ETSIDI

TRABAJO SISTEMAS ELECTRÓNICOS DIGITALES

MULTIPLICADOR N BITS EN COMPLEMENTO A 2

Gustavo Puig Baeza (49795) | Emilio Martínez Míguez (48938) | Carlos Rodriguez Diez (49796)

15/01/2014

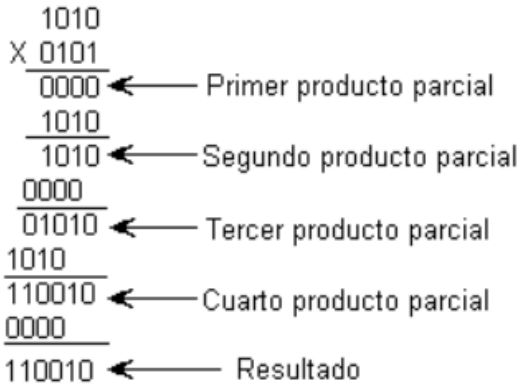
0.- CONCEPTOS PREVIOS A LA PROGRAMACIÓN VHDL:

a) Multiplicación binaria.

En toda multiplicación se distinguen dos integrantes: el multiplicando (aquel sobre el que se realiza la operación) y el multiplicador (aquel que indica la tasa de cambio del multiplicando), es decir, el que es multiplicado y el que multiplica.

Un producto binario implica los mismos procedimientos que un producto convencional en base 10: cada dígito del multiplicador se multiplica individualmente con el multiplicando (a cada resultado se le llama multiplicación parcial). Cada una de estas parciales (coincide con la cantidad de dígitos del multiplicador), se sumarán posteriormente de una manera especial. Cada sumando se verá desplazado lógico a la izquierda con el anterior tantas posiciones como índice de producto parcial le corresponda.

El mismo resultado se obtendría si sumamos recursivamente cada producto parcial al anterior, en vez de sumarlos todos al final. Este hecho nos permite distinguir el sistema en sus funciones modulares como se ilustrará a continuación con un ejemplo:



Bajo los últimos criterios y procedimientos de operación enunciados cambiará nuestro concepto de producto parcial. En estas condiciones por tanto, el producto parcial será el resultado de la suma del producto de un dígito del multiplicador (i) con el multiplicando (i), con el anterior producto parcial (i-1).

Para empezar definiríamos que nuestro primer producto parcial es el '0' al que le sumaremos recursivamente los sucesivos (no olvidando el desplazamiento implícito en el grado del multiplicador que obliga a las sumas a correrse a la izquierda).

Como se observa en el ejemplo, multiplicar por '0' no tiene repercusión en el valor del producto parcial resultante. No así sucede con el producto por '1'.

De este comportamiento se desprende:

El sistema debería comprobar los bits del multiplicador, en el caso de que fuese '0', no modificaría el resultado, pero si fuese '1', se debe sumar el multiplicando al resultado desplazado a la izquierda correctamente. El proceso iteraría tantas veces como dígitos tenga el multiplicador.

b) Multiplicación binaria en complemento a 2:

El complemento a dos es una transformación que se le aplica a los números binarios para facilitar su tratamiento y su operabilidad principalmente en sistemas computacionales.

Sea un numero binario expresando su MSB como el bit de signo: '1' para números negativos '0' para los positivos.

La función de la complementación no es otra que realizar un pretratamiento sobre los números binarios negativos para facilitar operaciones como la resta, la multiplicación y la división.

En nuestro caso, el complemento a dos nos permite trabajar siempre con un producto de números positivos, de tal manera que antes de la operación, si algún operando es negativo, este se transforma, se opera, obteniéndose un resultado siempre positivo.

En este punto, el usuario debe plantearse, por inspección de los operandos sin complementar, si el resultado debiera ser positivo o negativo. Si la conclusión es la primera, el número obtenido sería el resultado de la operación, pero si por el contrario el resultado debiera ser negativo, el número obtenido debería ser complementado a dos para expresarlo con negatividad.

