producto y la suma lógica con un retraso fijo independientemente del número de entradas debido a la implementación de AND y OR cableadas).

Con estas funciones las funciones suma se pueden construir como:

$$Si=Ai \oplus Bi \oplus Ci = Pi \oplus Ci$$

Lo que dará un retraso de dos puertas más debido a la función XOR.

Con esta política de cálculo de acarreos adelantados es posible dividir el diseño de un sumador de 4 bits construyendo dos tipos de módulos, tal como muestra la figura 2:

- El sumador parcial completo (en la figura 2 como PFA *Partial Full Adder*)
- La lógica de cálculo de acarreo adelantado.

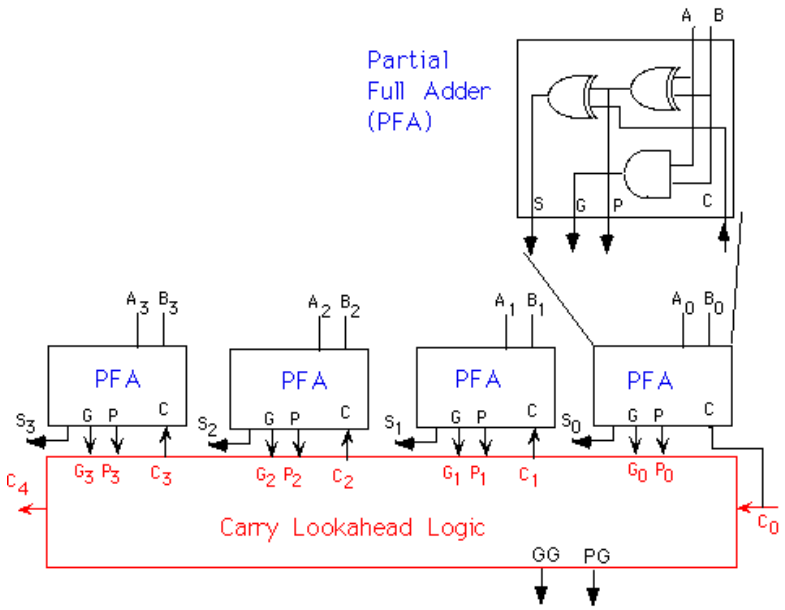

Figura 2: *Sumador de 4 bits con acarreo adelantado*

Las salidas P_G y G_G se definen como:

$$P_G = P3 \cdot P2 \cdot P1 \cdot P0$$
$$G_G = G3 + P3 \cdot G2 + P3 \cdot P2 \cdot G1 + P3 \cdot P2 \cdot P1 \cdot G0$$

De esta manera es posible encadenar la construcción de un sumador de 16 bits con cuatro sumadores de 4 bits y la utilización de nuevo del módulo cálculo de acarreos adelantados, tal como se muestra en la figura 3:

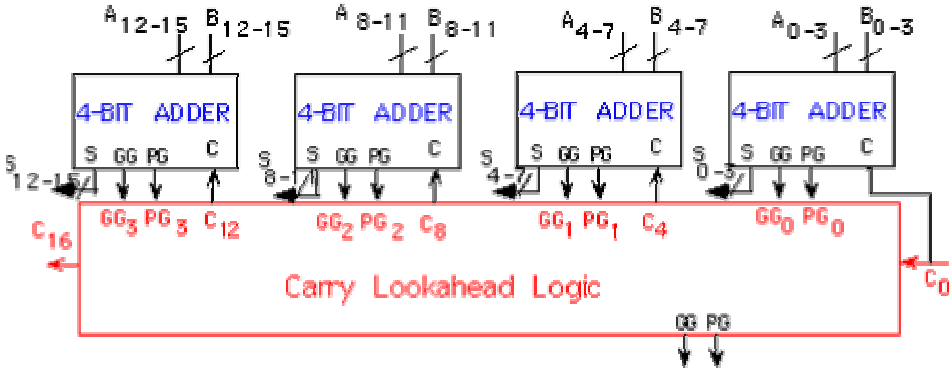

Figura 3: *Sumador de 16 bits con acarreo adelantado*

2 Trabajo de laboratorio

En el trabajo de laboratorio se propone sintetizar el sumador de 16 bits con acarreo adelantado en una FPGA. Para ello se va a utilizar el entorno de síntesis de PLDs Max+Plus II.

En primer lugar se van a describir en VHDL los módulos: sumador parcial completo (PFA) y la lógica de cálculo adelantado de los acarreos (ACARREOS) que aparecen en la figura 2. A partir de la descripción VHDL de estos módulos se generarán automáticamente sendos símbolos que servirán para montar un esquema del sumador con acarreo adelantado de 4 bits que aparece en la figura 2. A partir de este sumador se 4 bits se generará un símbolo con la intención de construir un sumador de 16 bits, tal como aparece en la figura 3.

