DISEÑO AVANZADO DE SISTEMAS DIGITALES Y MICROPROCESADORES

PRÁCTICA 4: División binaria

La siguiente figura representa un algoritmo sin restauración para el desarrollo de la división de dos números binarios (7 bits para el dividendo y 4 bits para el divisor) basado en la realización de desplazamientos y operaciones de sumas y restas.

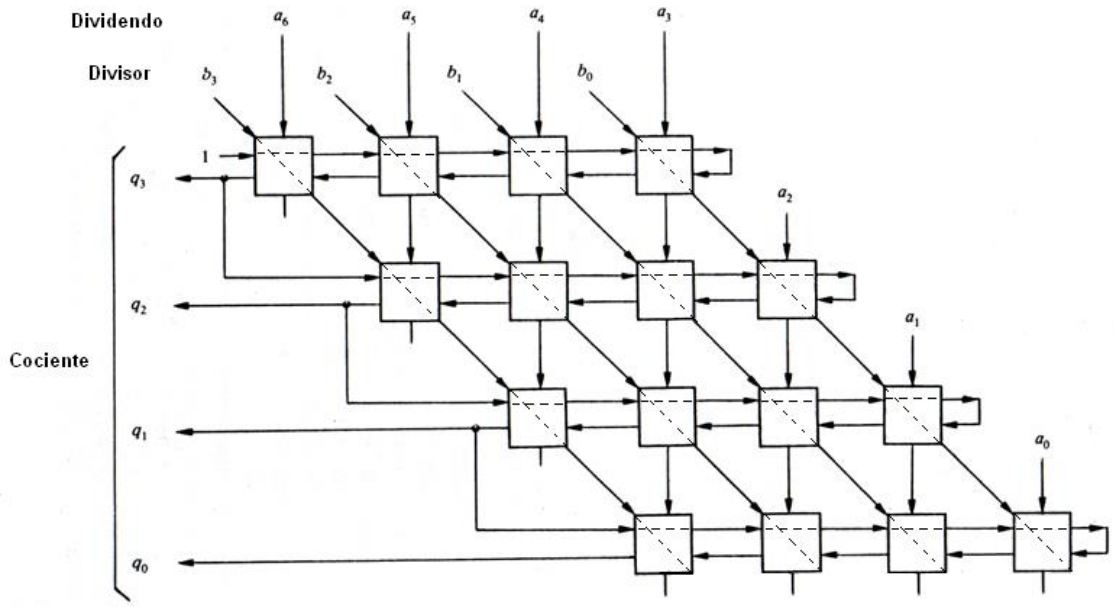

Figura 1.- Algoritmo de división sin restauración para un dividendo de 7 bits y y un divisor de 4 bits

La celda que se repite en la figura anterior corresponde a un sumador/restador de un bit con acarreo de entrada y de salida

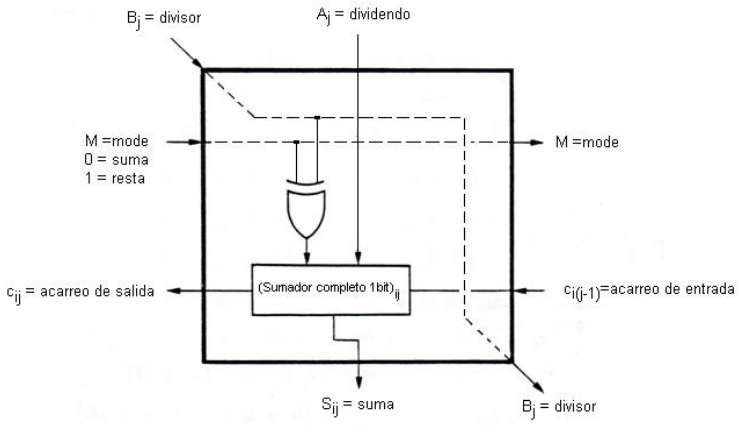

Figura 2.- Celda del algoritmo de división sin restauración

Este algoritmo puede ser resumido como se describe a continuación:

- En primer lugar, se resta el divisor al dividendo: $A - B = A + \overline{B} + 1$. De esta operación se obtiene el primer resto parcial.
- Si el acarreo de salida obtenido en esta primera fase ha sido cero es porque el resto parcial fue negativo. Por tanto, el bit más significativo del cociente es cero, el divisor es desplazado hacia la derecha, y una operación de suma es realizada en el siguiente nivel.
- Si el acarreo de salida obtenido en esta primera fase ha sido uno es porque el resto parcial fue positivo. Por tanto, el bit más significativo del cociente en este caso es uno, el divisor es desplazado hacia la derecha, y una operación de resta es realizada en el próximo nivel.

Desarrollad en VHDL el algoritmo de división anterior basándose en la descripción de la celda de la figura 2. Como guía de descripción, bastaría con declarar unas señales S y c de interconexión de tipo array de manera que la interconexión de un módulo con otro sería tal y como aparece en la figura 1. Para facilitar la tarea de descripción, resulta recomendable realizar la declaración basándose en sentencias **GENERATE** que realicen de forma general el interconexionado. Estas sentencias utilizadas para generar estructuras generales de conexión con un estilo propio y parametrizable, sirven para instanciar componentes idénticos bajo el control de un índice o de una condición. Con este tipo de sentencias, expandir el algoritmo de división a cualquier número de bits resultaría inmediato. A continuación se muestra la sintaxis de los dos tipos de sentencias **GENERATE** que soporta el lenguaje:

- **SENTENCIA GENERATE TIPO IF**

```
<LABEL_1>:
 if <condition> generate
  <statement>;
 end generate;
```

- **SENTENCIAS GENERATE TIPO FOR**

```
<LABEL_1>:
 for <name> in <lower_limit> to <upper_limit> generate
  <statement>;
 end generate;
```

Por ejemplo, una batería de puertas and se modelaría del siguiente modo:

```
-- A, B y C están declaradas como STD_LOGIC_VECTOR (N downto 0)
Puertas_and: for i in 0 to N generate
                C(i)<=A(i) and B(i);
                end generate;
```

Como se observa, en ambos tipos de sentencias es obligatorio incluir una etiqueta. La sentencia **FOR** se utiliza para generar la estructura regular, mientras que la sentencia **IF** es utilizada para especificar excepciones a la regla de conexión. En la zona de descripción puede

haber no sólo estancias de emplazamiento de instancias de componentes, sino cualquier tipo de sentencia concurrente.

Finalmente crear un banco de estímulos asociado al diseño y descrito en VHDL que permita verificar el correcto funcionamiento del algoritmo de división desarrollado:

Tiempo	Dividendo	Divisor
0 ns	6	3
200 ns	14	2
400 ns	16	4
600 ns	20	4
800 ns	21	6
1000 ns	17	3