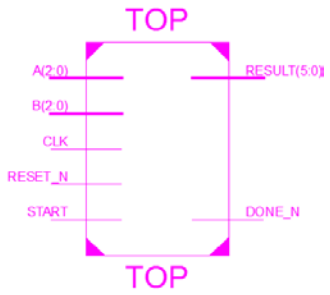
El complemento a dos presenta una ventaja principal frente a otros procedimientos de la misma naturaleza como el complemento a 1, y es que solo existe una única manera de expresar sus binarios positivos y negativos. En el complemento a 1, tenemos dos maneras de expresar el cero binario: como '0' positivo ("00…0") o como '0' negativo ("11..1"). Esto es así pues complementar a 1 no implica más que cambiar unos por ceros. Sin embargo, en el complemento a dos esto no es posible pues para llevarlo a cabo sobre un número, tenemos primero que complementar a 1 y al resultado sumarle 1 (binario) de tal manera que, el caso del "cero problema" se esfumaría, puesto que ahora 0="00…0" en ambos casos.

Por último se recuerda que la operación de complementación es reversible, aplicando el mismo procedimiento de nuevo sobre el número complementado que se pretende antitransformar.

1.- DESCRIPCIÓN DE LA SOLUCIÓN:

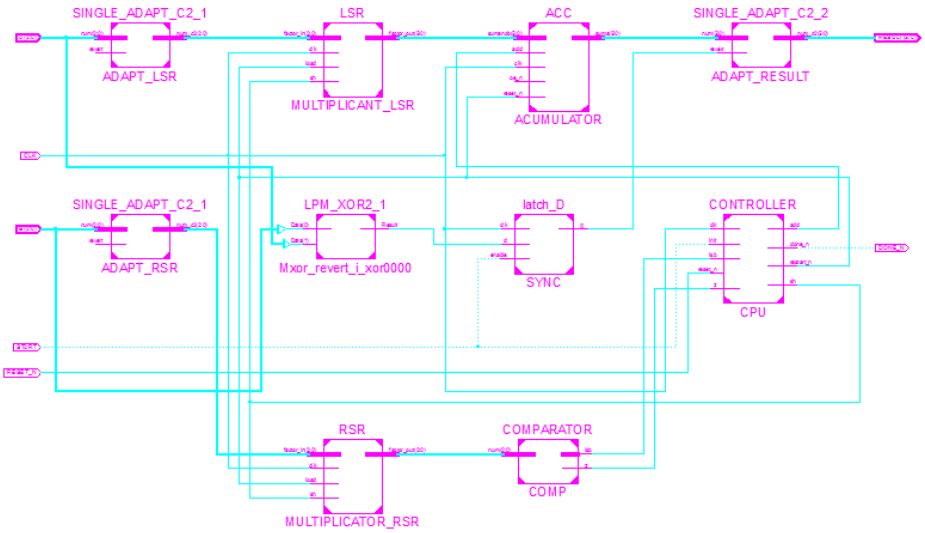
A continuación, intentaremos enunciar el funcionamiento del multiplicador de N bits que aportamos, así como su arquitectura de componentes internos y la descripción de cada uno de ellos.

El esquema de patillas de nuestro multiplicador sería el siguiente:



Como se puede observar dispone de dos patillas para operandos de entrada y una salida de resultado. Por otro lado dispone de una entrada START para iniciar la cuentra, un reset asíncrono a nivel bajo y una marca de salida de fin de cuenta DONE_N (también a nivel bajo).

El esquema de su composición interna sería el siguiente:



Donde se observa que se dispone de:

- Dos registros de desplazamiento, LSR y RSR.
- Un acumulador.
- Una etapa de comparación combinacional.
- Un control (o máquina de estados).
- Adaptadores combinacionales para la complementación a 2.
- Un biestable tipo D.

Los operandos, multiplicando y multiplicador, se pretenden cargar en los registros de desplazamiento LSR y RSR.

LSR nos ofrece en cada producto parcial de la multiplicación, el número desplazado a izquierdas adecuado en cada caso, el cual deberíamos sumar al anterior si el bit del multiplicador en la suboperación (i) fuese un ‘1’.

RSR ofrece el bit del multiplicador que estamos operando en cada suboperación. Esta adquisición de este bit no se produce directamente: el registro ofrece un número desplazado a derechas que se introduce en una etapa combinacional, que es la encargada de informar al control cuál es el bit que estamos operando (bit menos significativo) y si ha terminado la cuenta (cuando el registro de desplazamiento RSR haya desplazado todos los números y se encuentre cargado con ceros).

El acumulador, como su nombre indica, es el encargado de hacer la operación suma de productos parciales cuando el control lo estime apropiado. La entrada, que es el sumando del acumulador, corresponde con la salida del LSR.