2.1 Los módulos VHDL

El primer paso es invocar la herramienta Max+plus II y abrir un nuevo fichero de texto para realizar la descripción del PFA. De esta manera el bloque que describe el sumador parcial completo tendrá un aspecto como el que sigue:

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY pfa IS
     PORT(A, B, C : IN std_logic;
          S, G, P : OUT std_logic);
END pfa;

ARCHITECTURE flujo OF pfa IS
….
```

Una vez completada la descripción VHDL se debe salvar el fichero **con el mismo nombre** que la entidad. En este caso `pfa.vhd` y en un directorio de trabajo dentro de `C:\temp`, directorio donde se guardará todo el trabajo de la sesión. Se debe hacer notar que se debe modificar la extensión automática que por defecto es `.tdf`. A partir de ese instante el editor colorea las palabras clave VHDL, con lo que la mejora la legibilidad del código.

Es importante recordar el concepto de proyecto en Max+plus. Todas las acciones (síntesis, simulación, selección de opciones…) se realizan siempre sobre el proyecto seleccionado con lo que se debe asegurar que se está trabajando con el proyecto correcto. El nombre del proyecto aparece en la parte superior más externa de la ventana de Windows, después del logotipo y del nombre de MAX+plus II. Al inicio la herramienta recuerda el último proyecto activo con que se cerró, con lo que se debe cambiar el proyecto al empezar cada nuevo símbolo o nivel jerárquico. Para ello se dejará activa la ventana con el `pfa.vhd` y se seleccionará:

- *File>Project>Set Project to current file*

El nombre del fichero, con el camino completo, debe aparecer en la parte superior de la ventana junto al logotipo y nombre de la herramienta. Ahora ya se está en disposición de generar el símbolo que representará al sumador parcial completo mediante:

- *File>Create default symbol*

Esta acción lanzará el compilador de la herramienta que verificará que la sintaxis es correcta y que no hay errores en la descripción. Si no hay ningún problema aparecerá un mensaje informando del éxito de la operación, a pesar de que el símbolo no aparece inicialmente en la pantalla. A continuación se debe utilizar la misma metodología para la lógica de cálculo de acarreo adelantado. De nuevo hay que hacer notar que el nombre de la entidad se debe llamar igual que el fichero y que para generar el símbolo se debe cambiar primero el proyecto que será en este caso el nuevo fichero VHDL.

