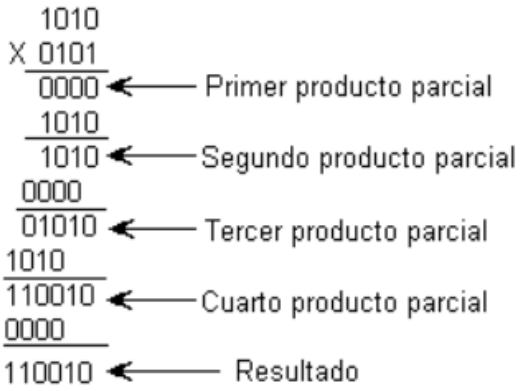
MULTIPLICADOR_NbiTS:

Como coordinador del trabajo me dispongo a afianzar las bases del mismo con este documento. El fin no es sólo este: lo conveniente es que lo leáis con atención y lo discutamos juntos. Por tanto, según el papel que cumplo, una posible organización del trabajo puede ser la siguiente:

CONCEPTOS PREVIOS A LA PROGRAMACIÓN VHDL: 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 aritméticamente 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 cambiará nuestro concepto de producto parcial. En estas condiciones, el producto parcial será el resultado de la suma del producto de un dígito del multiplicador con el multiplicando, con el anterior producto parcial.

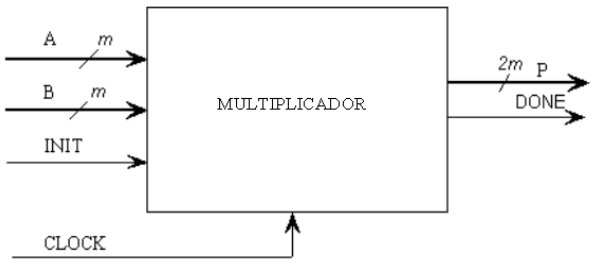
Para empezar definiríamos que nuestro primer producto parcial es el ‘0’ al que le sumaremos recursivamente los sucesivos (no olvidando el desplazamiento implícito en el grado del multiplicador que obliga a las sumas a desplazarse).

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.

A priori podemos definir ya una interfaz para nuestra solución final:



INIT: marca de inicio.

DONE: marca de fin de multiplicación.

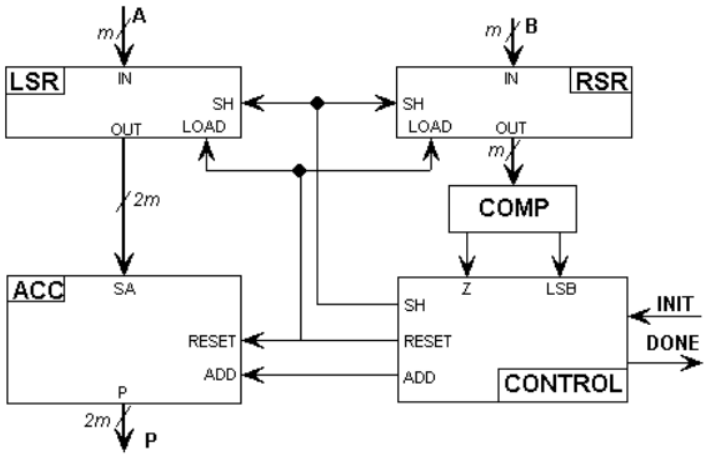
MUDULARIZACIÓN DE LA SOLUCIÓN:

Lo lógico es pensar en dos registros de desplazamiento, uno para el multiplicando y otro para el multiplicador. Su fin no es hacer el producto dígito por dígito, si no:

Registro de del multiplicador: lo recibe y devuelve el mismo número desplazado convenientemente. Su objetivo es proporcionarle a un centro de control (gracias a un circuito combinacional intermediario) si toca multiplicar por ‘0’ o por ‘1’ (informándole del bit menos significativo del registro) y si el proceso iterativo a recorrido todos los dígitos del multiplicador. Se podría decir que se trata de un registro de corrimiento a derechas con salida y entrada paralela.

Registro del multiplicando: lo recibe y entrega en paralelo el multiplicando desplazado convenientemente para ser sumado en una unidad de acumulación, en el caso de que la unidad de control interpretara que existe un ‘1’ en el multiplicador en la posición i. Se podría decir que se trata de un registro de corrimiento a izquierdas con salida y entrada paralela. Cabe destacar que el rango de salida del registro debe ser de n=2m donde m corresponde con el numero de bits de entrada al multiplicador.

Por tanto, una estructura modular del proyecto puede ser el siguiente:



CONTROL :

Z: recordad que la inserción en un registro de desplazamiento es un '0' serie, en el caso del RSR por la izquierda, lo que implica que, cuando todo el multiplicador haya sido leído, es decir, el número se ha desplazado tantas veces como bits tenía, la salida paralela debe ser cero. Este es el destino de Z, informar de cuando la salida es cero, y por tanto la multiplicación ha llegado a su fin.

LSB: informa al control si toca sumar en el acumulador (ACC) o no.

Toda esta información es proporcionada al control mediante un sistema combinacional intermediario.

SH: manda pulsos que obligan a los registros a desplazarse simultáneamente.

ADD: informa al acumulador de cuando debe realizar la suma.

RESET: tal y como cualquiera convencional.

El resultado corresponde con la salida del acumulador cuando DONE esta activo.