Como ya se ha enunciado, el control es el encargado de interpretar la información que le llega del multiplicador, lanzando la orden de suma al acumulador, hacer que los registros corran o dar la operación por finalizada. También tiene como función coordinar la carga y reseteo de todos sus componentes.

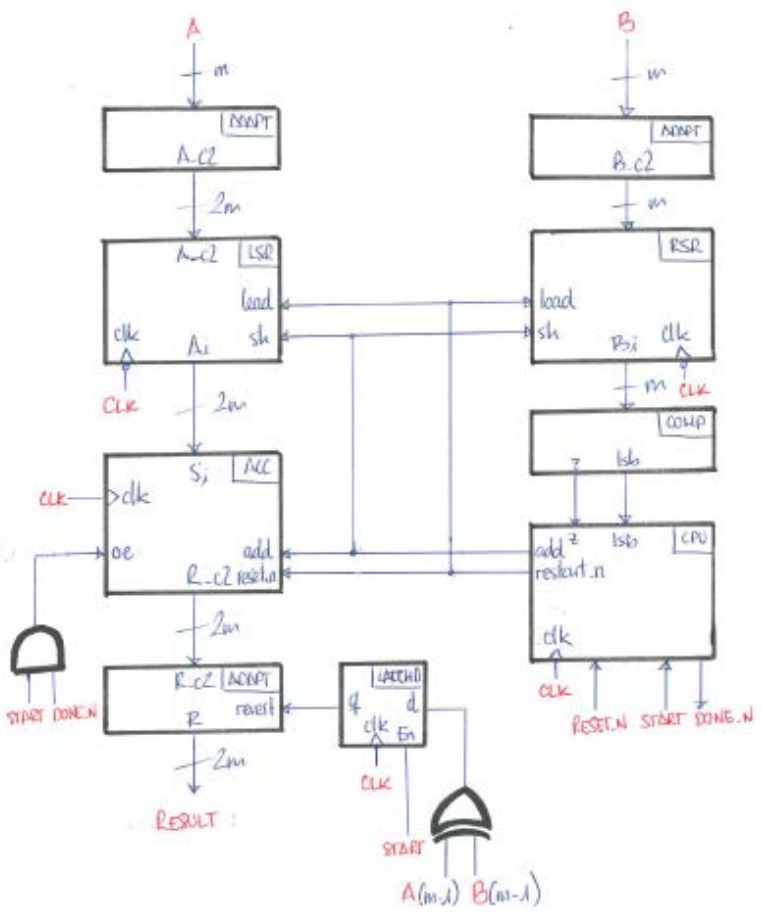
Para la complementación se ha optado por aplicar un equivalente al patrón adaptador GoF. Para ello se han incluido tres transformadores a complemento a dos individuales para tratar las dos entradas y la salida. Esto nos ofrece bajar el acoplamiento de nuestro diseño, de tal manera que si en posteriores requerimientos se necesitase un multiplicador con cualquier otra complementación, solo fuese necesario cambiar estos adaptadores, viéndose el control del sistema inalterado.

La inclusión del biestable tipo D tiene dos razones de ser:

1.- Sincronizar la parte secuencial de adquisición de resultados, con la parte combinacional de muestreo.

2.- Independizar las entradas de una influencia directa sobre la salida. Para ello se incorpora un enable en el biestable de tal manera que si las entradas al sistema se cambiasen durante una operación, la entrada revert al adaptador del resultado (que le informa de si tiene que antitransformar a negativo o no) no se viese afectada con respecto a los operandos del momento del lanzamiento del sistema, hasta que una nueva operación fuese solicitada.