2.2 El sumador de 4 bits con acarreo adelantado

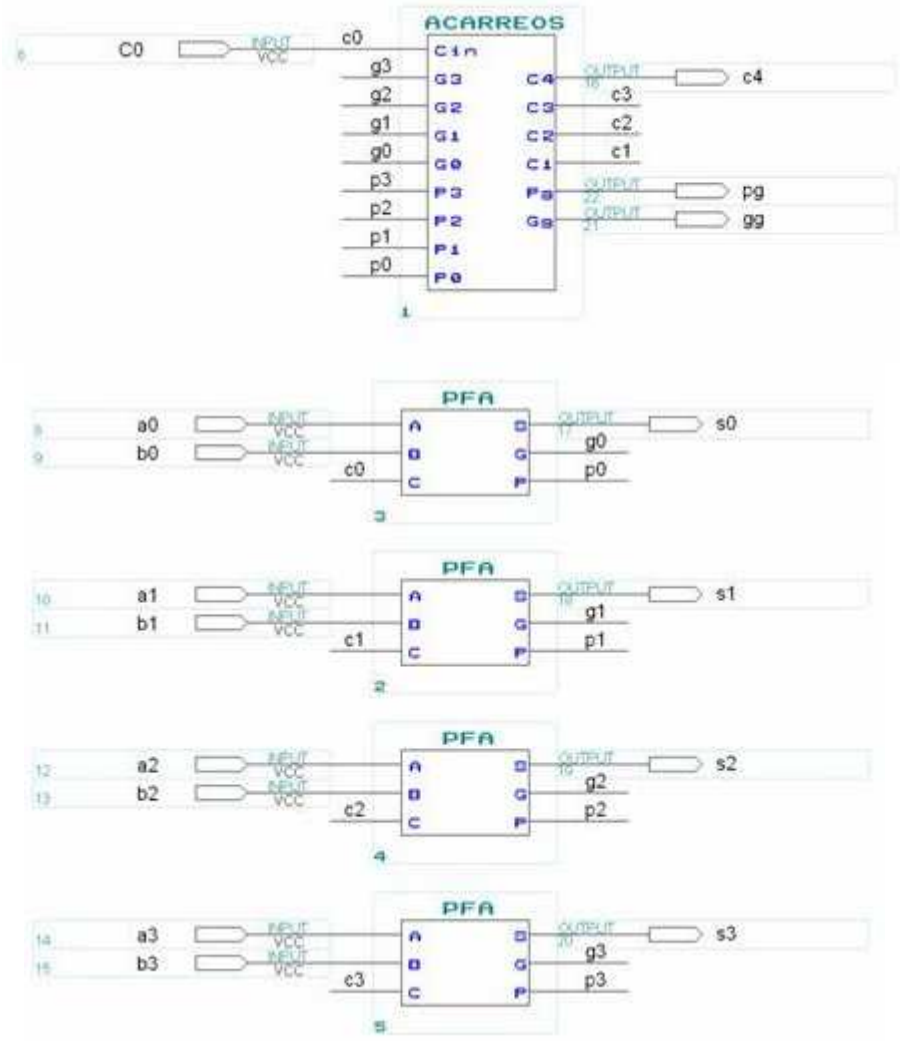

Figura 4*: Esquema del sumador de 4 bits con acarreo adelantado*

Ahora ya se está en disposición de crear el sumador de 4 bits con acarreo adelantado que englobará los módulos antes diseñados. Para ello es necesario abrir una nueva carpeta de esquema con el editor de esquemas. Para añadir los componentes simplemente se pinchará dos veces consecutivamente sobre la pantalla para que aparezca la ventana *Enter symbol*. Los símbolos generados aparecerán en la ventana Symbol files. Adicionalmente se añadirán los pines de entrada y salida `INPUT` y `OUTPUT` de la librería `prim`. Es interesante hacer notar que para evitar un cableado excesivo y mejorar la legibilidad del esquema se ha optado simplemente por añadir cables y etiquetas para conectar entre si los nodos.

Se debe salvar el esquema con:

- File>*Save as*

Escogiendo un nombre razonable como *sum_4_CLL*. A partir de este esquema se debe generar un nuevo bloque que servirá para construir el sumador de 16 bits con acarreo adelantado. De nuevo para generar el símbolo hay que indicar que el proyecto sea el sumador de 4 bits realizado dejando activa la ventana del esquema y seleccionando:

- *File>Project>Set Project to current file*

Ahora ya se está en disposición de generar el símbolo que representará al sumador de 4 bits con lógica de acarreo adelantado:

- *File>Create default symbol*

2.3 El sumador de 16 bits con acarreo adelantado

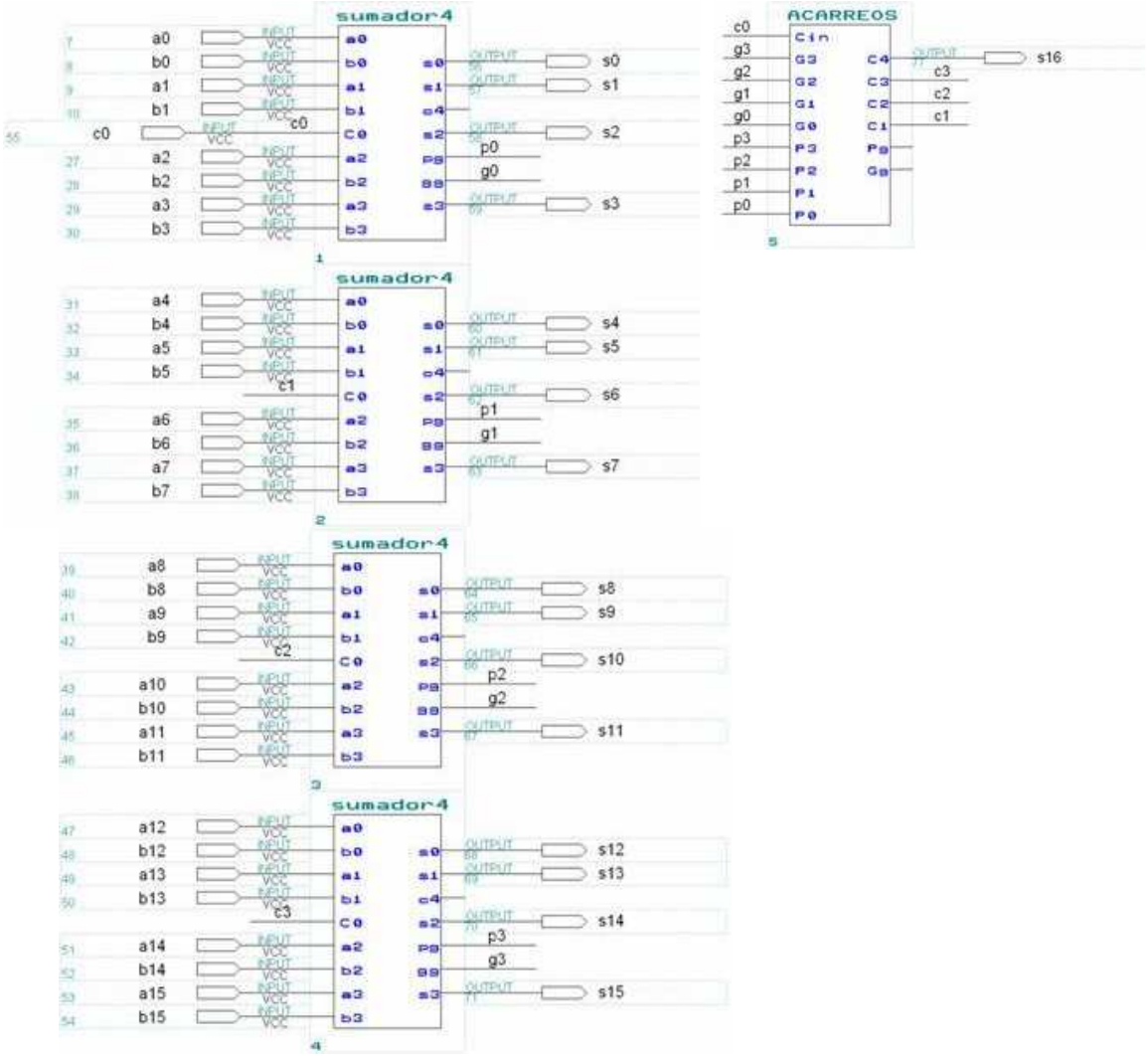

Figura 5*: Implementación en ORCAD del sumador de 16 bits con acarreo adelantado*

Construir el sumador de 16 bits con acarreo adelantado es inmediato a partir de los bloques ya creados y de los procesos descritos. Para ello se debe crear un esquema nuevo que implemente el sumador de la figura 3. De nuevo es aconsejable utilizar la

facilidad de nombrar los nodos para conectar los sumadores de 4 bits y el módulo de cálculo de acarreo adelantado entre si.