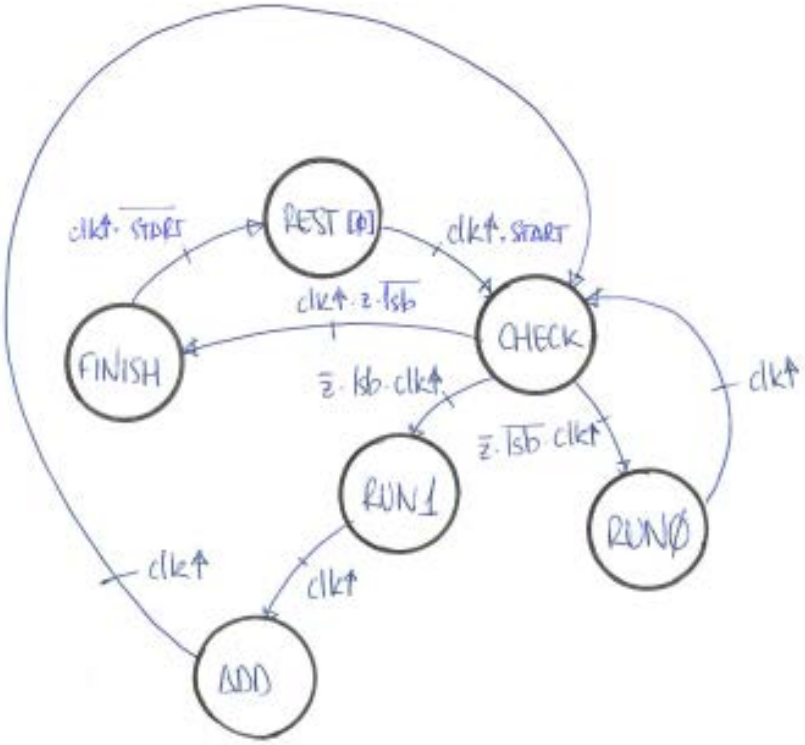
1.2.- DESCRIPCIÓN ESQUEMATICA DE LA ARQUITECTURA INTERNA:



2.2.- ENTRADAS/SALIDAS DE COMPONENTES:

- Adaptador C2 LSR:
  - **B:** multiplicando en complemento a 2.
  - **Revert:** '0'.
  - **B_c2:** multiplicando complementado si negativo.
- Registro LSR:
  - **A_in:** multiplicando de entrada
  - **Load:** señal de carga
  - **Sh:** señal de desplazamiento.
  - **CLK:** reloj del sistema
  - **Ai_out:** multiplicando.
- Acumulador ACC:
  - **S_in:** sumando de entrada
  - **Output_Enable_n:** habilita la salida.
  - **CLK:** reloj del sistema.
  - **Add:** señal de suma.
  - **Reset_n:** señal de reinicio de suma.
- Adaptador C2 ACC:
  - **R:** resultado complementado.
  - **Revert:** señal de sincronía.
  - **R_c2:** Resultado se complementa solo si uno de los factores tiene bit de signo 1.
- Latch D:
  - **D:** A(n-1) xor B(n-1), uno y solo un bit de signo negativo.
  - **CLK:** reloj del sistema
  - **Enable:** START
  - **Q:** actua como señal de sincronía.
- Adaptador C2 RSR:
  - **A:** multiplicador en complemento a 2.
  - **Revert:** '0'.
  - **A_c2:** multiplicador complementado si negativo.
- Registro RSR:
  - **B_in:** multiplicador de entrada en complemento a 2.
  - Load: señal de carga del vector de bits al resgistro.
  - **Sh:** señal de desplazamiento.
  - **CLK:** reloj del sistema.
  - **Bi_out:** multiplicador en la iteración i.
- Comparador COMP:
  - **Factor_in:** multiplicador en la iteración i.
  - **Z:** bit de fin de operación.
  - **LSB:** bit menos significativo (bit multiplicador).
- Controlador CPU:
  - **START:** inicio.
  - **RESET_N:** apagado.
  - **CLK:** reloj del sistema
  - **Z:** bit de fin de operación.
  - **Lsb:** bit menos significativo (bit multiplicador).
  - **Add:** señal de adición.
  - **Restart_n:** señal de carga si '1' y reseteo suma si '0'.
  - **DONE_N:** señal de fin de operación.

2.3.- DIAGRAMA DE ESTADOS DEL SISTEMA:



REST [estado 0]:

done_n = 1
sh = 0
restart_n = 0
add = 0

CHECK:

done_n = 1
sh = 0
restart_n = 1
add = 0

RUN 0:

done_n = 1
sh = 1
restart_n = 1
add = 0

RUN 1:

done_n = 1
sh = 0
restart_n = 1
add = 1

ADD:

done_n = 1
sh = 1
restart_n = 1
add = 0

FINISH:

done_n = 0
sh = 0
restart_n = 1
add = 0

2.5.- CONTROL DE VERSIONES: GITHUB

Para el desarrollo del proyecto hemos utilizado la herramienta GitHub en la que se han almacenado en un repositorio el progreso del mismo.

El repositorio puede consultarse en la dirección:

https://github.com/GustavoPB/MULTIPLICADOR_Nbits.git

El archivo terminado responde al nombre del último commit:

MULTIPLICADOR NBITS v3.1. (vSync)