2.4 Síntesis

Se propone sintetizar el sumador en una FPGA de la familia Flex6000 de Altera. Para ello se va a seleccionar la opción

- *Assign>Device*

Aparecerá una ventana de diálogo donde se escogerá la familia Flex6000, sin escoger ninguna FPGA concreta (hay que dejar la opción AUTO en *Devices* para que escoja el dispositivo más pequeño donde quepa). Posteriormente se validará la ventana y se lanzará el sintetizador.

3 Análisis comparativo de resultados

Una vez la síntesis se ha realizado con éxito se deben observar los 2 parámetros más importantes de cualquier síntesis en un dispositivo programable: la cantidad de lógica empleada y la velocidad de funcionamiento. Para observar los recursos empleados se puede acceder al informe de la síntesis directamente desde la ventana del compilador pinchando dos veces sobre el icono con las siglas `rpt.` Es importante anotar el dispositivo escogido, el número absoluto y porcentual de LCs (Logic Cells) utilizadas.

Para analizar la velocidad del sumador, al ser un módulo plenamente combinacional, se debe analizar la velocidad de propagación desde todas las entradas a todas las salidas y ver cual es el peor caso. Afortunadamente Max+plus II calcula la matriz de retrasos automáticamente a partir de la síntesis realizada. Para ello se debe seleccionar:

- *Max+plus II>Timing Analyzer*

En realidad para poder comparar las prestaciones del sumador sintetizado se hace necesario implementar otras opciones. Para ello se propone implementar y sintetizar (en el mismo dispositivo que se ha utilizado antes)

1. **Sumador trivial VHDL**: Para ello simplemente se sintetizará la descripción VHDL de la suma aritmética de dos vectores utilizando el paquete *unsigned* de la librería *std_logic*:

   ```
   USE ieee.std_logic_unsigned.all;
   ```

2. **Sumador con acarreos encadenados**: En este caso se describirá el bloque sumador completo de 1 bit y se montará un esquema encadenando 16 de estos sumadores.

Se deben comparar los resultados de la síntesis para todos los casos y explicar aparentes resultados contradictorios. Adicionalmente se pueden probar opciones de síntesis en Max+plus II.